

Método de Treinamento

Residência Tecnológica

Projeto:

**Modelo de linguagem para consulta e análise de normas técnicas do 3GPP**

Residentes:

**Alessandra Aparecida Toyama**

**Beatriz de Brito**

**Paulo Henrique Bessani Salkys**

09/04/2024

**Histórico de Revisões**

|  |  |  |
| --- | --- | --- |
| **Data** | **Versão** | **Descrição** |
| 09/04/2024 | 1.0 | Versão inicial do documento |

**Conteúdo**

[**1. Introdução 3**](#_cshhsezgyaxu)

[**2. Método de Treinamento 3**](#_b0cjpqewwp87)

[2.1. Código para treinamento do Modelo 4](#_uvkab0afj1l4)

[2.2. Manual para treinamento do Modelo 4](#_5o976jjtfnvo)

2.2.1. Visão Geral do Processo de Treinamento 4

2.2.2. Estrutura de Diretórios, Segredos e Arquivos do Projeto 5

2.2.3. Premissas e Requisitos para Execução do Treinamento 5

2.2.4. Execução do Código para Treinamento do Modelo 5

2.2.5. Comandos, Parâmetros e Configurações 6

Referências 7

# Introdução

Este relatório expõe o método de treinamento adotado para modelos de inteligência artificial, incluindo uma descrição do código específico utilizado, além de um manual que detalha o processo e as ferramentas aplicadas. Destacamos a utilização do Gemma, uma série de modelos abertos e avançados, desenvolvidos com base nos modelos Gemini do Google, que, nos testes realizados para nossas pesquisas até o momento, se mostrou muito promissor.

# Método de Treinamento

Este material destaca os testes que realizamos com o método de treinamento adotado, integrando o Gemma com KerasNLP para o desenvolvimento de modelos de processamento de linguagem natural (NLP). Gemma, caracterizada por ser uma família de modelos abertos, leves e de última geração, é desenvolvida com a mesma base tecnológica e de pesquisa que os modelos Gemini. Por sua vez, KerasNLP oferece um conjunto de modelos NLP implementados em Keras, compatíveis com JAX, PyTorch e TensorFlow, proporcionando uma plataforma versátil para o treinamento e a implementação de modelos NLP.

Através deste tutorial, o foco é demonstrar nossos testes ao utilizar o Gemma, em conjunto com KerasNLP, para gerar respostas de texto a diversos prompts, explorando a eficácia e a flexibilidade dessa integração no treinamento de modelos de NLP. Além disso, outro objetivo alcançado foi a obtenção de conhecimentos sobre Keras; este material proporciona um ponto de partida para aqueles que estão iniciando, com a intenção de um aprendizado progressivo sobre Keras e outras ferramentas.]

## Código para treinamento do modelo

O código necessário para o treinamento do modelo desenvolvido está disponível no link abaixo. Neste repositório está o notebook em Python com todas as descrições que fornecem os detalhes e requisitos para a implementação e experimentação com o modelo, juntamente com este manual, conforme solicitado para entrega.]

* link

## Manual para treinamento do modelo

2.2.1 Visão Geral do Processo de Treinamento

Para apresentar uma visão geral do processo de treinamento do modelo desenvolvido com Gemma, iniciamos destacando que Gemma faz parte de uma família de modelos de linguagem grande (LLMs) de última geração, beneficiando-se das pesquisas e tecnologias empregadas nos modelos Gemini. Estes modelos são reconhecidos por sua eficácia em várias tarefas de Processamento de Linguagem Natural (NLP), graças a um processo de pré-treinamento auto-supervisionado em extensos corpus textuais. Este pré-treinamento permite que os LLMs adquiram conhecimentos de uso geral, identificando relações estatísticas entre palavras.

Para ajustar esses modelos a tarefas específicas de domínio, como a análise de sentimentos, adotamos técnicas que não requerem a atualização de todos os bilhões de parâmetros. Uma dessas técnicas é a Low Rank Adaptation (LoRA), que minimiza o número de parâmetros treináveis para tarefas subsequentes, fixando os pesos originais do modelo e introduzindo um número reduzido de novos pesos. Essa abordagem não apenas agiliza o processo de treinamento e o torna mais eficiente em termos de uso da memória, mas também resulta em modelos mais leves, com os pesos finais ocupando apenas algumas centenas de megabytes, sem comprometer a qualidade do modelo.

Para acomodar o uso limitado da GPU gratuita disponível, o período de treinamento e outros parâmetros foram ajustados para otimizar o uso dos recursos. Assim, o modelo foi treinado por apenas 3 épocas com um tamanho de batch de 1 sobre o conjunto de dados Databricks Dolly 15k. Este conjunto, composto por 15.000 pares de prompt/resposta de alta qualidade gerados por humanos, concebidos para otimizar o ajuste fino de LLMs.

2.2.2 Estrutura de Diretórios, Segredos e Arquivos do Projeto.

O código desenvolvido para executar o modelo Gemma requer um runtime do Colab equipado com recursos adequados, incluindo a necessidade de uma GPU T4 para seu funcionamento eficiente. Para configurar este ambiente, acesse as Opções de conexão adicionais, localizadas no canto superior direito da janela do Colab, selecionando o ícone ▾. Em seguida, escolha "Alterar tipo de runtime" e, na seção Acelerador de hardware, selecione a opção "T4 GPU".

2.2.3 Premissas e Requisitos para Execução do Treinamento.

A preparação necessária para executar o treinamento, incluindo a instalação de dependências, configuração do ambiente e o próprio código, está minuciosamente detalhada e comentada no código Python do notebook que disponibilizamos. Este guia completo, referenciado na seção anterior 2.1, fornece instruções passo a passo para assegurar que usuários de todos os níveis possam configurar corretamente o ambiente necessário para o treinamento do modelo, facilitando um processo de testes.

2.2.4 Execução do Código para Treinamento do Modelo.

Para realizar o treinamento do modelo conforme planejado, é essencial seguir as instruções detalhadas nos comentários que precedem cada trecho de código no notebook, mencionado anteriormente na seção 2.1. Estes comentários fornecem orientações específicas, garantindo que cada etapa do processo de treinamento seja executada corretamente e sem erros.

2.2.5 Comandos, Parâmetros e Configurações.

O resumo apresentado pelo comando gemma\_lm.summary() fornece uma visão abrangente dos comandos, parâmetros e configurações disponíveis para o modelo Gemma. O modelo totaliza 2,506,172,416 parâmetros treináveis, equivalentes a 9.34 GB, todos configuráveis e ajustáveis conforme necessário. O resumo detalhado na figura abaixo demonstra a estrutura interna e a capacidade do modelo além de outras informações de camada, facilitando a compreensão e a manipulação de suas configurações para treinamento e aplicação específicos.

Figura 1 – Saida do comando “gemma\_lm.summary()”

Tela de computador com fundo verde

Descrição gerada automaticamenteFonte: Elaborada a partir dos testes realizados pelos autores.

## Referência