Treinamento de modelo LLM fundacional para a descricão de produtos da Amazon

1.Quanto ao objetivo

O objetivo do finetuning é treinar um modelo LLM fundacional com um dataset de produtos da Amazon, com a finalidade de que o modelo possa descrever os produtos a partir do título do mesmo.

2.Quanto à estrutura do projeto

Além deste relatório, o projeto contém os seguintes arquivos:

- dataset_preparation.ipynb = Notebook responsável pela preparação do dataset para posterior uso no fine tuning e no RAG.
- model_import.ipynb = Notebook responsável por realizar o download do modelo base e armazená-lo em disco.
- fine_tuning.ipynb = Notebook responsável pelo processo de fine tuning do modelo base.
- model_testing.ipynb = Notebook utilizado para comparar o modelo base com o modelo treinado, realizando um conjunto de perguntas para cada um deles.
- RAG.ipynb = Notebook responsável pelo processo de RAG, realizado utilizando o modelo treinado.
- evaluation_history.pdf = Arquivo contendo a comparação das respostas em cada etapa do processo de fine tuning e, depois, com o modelo final utilizando RAG.

3.Quanto ao modelo base escolhido

O foundational model escolhido para treinamento foi o GOOGLE/FLAN-T5, uma variante mais recente e aprimorada do T5, disponível inclusive para uso comercial. O modelo foi pré treinado com o objetivo de mapear sequências de texto, com bases supervisionadas e não supervisionadas.

O modelo está disponível em 5 variações de tamanho, sendo elas em ordem crescente: small, base, large, xl e xxl. Os modelos variam de 80 milhões de parâmetros à 11 bilhões. Por limitações de hardware para treinamento, o modelo escolhido para uso foi o

base, pois a partir do large o hardware disponível se mostrou inviável. A versão base do modelo possui 248 milhões de parâmetros.

4.Quanto ao dataset de treinamento

Os dados foram tratados de forma a buscar a otimização do treinamento, a limpeza foi realizada removendo as linhas que pussuíssem alguma das colunas (title ou content) sem conteúdo. As demais colunas foram também removidas. O texto foi tratado removendo entidades html, que alguns registros possuíam, e removendo caracteres especiais. Para preservar o contexto do texto, letras maiúsculas e pontuação convencional (. , ! ?) foram mantidas.

O dataset tratado, com aproximadamente 1.3 milhões de registros, foi dividido em datasets menores com 20 mil registros cada, com a finalidade de facilitar o treinamento, visto que o tempo e consumo de hardware para treinar com o dataset inteiro de uma única vez seria muito grande e inviável para o hardware disponível.

5.Quanto ao processo de fine tuning (treinamento)

O treinamento foi realizado iterativamente sobre os subdatasets de 20 mil registros, cada um com 3 épocas de treinamento. Cada subdataset treinado gerou um novo modelo, que foi usado para treinar com o próximo dataset.

O imput foi estruturado da seguinte maneira: Please describe this product: {título do produto}, com a descrição do produto fornecida como resposta.

O Google Colab foi utilizado para treinamento, com GPU Nvidia L4, disponível de forma paga na plataforma.

5.1.Quanto aos detalhes de implementação

O fine tuning foi realizado utilizando a biblioteca transformers (huggingface), utilizando T5Tokenizer para tokenização, e T5ForConditionalGeneration para manusear o modelo. Para o treinamento foi utilizado a classe Seq2SeqTrainer, a qual é focada em atividades sequence-to-sequence, como sumarização e tradução.

Os hiperparâmetros mais relevantes optados durante o processo de treinamento são descritos abaixo:

- LEARNING_RATE: Aqui buscou-se no valor de 1e-4 (0.0001) um equilíbrio. Como o modelo é relativamente pequeno e não possui uma quantidade grande de parâmetros, ele deve aprender bem com os novos dados, mas a taxa não deve ser grande ao ponto de bagunçar o que o modelo já sabe.

- PER_DEVICE_TRAIN_BATCH_SIZE: Optou-se pelo valor de 8, para que o treinamento não ficasse lento demais, e pudesse ser realizado com a memória disponível.

- GRADIENT_ACCUMULATION_STEPS: Utilizado para simular um batch maior, interessante quando se possui limitação de memória.

- WEIGHT_DECAY: Utilizado o valor de 0.01 como medida para prevenção de overfitting do modelo, aplicando penalidade aos pesos do modelo.

- NUM_TRAIN_EPOCHS: Para o caso foi utilizado 3 épocas de treinamento, visto que a progressão é lenta, e um maior número e épocas demandaria maior capacidade computacional.

5.2.Quanto à execução

Cada subdataset de 20.000 registros foi treinado em 3 épocas, sendo 5% (1.000) para teste e o restante para treinamento(19.000).

Para cada treinamento foi exibido o training loss, validation loss e o rouge score (Recall-Oriented Understudy for Gisting Evaluation), que calcula a eficiência do modelo para tarefas de question/answer, quanto maior o score, melhor a performance do modelo.

Por motivos de limitação de hardware/custo, não foi possível realizar o treinamento do dataset de forma total, foram treinados 10 subdatasets, correspondendo à um total de 200.000 registros.

Abaixo segue a execução de cada um dos subdatasets treinados:

Subdataset 1:

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	Rougel	Rougelsum
0	3.578700	3.193809	0.107531	0.026988	0.088663	0.096138
1	3.432500	3.143379	0.108257	0.031477	0.090435	0.097658
2	3.377900	3.127922	0.109883	0.031810	0.091686	0.099655

Subdataset 2:

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	Rougel	Rougelsum
0	3.478000	3.166302	0.102071	0.018460	0.079640	0.089572

1	3.409200	3.139682	0.105927	0.023222	0.084164	0.093236
2	3.370900	3.130713	0.105818	0.023158	0.083523	0.093628

Subdataset 3:

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	Rougel	Rougelsum
0	3.473200	3.161193	0.094454	0.018120	0.075001	0.083629
1	3.399500	3.127282	0.095352	0.018610	0.075337	0.084209
2	3.354300	3.118323	0.095412	0.018748	0.075745	0.084342

Subdataset 4:

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	Rougel	Rougelsum
0	3.487300	3.223335	0.083462	0.013416	0.067932	0.075505
1	3.429400	3.199837	0.081381	0.013026	0.066298	0.073516
2	3.401500	3.192307	0.080658	0.012900	0.065473	0.072030

Subdataset 5:

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	Rougel	Rougelsum
0	3.404000	3.131171	0.098712	0.017809	0.078663	0.087920
1	3.338900	3.106844	0.100437	0.019132	0.079970	0.089308
2	3.309100	3.099408	0.099148	0.018768	0.079575	0.088745

Subdataset 6:

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	Rougel	Rougelsum
0	3.349000	3.096937	0.120741	0.026844	0.095862	0.106956

1	3.299900	3.083470	0.112142	0.025909	0.089888	0.099633
2	3.260500	3.077748	0.115197	0.027481	0.091893	0.102308

Subdataset 7:

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	Rougel	Rougelsum
0	3.366300	3.109919	0.110674	0.021946	0.087172	0.097535
1	3.309600	3.089842	0.108611	0.024085	0.087394	0.097108
2	3.268400	3.083262	0.109634	0.024211	0.088312	0.097732

Subdataset 8:

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	Rougel	Rougelsum
0	3.349800	3.077527	0.110075	0.020854	0.086525	0.096420
1	3.264400	3.048553	0.104256	0.018211	0.082438	0.091196
2	3.228000	3.040243	0.103955	0.018101	0.081902	0.091138

Subdataset 9:

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	Rougel	Rougelsum
0	3.344600	3.076648	0.100129	0.022501	0.080414	0.089185
1	3.288300	3.057238	0.100473	0.022367	0.081708	0.090020
2	3.261500	3.050043	0.103048	0.023704	0.083299	0.092229

Subdataset 10:

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	Rougel	Rougelsum
0	3.313600	3.065063	0.106513	0.020318	0.083613	0.093346

1	3.257800	3.047984	0.105555	0.020653	0.082840	0.092829
2	3.231800	3.043123	0.105747	0.019800	0.082561	0.092483

6.Quanto à conclusão e aos resultados

A cada etapa de treinamento(20.000 registros) foi realizado a comparação entre o modelo treinado e o modelo base, realizando uma série de perguntas solicitando a descrição dos produtos. Estes resultados estão no arquivo evaluation_history.pdf.

É notável que antes do treinamento o modelo não tinha capacidade nenhuma para descrever os produtos, na maioria das vezes retornando o nome do próprio produto ou alucinando. Após o treinamento com 200.000 registros, o modelo consegue descrever de forma simples e curta, as vezes incompleta, alguns dos produtos solicitados, para outros produtos a resposta ficou confusa e também houve alucinação. Com isso, para melhorar a performance, foi realizado o RAG sobre o modelo treinado, para buscar descrições mais precisas dos produtos solicitados. Vale ressaltar ainda que os produtos solicitados nas perguntas testes foram escolhidos aleatóriamente entre os 200.000 que foram utilizados para o treinamento e para o RAG.

Após a execução do RAG, foram realizadas novamente as mesmas perguntas e, claramente, houve um grande ganho de performance nas respostas. Apesar de fornecer respostas legais, o maior problema do modelo foi a repetição indiscriminada de texto para algumas das respostas, repetindo palavras e trechos de forma confusa.

Apesar de algumas respostas confusas, o modelo final com a utilização do RAG forneceu descrições legais para alguns produtos, não de forma perfeita, mas para um modelo base relativamente pequeno, que não conseguia descrever nenhum dos produtos solicitados, houve considerável progresso na tarefa.

Fontes utilizadas:
https://www.datacamp.com/tutorial/flan-t5-tutorial
https://huggingface.co/google/flan-t5-base
https://medium.com/georgian-impact-blog/the-practical-guide-to-llms-flan-t5-6d26cc5f14c0