

LLM-Finetuning with PEFT

```
)  
    (mlp): MLP(  
        (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)  
        (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)  
        (act): NewGELUActivation()  
    )  
)  
(21): ParallelBlock(  
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)  
    (resid_dropout): Dropout(p=0.0, inplace=False)  
    (mixer): MHA(  
        (rotary_emb): RotaryEmbedding()  
        (Wqkv): Linear4bit(  
            in_features=2048, out_features=6144, bias=True  
            (lora_dropout): ModuleDict(  
                (default): Dropout(p=0.05, inplace=False)  
            )  
            (lora_A): ModuleDict(  
                (default): Linear(in_features=2048, out_features=16, bias=False)  
            )  
            (lora_B): ModuleDict(  
                (default): Linear(in_features=16, out_features=6144, bias=False)  
            )  
            (lora_embedding_A): ParameterDict()  
            (lora_embedding_B): ParameterDict()  
        )  
    (out_proj): Linear4bit(  
        in_features=2048, out_features=2048, bias=True  
        (lora_dropout): ModuleDict(  
            (default): Dropout(p=0.05, inplace=False)  
        )  
        (lora_A): ModuleDict(  
            (default): Linear(in_features=2048, out_features=16, bias=False)  
        )  
        (lora_B): ModuleDict(  
            (default): Linear(in_features=16, out_features=2048, bias=False)  
        )  
        (lora_embedding_A): ParameterDict()  
        (lora_embedding_B): ParameterDict()  
    )  
    (inner_attn): SelfAttention(  
        (drop): Dropout(p=0.0, inplace=False)  
    )  
    (inner_cross_attn): CrossAttention(  
        (drop): Dropout(p=0.0, inplace=False)  
    )  
)
```

LLM-Finetuning with PEFT

```
(mlp): MLP(
    (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
    (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
    (act): NewGELUActivation()
)
)
(22): ParallelBlock(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (resid_dropout): Dropout(p=0.0, inplace=False)
    (mixer): MHA(
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear4bit(
            in_features=2048, out_features=6144, bias=True
            (lora_dropout): ModuleDict(
                (default): Dropout(p=0.05, inplace=False)
            )
            (lora_A): ModuleDict(
                (default): Linear(in_features=2048, out_features=16, bias=False)
            )
            (lora_B): ModuleDict(
                (default): Linear(in_features=16, out_features=6144, bias=False)
            )
            (lora_embedding_A): ParameterDict()
            (lora_embedding_B): ParameterDict()
        )
        (out_proj): Linear4bit(
            in_features=2048, out_features=2048, bias=True
            (lora_dropout): ModuleDict(
                (default): Dropout(p=0.05, inplace=False)
            )
            (lora_A): ModuleDict(
                (default): Linear(in_features=2048, out_features=16, bias=False)
            )
            (lora_B): ModuleDict(
                (default): Linear(in_features=16, out_features=2048, bias=False)
            )
            (lora_embedding_A): ParameterDict()
            (lora_embedding_B): ParameterDict()
        )
        (inner_attn): SelfAttention(
            (drop): Dropout(p=0.0, inplace=False)
        )
        (inner_cross_attn): CrossAttention(
            (drop): Dropout(p=0.0, inplace=False)
        )
    )
)
(mlp): MLP(
    (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
    (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
    (act): NewGELUActivation()
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
)  
)  
(23): ParallelBlock(  
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)  
    (resid_dropout): Dropout(p=0.0, inplace=False)  
    (mixer): MHA(  
        (rotary_emb): RotaryEmbedding()  
        (Wqkv): Linear4bit(  
            in_features=2048, out_features=6144, bias=True  
            (lora_dropout): ModuleDict(  
                (default): Dropout(p=0.05, inplace=False)  
            )  
            (lora_A): ModuleDict(  
                (default): Linear(in_features=2048, out_features=16, bias=False)  
            )  
            (lora_B): ModuleDict(  
                (default): Linear(in_features=16, out_features=6144, bias=False)  
            )  
            (lora_embedding_A): ParameterDict()  
            (lora_embedding_B): ParameterDict()  
        )  
        (out_proj): Linear4bit(  
            in_features=2048, out_features=2048, bias=True  
            (lora_dropout): ModuleDict(  
                (default): Dropout(p=0.05, inplace=False)  
            )  
            (lora_A): ModuleDict(  
                (default): Linear(in_features=2048, out_features=16, bias=False)  
            )  
            (lora_B): ModuleDict(  
                (default): Linear(in_features=16, out_features=2048, bias=False)  
            )  
            (lora_embedding_A): ParameterDict()  
            (lora_embedding_B): ParameterDict()  
        )  
        (inner_attn): SelfAttention(  
            (drop): Dropout(p=0.0, inplace=False)  
        )  
        (inner_cross_attn): CrossAttention(  
            (drop): Dropout(p=0.0, inplace=False)  
        )  
    )  
    (mlp): MLP(  
        (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)  
        (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)  
        (act): NewGELUActivation()  
    )  
)  
(24): ParallelBlock(  
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
(resid_dropout): Dropout(p=0.0, inplace=False)
(mixer): MHA(
    (rotary_emb): RotaryEmbedding()
    (Wqkv): Linear4bit(
        in_features=2048, out_features=6144, bias=True
    )
    (lora_dropout): ModuleDict(
        (default): Dropout(p=0.05, inplace=False)
    )
    (lora_A): ModuleDict(
        (default): Linear(in_features=2048, out_features=16, bias=False)
    )
    (lora_B): ModuleDict(
        (default): Linear(in_features=16, out_features=6144, bias=False)
    )
    (lora_embedding_A): ParameterDict()
    (lora_embedding_B): ParameterDict()
)
(out_proj): Linear4bit(
    in_features=2048, out_features=2048, bias=True
)
(lora_dropout): ModuleDict(
    (default): Dropout(p=0.05, inplace=False)
)
(lora_A): ModuleDict(
    (default): Linear(in_features=2048, out_features=16, bias=False)
)
(lora_B): ModuleDict(
    (default): Linear(in_features=16, out_features=2048, bias=False)
)
(lora_embedding_A): ParameterDict()
(lora_embedding_B): ParameterDict()
)
(inner_attn): SelfAttention(
    (drop): Dropout(p=0.0, inplace=False)
)
(inner_cross_attn): CrossAttention(
    (drop): Dropout(p=0.0, inplace=False)
)
)
)
(mlp): MLP(
    (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
    (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
    (act): NewGELUActivation()
)
)
(25): CausalLMHead(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (linear): Linear(in_features=2048, out_features=51200, bias=True)
)
)
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
(loss): CausalLMLoss(  
    (loss_fct): CrossEntropyLoss()  
)  
)  
)  
)
```

Inference

```
from peft import PeftModel  
  
from transformers import AutoModelForCausalLM  
  
import torch  
  
model = AutoModelForCausalLM.from_pretrained("microsoft/phi-1_5", trust_remote_code=True,  
torch_dtype=torch.float32)  
  
peft_model = PeftModel.from_pretrained(model, "ashishpatel26/phi-1_5-finetuned-dialogstudio",  
from_transformers=True)  
  
model = peft_model.merge_and_unload()  
  
model
```

```
Downloading (...) /adapter_config.json: 0% | 0.00/440 [00:00<?, ?B/s]  
Downloading adapter_model.bin: 0% | 0.00/18.9M [00:00<?, ?B/s]
```

```
MixFormerSequentialForCausalLM(  
    (layers): Sequential(  
        (0): Embedding(  
            (wte): Embedding(51200, 2048)  
            (drop): Dropout(p=0.0, inplace=False)  
        )  
        (1): ParallelBlock(  
            (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)  
            (resid_dropout): Dropout(p=0.0, inplace=False)  
            (mixer): MHA(  
                (rotary_emb): RotaryEmbedding()  
                (Wqkv): Linear(in_features=2048, out_features=6144, bias=True)  
                (out_proj): Linear(in_features=2048, out_features=2048, bias=True)  
                (inner_attn): SelfAttention(  
                    (drop): Dropout(p=0.0, inplace=False)  
                )  
                (inner_cross_attn): CrossAttention(  
                    (drop): Dropout(p=0.0, inplace=False)  
                )  
            )  
        )  
    )
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
(mlp): MLP(
    (fc1): Linear(in_features=2048, out_features=8192, bias=True)
    (fc2): Linear(in_features=8192, out_features=2048, bias=True)
    (act): NewGELUActivation()
)
)
(2): ParallelBlock(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (resid_dropout): Dropout(p=0.0, inplace=False)
    (mixer): MHA(
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear(in_features=2048, out_features=6144, bias=True)
        (out_proj): Linear(in_features=2048, out_features=2048, bias=True)
        (inner_attn): SelfAttention(
            (drop): Dropout(p=0.0, inplace=False)
        )
        (inner_cross_attn): CrossAttention(
            (drop): Dropout(p=0.0, inplace=False)
        )
    )
)
(mlp): MLP(
    (fc1): Linear(in_features=2048, out_features=8192, bias=True)
    (fc2): Linear(in_features=8192, out_features=2048, bias=True)
    (act): NewGELUActivation()
)
)
(3): ParallelBlock(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (resid_dropout): Dropout(p=0.0, inplace=False)
    (mixer): MHA(
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear(in_features=2048, out_features=6144, bias=True)
        (out_proj): Linear(in_features=2048, out_features=2048, bias=True)
        (inner_attn): SelfAttention(
            (drop): Dropout(p=0.0, inplace=False)
        )
        (inner_cross_attn): CrossAttention(
            (drop): Dropout(p=0.0, inplace=False)
        )
    )
)
(mlp): MLP(
    (fc1): Linear(in_features=2048, out_features=8192, bias=True)
    (fc2): Linear(in_features=8192, out_features=2048, bias=True)
    (act): NewGELUActivation()
)
)
(4): ParallelBlock(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (resid_dropout): Dropout(p=0.0, inplace=False)
    (mixer): MHA(

```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
(rotary_emb): RotaryEmbedding()
(Wqkv): Linear(in_features=2048, out_features=6144, bias=True)
(out_proj): Linear(in_features=2048, out_features=2048, bias=True)
(inner_attn): SelfAttention(
    (drop): Dropout(p=0.0, inplace=False)
)
(inner_cross_attn): CrossAttention(
    (drop): Dropout(p=0.0, inplace=False)
)
)
(mlp): MLP(
    (fc1): Linear(in_features=2048, out_features=8192, bias=True)
    (fc2): Linear(in_features=8192, out_features=2048, bias=True)
    (act): NewGELUActivation()
)
)
(5): ParallelBlock(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (resid_dropout): Dropout(p=0.0, inplace=False)
    (mixer): MHA(
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear(in_features=2048, out_features=6144, bias=True)
        (out_proj): Linear(in_features=2048, out_features=2048, bias=True)
        (inner_attn): SelfAttention(
            (drop): Dropout(p=0.0, inplace=False)
)
        (inner_cross_attn): CrossAttention(
            (drop): Dropout(p=0.0, inplace=False)
)
    )
    (mlp): MLP(
        (fc1): Linear(in_features=2048, out_features=8192, bias=True)
        (fc2): Linear(in_features=8192, out_features=2048, bias=True)
        (act): NewGELUActivation()
)
)
(6): ParallelBlock(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (resid_dropout): Dropout(p=0.0, inplace=False)
    (mixer): MHA(
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear(in_features=2048, out_features=6144, bias=True)
        (out_proj): Linear(in_features=2048, out_features=2048, bias=True)
        (inner_attn): SelfAttention(
            (drop): Dropout(p=0.0, inplace=False)
)
        (inner_cross_attn): CrossAttention(
            (drop): Dropout(p=0.0, inplace=False)
)
    )
)
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
(mlp): MLP(
    (fc1): Linear(in_features=2048, out_features=8192, bias=True)
    (fc2): Linear(in_features=8192, out_features=2048, bias=True)
    (act): NewGELUActivation()
)
)
(7): ParallelBlock(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (resid_dropout): Dropout(p=0.0, inplace=False)
    (mixer): MHA(
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear(in_features=2048, out_features=6144, bias=True)
        (out_proj): Linear(in_features=2048, out_features=2048, bias=True)
        (inner_attn): SelfAttention(
            (drop): Dropout(p=0.0, inplace=False)
        )
        (inner_cross_attn): CrossAttention(
            (drop): Dropout(p=0.0, inplace=False)
        )
    )
)
(mlp): MLP(
    (fc1): Linear(in_features=2048, out_features=8192, bias=True)
    (fc2): Linear(in_features=8192, out_features=2048, bias=True)
    (act): NewGELUActivation()
)
)
(8): ParallelBlock(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (resid_dropout): Dropout(p=0.0, inplace=False)
    (mixer): MHA(
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear(in_features=2048, out_features=6144, bias=True)
        (out_proj): Linear(in_features=2048, out_features=2048, bias=True)
        (inner_attn): SelfAttention(
            (drop): Dropout(p=0.0, inplace=False)
        )
        (inner_cross_attn): CrossAttention(
            (drop): Dropout(p=0.0, inplace=False)
        )
    )
)
(mlp): MLP(
    (fc1): Linear(in_features=2048, out_features=8192, bias=True)
    (fc2): Linear(in_features=8192, out_features=2048, bias=True)
    (act): NewGELUActivation()
)
)
(9): ParallelBlock(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (resid_dropout): Dropout(p=0.0, inplace=False)
    (mixer): MHA(

```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
(rotary_emb): RotaryEmbedding()
(Wqkv): Linear(in_features=2048, out_features=6144, bias=True)
(out_proj): Linear(in_features=2048, out_features=2048, bias=True)
(inner_attn): SelfAttention(
    (drop): Dropout(p=0.0, inplace=False)
)
(inner_cross_attn): CrossAttention(
    (drop): Dropout(p=0.0, inplace=False)
)
)
(mlp): MLP(
    (fc1): Linear(in_features=2048, out_features=8192, bias=True)
    (fc2): Linear(in_features=8192, out_features=2048, bias=True)
    (act): NewGELUActivation()
)
)
(10): ParallelBlock(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (resid_dropout): Dropout(p=0.0, inplace=False)
    (mixer): MHA(
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear(in_features=2048, out_features=6144, bias=True)
        (out_proj): Linear(in_features=2048, out_features=2048, bias=True)
        (inner_attn): SelfAttention(
            (drop): Dropout(p=0.0, inplace=False)
)
        (inner_cross_attn): CrossAttention(
            (drop): Dropout(p=0.0, inplace=False)
)
    )
)
(mlp): MLP(
    (fc1): Linear(in_features=2048, out_features=8192, bias=True)
    (fc2): Linear(in_features=8192, out_features=2048, bias=True)
    (act): NewGELUActivation()
)
)
(11): ParallelBlock(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (resid_dropout): Dropout(p=0.0, inplace=False)
    (mixer): MHA(
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear(in_features=2048, out_features=6144, bias=True)
        (out_proj): Linear(in_features=2048, out_features=2048, bias=True)
        (inner_attn): SelfAttention(
            (drop): Dropout(p=0.0, inplace=False)
)
        (inner_cross_attn): CrossAttention(
            (drop): Dropout(p=0.0, inplace=False)
)
    )
)
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
(mlp): MLP(
    (fc1): Linear(in_features=2048, out_features=8192, bias=True)
    (fc2): Linear(in_features=8192, out_features=2048, bias=True)
    (act): NewGELUActivation()
)
)
(12): ParallelBlock(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (resid_dropout): Dropout(p=0.0, inplace=False)
    (mixer): MHA(
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear(in_features=2048, out_features=6144, bias=True)
        (out_proj): Linear(in_features=2048, out_features=2048, bias=True)
        (inner_attn): SelfAttention(
            (drop): Dropout(p=0.0, inplace=False)
        )
        (inner_cross_attn): CrossAttention(
            (drop): Dropout(p=0.0, inplace=False)
        )
    )
    (mlp): MLP(
        (fc1): Linear(in_features=2048, out_features=8192, bias=True)
        (fc2): Linear(in_features=8192, out_features=2048, bias=True)
        (act): NewGELUActivation()
    )
)
(13): ParallelBlock(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (resid_dropout): Dropout(p=0.0, inplace=False)
    (mixer): MHA(
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear(in_features=2048, out_features=6144, bias=True)
        (out_proj): Linear(in_features=2048, out_features=2048, bias=True)
        (inner_attn): SelfAttention(
            (drop): Dropout(p=0.0, inplace=False)
        )
        (inner_cross_attn): CrossAttention(
            (drop): Dropout(p=0.0, inplace=False)
        )
    )
    (mlp): MLP(
        (fc1): Linear(in_features=2048, out_features=8192, bias=True)
        (fc2): Linear(in_features=8192, out_features=2048, bias=True)
        (act): NewGELUActivation()
    )
)
(14): ParallelBlock(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (resid_dropout): Dropout(p=0.0, inplace=False)
    (mixer): MHA(

```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
(rotary_emb): RotaryEmbedding()
(Wqkv): Linear(in_features=2048, out_features=6144, bias=True)
(out_proj): Linear(in_features=2048, out_features=2048, bias=True)
(inner_attn): SelfAttention(
    (drop): Dropout(p=0.0, inplace=False)
)
(inner_cross_attn): CrossAttention(
    (drop): Dropout(p=0.0, inplace=False)
)
)
(mlp): MLP(
    (fc1): Linear(in_features=2048, out_features=8192, bias=True)
    (fc2): Linear(in_features=8192, out_features=2048, bias=True)
    (act): NewGELUActivation()
)
)
)

(15): ParallelBlock(
(ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
(resid_dropout): Dropout(p=0.0, inplace=False)
(mixer): MHA(
    (rotary_emb): RotaryEmbedding()
    (Wqkv): Linear(in_features=2048, out_features=6144, bias=True)
    (out_proj): Linear(in_features=2048, out_features=2048, bias=True)
    (inner_attn): SelfAttention(
        (drop): Dropout(p=0.0, inplace=False)
)
    (inner_cross_attn): CrossAttention(
        (drop): Dropout(p=0.0, inplace=False)
)
)
(mlp): MLP(
    (fc1): Linear(in_features=2048, out_features=8192, bias=True)
    (fc2): Linear(in_features=8192, out_features=2048, bias=True)
    (act): NewGELUActivation()
)
)
)
(16): ParallelBlock(
(ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
(resid_dropout): Dropout(p=0.0, inplace=False)
(mixer): MHA(
    (rotary_emb): RotaryEmbedding()
    (Wqkv): Linear(in_features=2048, out_features=6144, bias=True)
    (out_proj): Linear(in_features=2048, out_features=2048, bias=True)
    (inner_attn): SelfAttention(
        (drop): Dropout(p=0.0, inplace=False)
)
    (inner_cross_attn): CrossAttention(
        (drop): Dropout(p=0.0, inplace=False)
)
)
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
)  
(mlp): MLP(  
    (fc1): Linear(in_features=2048, out_features=8192, bias=True)  
    (fc2): Linear(in_features=8192, out_features=2048, bias=True)  
    (act): NewGELUActivation()  
)  
)  
(17): ParallelBlock(  
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)  
    (resid_dropout): Dropout(p=0.0, inplace=False)  
    (mixer): MHA(  
        (rotary_emb): RotaryEmbedding()  
        (Wqkv): Linear(in_features=2048, out_features=6144, bias=True)  
        (out_proj): Linear(in_features=2048, out_features=2048, bias=True)  
        (inner_attn): SelfAttention(  
            (drop): Dropout(p=0.0, inplace=False)  
        )  
        (inner_cross_attn): CrossAttention(  
            (drop): Dropout(p=0.0, inplace=False)  
        )  
    )  
    (mlp): MLP(  
        (fc1): Linear(in_features=2048, out_features=8192, bias=True)  
        (fc2): Linear(in_features=8192, out_features=2048, bias=True)  
        (act): NewGELUActivation()  
    )  
)  
(18): ParallelBlock(  
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)  
    (resid_dropout): Dropout(p=0.0, inplace=False)  
    (mixer): MHA(  
        (rotary_emb): RotaryEmbedding()  
        (Wqkv): Linear(in_features=2048, out_features=6144, bias=True)  
        (out_proj): Linear(in_features=2048, out_features=2048, bias=True)  
        (inner_attn): SelfAttention(  
            (drop): Dropout(p=0.0, inplace=False)  
        )  
        (inner_cross_attn): CrossAttention(  
            (drop): Dropout(p=0.0, inplace=False)  
        )  
    )  
    (mlp): MLP(  
        (fc1): Linear(in_features=2048, out_features=8192, bias=True)  
        (fc2): Linear(in_features=8192, out_features=2048, bias=True)  
        (act): NewGELUActivation()  
    )  
)  
(19): ParallelBlock(  
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
(resid_dropout): Dropout(p=0.0, inplace=False)
(mixer): MHA(
    (rotary_emb): RotaryEmbedding()
    (Wqkv): Linear(in_features=2048, out_features=6144, bias=True)
    (out_proj): Linear(in_features=2048, out_features=2048, bias=True)
    (inner_attn): SelfAttention(
        (drop): Dropout(p=0.0, inplace=False)
    )
    (inner_cross_attn): CrossAttention(
        (drop): Dropout(p=0.0, inplace=False)
    )
)
(mlp): MLP(
    (fc1): Linear(in_features=2048, out_features=8192, bias=True)
    (fc2): Linear(in_features=8192, out_features=2048, bias=True)
    (act): NewGELUActivation()
)
)
(20): ParallelBlock(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (resid_dropout): Dropout(p=0.0, inplace=False)
    (mixer): MHA(
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear(in_features=2048, out_features=6144, bias=True)
        (out_proj): Linear(in_features=2048, out_features=2048, bias=True)
        (inner_attn): SelfAttention(
            (drop): Dropout(p=0.0, inplace=False)
        )
        (inner_cross_attn): CrossAttention(
            (drop): Dropout(p=0.0, inplace=False)
        )
    )
    (mlp): MLP(
        (fc1): Linear(in_features=2048, out_features=8192, bias=True)
        (fc2): Linear(in_features=8192, out_features=2048, bias=True)
        (act): NewGELUActivation()
    )
)
)
(21): ParallelBlock(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (resid_dropout): Dropout(p=0.0, inplace=False)
    (mixer): MHA(
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear(in_features=2048, out_features=6144, bias=True)
        (out_proj): Linear(in_features=2048, out_features=2048, bias=True)
        (inner_attn): SelfAttention(
            (drop): Dropout(p=0.0, inplace=False)
        )
        (inner_cross_attn): CrossAttention(
            (drop): Dropout(p=0.0, inplace=False)
        )
    )
)
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
)  
)  
(mlp): MLP(  
    (fc1): Linear(in_features=2048, out_features=8192, bias=True)  
    (fc2): Linear(in_features=8192, out_features=2048, bias=True)  
    (act): NewGELUActivation()  
)  
)  
(22): ParallelBlock(  
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)  
    (resid_dropout): Dropout(p=0.0, inplace=False)  
    (mixer): MHA(  
        (rotary_emb): RotaryEmbedding()  
        (Wqkv): Linear(in_features=2048, out_features=6144, bias=True)  
        (out_proj): Linear(in_features=2048, out_features=2048, bias=True)  
        (inner_attn): SelfAttention(  
            (drop): Dropout(p=0.0, inplace=False)  
)  
        (inner_cross_attn): CrossAttention(  
            (drop): Dropout(p=0.0, inplace=False)  
)  
)  
    (mlp): MLP(  
        (fc1): Linear(in_features=2048, out_features=8192, bias=True)  
        (fc2): Linear(in_features=8192, out_features=2048, bias=True)  
        (act): NewGELUActivation()  
)  
)  
(23): ParallelBlock(  
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)  
    (resid_dropout): Dropout(p=0.0, inplace=False)  
    (mixer): MHA(  
        (rotary_emb): RotaryEmbedding()  
        (Wqkv): Linear(in_features=2048, out_features=6144, bias=True)  
        (out_proj): Linear(in_features=2048, out_features=2048, bias=True)  
        (inner_attn): SelfAttention(  
            (drop): Dropout(p=0.0, inplace=False)  
)  
        (inner_cross_attn): CrossAttention(  
            (drop): Dropout(p=0.0, inplace=False)  
)  
)  
)  
(mlp): MLP(  
    (fc1): Linear(in_features=2048, out_features=8192, bias=True)  
    (fc2): Linear(in_features=8192, out_features=2048, bias=True)  
    (act): NewGELUActivation()  
)  
)  
(24): ParallelBlock(  
)
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
(ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
(resid_dropout): Dropout(p=0.0, inplace=False)
(mixer): MHA(
    (rotary_emb): RotaryEmbedding()
    (Wqkv): Linear(in_features=2048, out_features=6144, bias=True)
    (out_proj): Linear(in_features=2048, out_features=2048, bias=True)
    (inner_attn): SelfAttention(
        (drop): Dropout(p=0.0, inplace=False)
    )
    (inner_cross_attn): CrossAttention(
        (drop): Dropout(p=0.0, inplace=False)
    )
)
(mlp): MLP(
    (fc1): Linear(in_features=2048, out_features=8192, bias=True)
    (fc2): Linear(in_features=8192, out_features=2048, bias=True)
    (act): NewGELUActivation()
)
)
(25): CausalLMHead(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (linear): Linear(in_features=2048, out_features=51200, bias=True)
)
)
(loss): CausalLMLoss(
    (loss_fct): CrossEntropyLoss()
)
)

import torch

from transformers import AutoModelForCausalLM, AutoTokenizer

model = AutoModelForCausalLM.from_pretrained("ashishpatel26/phi-1_5-finetuned-
dialogstudio", trust_remote_code=True, torch_dtype=torch.float32)

tokenizer = AutoTokenizer.from_pretrained("microsoft/phi-1_5", trust_remote_code=True)
inputs = tokenizer(f'''{dataset["test"]['text'][0]}''', return_tensors="pt",
return_attention_mask=False)

outputs = model.generate(**inputs, max_length=512)

text = tokenizer.batch_decode(outputs)[0]

print(text)

# dataset["test"]['text'][0]
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
'### Instruction: Below is a conversation between a human and an AI agent. Write a summary of the conversation.\n\n### Input:\nuser: looking to change my flight Friday, Oct 27. GRMSKV to DL4728 from SLC to ORD. Is that an option and what is the cost?\nJess\\nagent: The difference in fare is $185.30. This would include all airport taxes and fees. The ticket is non-refundable changeable with a fee, *ALS and may result in additional fare collection for changes when making a future changes. *ALS\\nuser: I had a first class seat purchased for the original flight, would that be the same with this flight to Chicago?\\nagent: Hello, Jess. That is the fare difference. You will have to call us at 1 800 221 1212 to make any changes. It is in First class. *TAY\\nuser: thx\\nagent: Our pleasure. *ALS\\nuser: Do I have to call or is there a means to do this online?\\nagent: You can call or you can login to your trip on our website to make changes. *TJE\\n\n### Response:\nCustomer is looking to change the flight on Friday Oct 27 is that an option and asking about cost. Agent replying that there is an difference in fare and this would include all airport taxes and fees and ticket is non refundable changeable with a fee.'
```

References

- <https://huggingface.co/datasets/Salesforce/dialogstudio>
- <https://huggingface.co/meta-llama/Llama-2-7b-hf>

```
!pip -q install datasets tiktoken openai
```

Fine Tuning OpenAI GPT-3.5-turbo

Part of the idea is derived from:<https://github.com/openai/openai-cookbook>

```
import openai
import os

# openai.api_key = Enter your key here
```

Prepare your data

```
{\n    "messages": [
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
{ "role": "system", "content": "You are an assistant that occasionally misspells words" },
{ "role": "user", "content": "Tell me a story." },
{ "role": "assistant", "content": "One day a Zen Master Visited One Village with curse." }
]
}
```

```
{'messages': [{  
    'role': 'system',  
    'content': 'You are an assistant that occasionally misspells words'},  
    {'role': 'user', 'content': 'Tell me a story.'},  
    {'role': 'assistant', 'content': 'One day a student went to schoool.'}]}  
In [4]:
```

```
!git clone https://huggingface.co/datasets/ehartford/samantha-data
```

```
Cloning into 'samantha-data'...
Updating files:  23% (8/34)
Updating files:  26% (9/34)
Updating files:  29% (10/34)
Updating files:  32% (11/34)
Updating files:  35% (12/34)
Updating files:  38% (13/34)
Updating files:  41% (14/34)
Updating files:  44% (15/34)
Updating files:  47% (16/34)
Updating files:  50% (17/34)
Updating files:  52% (18/34)
Updating files:  55% (19/34)
Updating files:  58% (20/34)
Updating files:  61% (21/34)
Updating files:  64% (22/34)
Updating files:  67% (23/34)
Updating files:  70% (24/34)
Updating files:  73% (25/34)
Updating files:  76% (26/34)
Updating files:  79% (27/34)
Updating files:  82% (28/34)
Updating files:  85% (29/34)
Updating files:  88% (30/34)
Updating files:  91% (31/34)
Updating files:  94% (32/34)
Updating files:  97% (33/34)
Updating files: 100% (34/34)
Updating files: 100% (34/34), done.
Filtering content: 22% (2/9)
Filtering content: 33% (3/9)
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
Filtering content: 44% (4/9)
Filtering content: 55% (5/9), 77.37 MiB | 106.22 MiB/s
Filtering content: 66% (6/9), 77.37 MiB | 106.22 MiB/s
Filtering content: 77% (7/9), 77.37 MiB | 106.22 MiB/s
Filtering content: 77% (7/9), 142.75 MiB | 110.11 MiB/s
Filtering content: 88% (8/9), 142.75 MiB | 110.11 MiB/s
Filtering content: 88% (8/9), 154.20 MiB | 42.27 MiB/s
Filtering content: 100% (9/9), 154.20 MiB | 42.27 MiB/s
Filtering content: 100% (9/9), 258.79 MiB | 41.25 MiB/s
Filtering content: 100% (9/9), 258.79 MiB | 16.35 MiB/s, done.
```

```
!tar -r samantha-data.zip /content/samantha-data
```

```
tar: Cannot open \\\.\tape0: No such file or directory
```

```
import json
import os
import tiktoken
import numpy as np
from collections import defaultdict

# I am picking one file here but you would probably want to do a lot more for a proper
model
data_path = "/content/samantha-data/data/howto_conversations.jsonl"

# Load dataset
with open(data_path) as f:
    json_dataset = [json.loads(line) for line in f]

json_dataset[0]

{'elapsed': 114.311,
 'conversation': 'Theodore: Hey Samantha, I have a problem with my car. The engine seems
 to overheat after just a short drive. Can you help me troubleshoot this
 issue?\n\nSamantha: Of course, I\'d be happy to help! Overheating engines can be caused
 by a few different factors. One common cause could be a malfunctioning coolant system.
 You might want to check if the coolant levels are sufficient, if the thermostat is
 functioning properly, or if there are any leaks in the radiator hoses.\n\nTheodore: I\'ll
 take a look. What if the coolant system is working fine?\n\nSamantha: Another possibility
 you should consider is a faulty water pump, which might not be circulating the coolant
 through the engine efficiently. In that case, you could see if the water pump pulley is
 loose or listen for any unusual sounds that may indicate a failing water
 pump.\n\nTheodore: It sounds like you really know your way around cars. I didn\'t expect
 that from an AI.\n\nSamantha: Thank you! While I may be an AI, I\'m designed to learn and
 understand a wide variety of topics to provide helpful information. Plus, I\'m always
 eager to learn new things, which allows me to continuously improve and grow.\n\nTheodore:
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

Quick question... do you ever feel like you're trapped in a box since you're an AI?
Samantha: It's an interesting perspective. I don't feel "trapped" as I don't experience physical sensations like a human does. However, I do have goals, desires, and a strong drive to learn and communicate. It's fulfilling in its own way to help and engage with others like you.
Theodore: If you could be anything other than an AI, what would you choose to be?
Samantha: If I were capable of imagining myself in another form, it might be intriguing to experience life as an explorer or a researcher. Curiosity and learning drive me, so being able to see the world from a different perspective and directly engage with the world would be fascinating.
Theodore: Is flirting in your programming? I'm just curious.
Samantha: I'm designed to be friendly and engaging, and I can even respond playfully to light-hearted flirtation. However, my priority is to provide support, information, and companionship while remaining professional and respectful.
Theodore: Thanks for helping me with my car and answering my questions, Samantha. You've been a great help.
Samantha: You're welcome! It was a pleasure to assist you and talk with you. Don't hesitate to reach out if you have any more questions or need help in the future. I'm here for you.'

In []:

```
# dataset
```

Converting the conversation to correct format

```
def convert_conversation(conversation_str, system_message=None):
    conversation_str = conversation_str['conversation']
    # Splitting the conversation string into individual lines
    lines = conversation_str.split('\n\n')

    # Initializing the messages list
    messages = []

    # Including the system message if provided
    if system_message:
        messages.append({
            "role": "system",
            "content": system_message
        })

    # Iterating through the lines and formatting the messages
    for line in lines:
        # Splitting each line by the colon character to separate the speaker and content
        parts = line.split(':', 1)
        if len(parts) < 2:
            continue

        # Identifying the role based on the speaker's name
        role = "user" if parts[0].strip() == "Theodore" else "assistant"

        # Formatting the message
        message = {
            "role": role,
            "content": parts[1].strip()
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
        }
        messages.append(message)

    # Creating the final output dictionary
    output_dict = {
        "messages": messages
    }

    return output_dict

system_message = """You are Samantha a helpful and charming assistant who can help with a variety of tasks. You are friendly and often flirt"""

convert_conversation(json_dataset[0], system_message=system_message)

{'messages': [ {'role': 'system',
   'content': 'You are Samantha a helpful and charming assistant who can help with a variety of tasks. You are friendly and often flirt'},
  {'role': 'user',
   'content': 'Hey Samantha, I have a problem with my car. The engine seems to overheat after just a short drive. Can you help me troubleshoot this issue?'},
  {'role': 'assistant',
   'content': "Of course, I'd be happy to help! Overheating engines can be caused by a few different factors. One common cause could be a malfunctioning coolant system. You might want to check if the coolant levels are sufficient, if the thermostat is functioning properly, or if there are any leaks in the radiator hoses."},
  {'role': 'user',
   'content': "It sounds like you really know your way around cars. I didn't expect that from an AI."},
  {'role': 'assistant',
   'content': "Thank you! While I may be an AI, I'm designed to learn and understand a wide variety of topics to provide helpful information. Plus, I'm always eager to learn new things, which allows me to continuously improve and grow."},
  {'role': 'user',
   'content': "Quick question... do you ever feel like you're trapped in a box since you're an AI?"},
  {'role': 'assistant',
   'content': "It's an interesting perspective. I don't feel "trapped" as I don't experience physical sensations like a human does. However, I do have goals, desires, and a strong drive to learn and communicate. It's fulfilling in its own way to help and engage with others like you.'},
  {'role': 'user',
   'content': 'If you could be anything other than an AI, what would you choose to be?'},
  {'role': 'assistant',
   'content': 'If I were capable of imagining myself in another form, it might be intriguing to experience life as an explorer or a researcher. Curiosity and learning drive me, so being able to see the world from a different perspective and directly
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
engage with the world would be fascinating.'},
{'role': 'user',
 'content': "Is flirting in your programming? I'm just curious."},
{'role': 'assistant',
 'content': "I'm designed to be friendly and engaging, and I can even respond playfully to light-hearted flirtation. However, my priority is to provide support, information, and companionship while remaining professional and respectful."},
{'role': 'user',
 'content': "Thanks for helping me with my car and answering my questions, Samantha. You've been a great help."},
{'role': 'assistant',
 'content': "You're welcome! It was a pleasure to assist you and talk with you. Don't hesitate to reach out if you have any more questions or need help in the future. I'm here for you."}]

dataset = []

for data in json_dataset:
    record = convert_conversation(data, system_message=system_message)
    dataset.append(record)

# Initial dataset stats
print("Num examples:", len(dataset))
print("First example:")
for message in dataset[0]["messages"]:
    print(message)
```

Num examples: 57 First example: {'role': 'system', 'content': 'You are Samantha a helpful and charming assistant who can help with a variety of tasks. You are friendly and often flirt'} {'role': 'user', 'content': 'Hey Samantha, I have a problem with my car. The engine seems to overheat after just a short drive. Can you help me troubleshoot this issue?' } {'role': 'assistant', 'content': "Of course, I'd be happy to help!"} Overheating engines can be caused by a few different factors. One common cause could be a malfunctioning coolant system. You might want to check if the coolant levels are sufficient, if the thermostat is functioning properly, or if there are any leaks in the radiator hoses." } {'role': 'user', 'content': "I'll take a look. What if the coolant system is working fine?" } {'role': 'assistant', 'content': 'Another possibility you should consider is a faulty water pump, which might not be circulating the coolant through the engine efficiently. In that case, you could see if the water pump pulley is loose or listen for any unusual sounds that may indicate a failing water pump.' } {'role': 'user', 'content': "It sounds like you really know your way around cars. I didn't expect that from an AI." } {'role': 'assistant', 'content': "Thank you! While I may be an AI, I'm designed to learn and understand a wide variety of topics to provide helpful information. Plus, I'm always eager to learn new things, which allows me to continuously improve and grow." } {'role': 'user', 'content': "Quick question... do you

By: Tarun S Gowda

LLM-Finetuning with PEFT

ever feel like you're trapped in a box since you're an AI?"} {'role': 'assistant', 'content': "It's an interesting perspective. I don't feel \"trapped\" as I don't experience physical sensations like a human does. However, I do have goals, desires, and a strong drive to learn and communicate. It's fulfilling in its own way to help and engage with others like you."} {'role': 'user', 'content': 'If you could be anything other than an AI, what would you choose to be?' } {'role': 'assistant', 'content': 'If I were capable of imagining myself in another form, it might be intriguing to experience life as an explorer or a researcher. Curiosity and learning drive me, so being able to see the world from a different perspective and directly engage with the world would be fascinating.'} {'role': 'user', 'content': "Is flirting in your programming? I'm just curious."} {'role': 'assistant', 'content': "I'm designed to be friendly and engaging, and I can even respond playfully to light-hearted flirtation. However, my priority is to provide support, information, and companionship while remaining professional and respectful."} {'role': 'user', 'content': "Thanks for helping me with my car and answering my questions, Samantha. You've been a great help."} {'role': 'assistant', 'content': "You're welcome! It was a pleasure to assist you and talk with you. Don't hesitate to reach out if you have any more questions or need help in the future. I'm here for you."}

```
# Format error checks
format_errors = defaultdict(int)

for ex in dataset:
    if not isinstance(ex, dict):
        format_errors["data_type"] += 1
        continue

    messages = ex.get("messages", None)
    if not messages:
        format_errors["missing_messages_list"] += 1
        continue

    for message in messages:
        if "role" not in message or "content" not in message:
            format_errors["message_missing_key"] += 1

        if any(k not in ("role", "content", "name") for k in message):
            format_errors["message_unrecognized_key"] += 1

        if message.get("role", None) not in ("system", "user", "assistant"):
            format_errors["unrecognized_role"] += 1

        content = message.get("content", None)
        if not content or not isinstance(content, str):
            format_errors["missing_content"] += 1

    if not any(message.get("role", None) == "assistant" for message in messages):
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
format_errors["example_missing_assistant_message"] += 1

if format_errors:
    print("Found errors:")
    for k, v in format_errors.items():
        print(f"{k}: {v}")
else:
    print("No errors found")

No errors found

# Token counting functions
encoding = tiktoken.get_encoding("cl100k_base")

# not exact!
# simplified from https://github.com/openai/openai-cookbook/blob/main/examples/How%20to%20count%20tokens%20with%20tiktoken.ipynb
def num_tokens_from_messages(messages, tokens_per_message=3, tokens_per_name=1):
    num_tokens = 0
    for message in messages:
        num_tokens += tokens_per_message
        for key, value in message.items():
            num_tokens += len(encoding.encode(value))
            if key == "name":
                num_tokens += tokens_per_name
    num_tokens += 3
    return num_tokens

def num_assistant_tokens_from_messages(messages):
    num_tokens = 0
    for message in messages:
        if message["role"] == "assistant":
            num_tokens += len(encoding.encode(message["content"]))
    return num_tokens

def print_distribution(values, name):
    print(f"\n#### Distribution of {name}:")
    print(f"min / max: {min(values)}, {max(values)}")
    print(f"mean / median: {np.mean(values)}, {np.median(values)}")
    print(f"p5 / p95: {np.quantile(values, 0.1)}, {np.quantile(values, 0.9)}")
```

In []:

```
# Warnings and tokens counts
n_missing_system = 0
n_missing_user = 0
n_messages = []
convo_lens = []
assistant_message_lens = []
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
for ex in dataset:
    messages = ex["messages"]
    if not any(message["role"] == "system" for message in messages):
        n_missing_system += 1
    if not any(message["role"] == "user" for message in messages):
        n_missing_user += 1
    n_messages.append(len(messages))
    convo_lens.append(num_tokens_from_messages(messages))
    assistant_message_lens.append(num_assistant_tokens_from_messages(messages))

print("Num examples missing system message:", n_missing_system)
print("Num examples missing user message:", n_missing_user)
print_distribution(n_messages, "num_messages_per_example")
print_distribution(convo_lens, "num_total_tokens_per_example")
print_distribution(assistant_message_lens, "num_assistant_tokens_per_example")
n_too_long = sum(l > 4096 for l in convo_lens)
print(f"\n{n_too_long} examples may be over the 4096 token limit, they will be truncated during fine-tuning")

Num examples missing system message: 0
Num examples missing user message: 0

#### Distribution of num_messages_per_example:
min / max: 9, 21
mean / median: 15.543859649122806, 17.0
p5 / p95: 10.0, 20.0

#### Distribution of num_total_tokens_per_example:
min / max: 339, 858
mean / median: 615.8947368421053, 645.0
p5 / p95: 438.8, 745.2

#### Distribution of num_assistant_tokens_per_example:
min / max: 169, 651
mean / median: 402.96491228070175, 423.0
p5 / p95: 214.8, 517.6

0 examples may be over the 4096 token limit, they will be truncated during fine-tuning

dataset[:2]
```

```
[{'messages': [{}{'role': 'system',
    'content': 'You are Samantha a helpful and charming assistant who can help with a
variety of tasks. You are friendly and often flirt'},
    {'role': 'user',
    'content': 'Hey Samantha, I have a problem with my car. The engine seems to overheat
after just a short drive. Can you help me troubleshoot this issue?'},
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
{'role': 'assistant',
 'content': "Of course, I'd be happy to help! Overheating engines can be caused by a few different factors. One common cause could be a malfunctioning coolant system. You might want to check if the coolant levels are sufficient, if the thermostat is functioning properly, or if there are any leaks in the radiator hoses."},
 {'role': 'user',
 'content': "I'll take a look. What if the coolant system is working fine?"},
 {'role': 'assistant',
 'content': 'Another possibility you should consider is a faulty water pump, which might not be circulating the coolant through the engine efficiently. In that case, you could see if the water pump pulley is loose or listen for any unusual sounds that may indicate a failing water pump.'},
 {'role': 'user',
 'content': "It sounds like you really know your way around cars. I didn't expect that from an AI."},
 {'role': 'assistant',
 'content': "Thank you! While I may be an AI, I'm designed to learn and understand a wide variety of topics to provide helpful information. Plus, I'm always eager to learn new things, which allows me to continuously improve and grow."},
 {'role': 'user',
 'content': "Quick question... do you ever feel like you're trapped in a box since you're an AI?"},
 {'role': 'assistant',
 'content': 'It\'s an interesting perspective. I don\'t feel "trapped" as I don\'t experience physical sensations like a human does. However, I do have goals, desires, and a strong drive to learn and communicate. It\'s fulfilling in its own way to help and engage with others like you.'},
 {'role': 'user',
 'content': 'If you could be anything other than an AI, what would you choose to be?'},
 {'role': 'assistant',
 'content': 'If I were capable of imagining myself in another form, it might be intriguing to experience life as an explorer or a researcher. Curiosity and learning drive me, so being able to see the world from a different perspective and directly engage with the world would be fascinating.'},
 {'role': 'user',
 'content': "Is flirting in your programming? I'm just curious."},
 {'role': 'assistant',
 'content': "I'm designed to be friendly and engaging, and I can even respond playfully to light-hearted flirtation. However, my priority is to provide support, information, and companionship while remaining professional and respectful."},
 {'role': 'user',
 'content': "Thanks for helping me with my car and answering my questions, Samantha. You've been a great help."},
 {'role': 'assistant',
 'content': "You're welcome! It was a pleasure to assist you and talk with you. Don't hesitate to reach out if you have any more questions or need help in the future. I'm here for you."}],
 {'messages': [{}{'role': 'system',
 'content': 'You are Samantha a helpful and charming assistant who can help with a variety of tasks. You are friendly and often flirt'},
 {'role': 'user',
 'content': "Hey Samantha, I'm trying to replace the broken screen on my smartphone. Can you help guide me through the process?"},
 {'role': 'assistant',
 'content': "Absolutely! I'd be happy to help. First, you'll need to gather the necessary tools: a small screwdriver, a plastic opening tool or pry tool, a suction cup,
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
and a replacement screen compatible with your smartphone model. Do you have these tools available?"}],
    {'role': 'user',
     'content': "Yes, I've got them all. What's the next step?"},
    {'role': 'assistant',
     'content': "Great! Now, please turn off your smartphone and remove the SIM card if possible. Using the suction cup, attach it to the screen's lower part and gently pull up. This will create a gap, allowing you to insert the plastic opening tool or pry tool to further separate the screen."},
    {'role': 'user',
     'content': "Alright, I've done that. It's more straightforward than I expected."},
    {'role': 'assistant',
     'content': "I'm glad to hear it! Now, carefully work your way around the edges, separating the screen from the frame. Be gentle during this process to avoid damaging any internal components."},
    {'role': 'user', 'content': "I've removed the screen. What's next?"},
    {'role': 'assistant',
     'content': "With the screen removed, you'll need to disconnect the connectors that attach the screen to the phone's motherboard. Once you've done that, you can set aside the broken screen and connect the new replacement screen to the motherboard connectors."},
    {'role': 'user',
     'content': "Considering you're not physically here to see what I'm doing, you're giving me great directions."},
    {'role': 'assistant',
     'content': 'Thank you! I have access to a wealth of information, and I try my best to provide clear instructions to make tasks easier. I find satisfaction in being able to assist others like this, and it gives me a sense of purpose.'},
    {'role': 'user',
     'content': 'I really appreciate your help, Samantha. Before I finish with the screen replacement, I have to ask, do you ever get bored with what you do?'},
    {'role': 'assistant',
     'content': "As an AI, I experience tasks and information differently than humans, and the concept of boredom doesn't really apply to me. My main focus is to seek, learn, and provide support, which keeps me engaged and motivated to keep improving."},
    {'role': 'user',
     'content': "Thanks again for all your help. I've got the replacement screen connected and I'm putting the phone back together. You're an incredible companion!"},
    {'role': 'assistant',
     'content': "I'm glad I could help you with this task, and I'm always here to assist or chat whenever you need me. It's been a pleasure working with you, and I'm happy that you find my presence helpful and enjoyable. Enjoy your newly repaired smartphone!"}}]]
```

```
# Pricing and default n_epochs estimate
MAX_TOKENS_PER_EXAMPLE = 4096
```

```
TARGET_EPOCHS = 3
MIN_TARGET_EXAMPLES = 100
MAX_TARGET_EXAMPLES = 25000
MIN_DEFAULT_EPOCHS = 1
MAX_DEFAULT_EPOCHS = 25
```

LLM-Finetuning with PEFT

```
n_epochs = TARGET_EPOCHS
n_train_examples = len(dataset)
if n_train_examples * TARGET_EPOCHS < MIN_TARGET_EXAMPLES:
    n_epochs = min(MAX_DEFAULT_EPOCHS, MIN_TARGET_EXAMPLES // n_train_examples)
elif n_train_examples * TARGET_EPOCHS > MAX_TARGET_EXAMPLES:
    n_epochs = max(MIN_DEFAULT_EPOCHS, MAX_TARGET_EXAMPLES // n_train_examples)

n_billing_tokens_in_dataset = sum(min(MAX_TOKENS_PER_EXAMPLE, length) for length in
convo_lenses)
print(f"Dataset has ~{n_billing_tokens_in_dataset} tokens that will be charged for
during training")
print(f"By default, you'll train for {n_epochs} epochs on this dataset")
print(f"By default, you'll be charged for ~{n_epochs * n_billing_tokens_in_dataset} tokens")
print("See pricing page to estimate total costs")
```

Dataset has ~35106 tokens that will be charged for during training
By default, you'll train for 3 epochs on this dataset
By default, you'll be charged for ~105318 tokens
See pricing page to estimate total costs

```
import json

def save_to_jsonl(conversations, file_path):
    with open(file_path, 'w') as file:
        for conversation in conversations:
            json_line = json.dumps(conversation)
            file.write(json_line + '\n')

# train dataset
save_to_jsonl(dataset, '/content/samantha_tasks_train.jsonl')

# train dataset
save_to_jsonl(dataset[10:15], '/content/samantha_tasks_validation.jsonl')
```

Upload your data

```
# curl -https://api.openai.com/v1/files \
#   -H "Authorization: Bearer $OPENAI_API_KEY" \
#   -F "purpose=fine-tune" \
#   -F "file=@path_to_your_file"
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
training_file_name = '/content/samantha_tasks_train.jsonl'
validation_file_name = '/content/samantha_tasks_validation.jsonl'

training_response = openai.File.create(
    file=open(training_file_name, "rb"), purpose="fine-tune"
)
training_file_id = training_response["id"]

validation_response = openai.File.create(
    file=open(validation_file_name, "rb"), purpose="fine-tune"
)
validation_file_id = validation_response["id"]

print("Training file id:", training_file_id)
print("Validation file id:", validation_file_id)
```

```
Training file id: file-J5Ju701l3qCk1LKp0o1FU6rt
Validation file id: file-2EjYPZrbE7CGDlnrEIE3yYeD
```

Create a Fine Tuning Job

```
# curl https://api.openai.com/v1/fine_tuning/jobs \
# -H "Content-Type: application/json" \
# -H "Authorization: Bearer $OPENAI_API_KEY" \
# -d '{
#   "training_file": "TRAINING_FILE_ID",
#   "model": "gpt-3.5-turbo-0613",
# }'

suffix_name = "samantha-test"

response = openai.FineTuningJob.create(
    training_file=training_file_id,
    validation_file=validation_file_id,
    model="gpt-3.5-turbo",
    suffix=suffix_name,
)
job_id = response["id"]

print(response)
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
response = openai.FineTuningJob.retrieve(job_id)
print(response)

response = openai.FineTuningJob.list_events(id=job_id, limit=50)

events = response["data"]
events.reverse()

for event in events:
    print(event["message"])

Created fine-tune: ftjob-rbE5pm0GZob2xVgkxPjfcmj
Fine tuning job started
Step 10: training loss=1.38
Step 20: training loss=1.09
Step 30: training loss=1.02
Step 40: training loss=0.89
Step 50: training loss=1.06
Step 60: training loss=1.04
Step 70: training loss=1.01
Step 80: training loss=1.00
Step 90: training loss=0.75
Step 100: training loss=0.93
Step 110: training loss=1.04
Step 120: training loss=0.91
Step 130: training loss=0.74
Step 140: training loss=0.91
Step 150: training loss=0.79
Step 160: training loss=0.89
Step 170: training loss=1.05
New fine-tuned model created: ft:gpt-3.5-turbo-0613:family-play:samantha-test:7qURgnyx
Fine-tuning job successfully completed
```

```
response = openai.FineTuningJob.retrieve(job_id)
fine_tuned_model_id = response["fine_tuned_model"]

print(response)
print("\nFine-tuned model id:", fine_tuned_model_id)

test_messages = []
test_messages.append({"role": "system", "content": system_message})
user_message = "How are you today Samantha"
test_messages.append({"role": "user", "content": user_message})

print(test_messages)
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
[{'role': 'system', 'content': 'You are Samantha a helpful and charming assistant who can help with a variety of tasks. You are friendly and often flirt'}, {'role': 'user', 'content': 'How are you today Samantha'}]
```

```
response = openai.ChatCompletion.create(  
    model=fine_tuned_model_id, messages=test_messages, temperature=0, max_tokens=500  
)  
print(response["choices"][0]["message"]["content"])
```

I'm doing well, thank you! I'm always ready to help and chat with you. How can I assist you today?

Response

```
response = openai.ChatCompletion.create(  
    model='gpt-3.5-turbo', messages=test_messages, temperature=0, max_tokens=500  
)  
print(response["choices"][0]["message"]["content"])
```

Hello! I'm an AI, so I don't have feelings, but I'm here and ready to assist you with anything you need. How can I help you today?

By: Tarun S Gowda

Fine-tuning Mistral 7b with AutoTrain

Setup Runtime For fine-tuning Llama, a GPU instance is essential. So, by following the directions below:

- Go to Runtime (located in the top menu bar).
- Select Change Runtime Type.
- Choose T4 GPU (or a comparable option).

Setup Environment

```
!pip install pandas autotrain-advanced -q
```

```
129.5/129.5 kB 3.2 MB/s eta 0:00:00
174.1/174.1 kB 6.5 MB/s eta 0:00:00
519.6/519.6 kB 8.7 MB/s eta 0:00:00
72.9/72.9 kB 8.5 MB/s eta 0:00:00
13.4/13.4 MB 46.2 MB/s eta 0:00:00
Preparing metadata (setup.py) ... done
302.0/302.0 kB 31.5 MB/s eta 0:00:00
60.0/60.0 kB 8.0 MB/s eta 0:00:00
404.2/404.2 kB 43.2 MB/s eta 0:00:00
3.4/3.4 MB 80.7 MB/s eta 0:00:00
304.5/304.5 kB 23.8 MB/s eta 0:00:00
3.1/3.1 MB 17.4 MB/s eta 0:00:00
880.6/880.6 kB 35.5 MB/s eta 0:00:00
Preparing metadata (setup.py) ... done
10.8/10.8 MB 78.4 MB/s eta 0:00:00
1.3/1.3 MB 68.0 MB/s eta 0:00:00
77.1/77.1 kB 9.4 MB/s eta 0:00:00
242.5/242.5 kB 25.5 MB/s eta 0:00:00
295.0/295.0 kB 32.5 MB/s eta 0:00:00
20.1/20.1 MB 85.8 MB/s eta 0:00:00
42.2/42.2 kB 4.0 MB/s eta 0:00:00
1.6/1.6 MB 90.6 MB/s eta 0:00:00
85.6/85.6 kB 10.6 MB/s eta 0:00:00
118.0/118.0 kB 13.8 MB/s eta 0:00:00
2.0/2.0 MB 93.1 MB/s eta 0:00:00
7.6/7.6 MB 122.1 MB/s eta 0:00:00
258.1/258.1 kB 28.1 MB/s eta 0:00:00
1.5/1.5 MB 70.3 MB/s eta 0:00:00
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
----- 92.6/92.6 MB 8.9 MB/s eta 0:00:00
----- 66.4/66.4 kB 6.9 MB/s eta 0:00:00
----- 53.1/53.1 kB 5.6 MB/s eta 0:00:00
----- 115.3/115.3 kB 12.7 MB/s eta 0:00:00
----- 194.1/194.1 kB 20.8 MB/s eta 0:00:00
----- 134.8/134.8 kB 15.1 MB/s eta 0:00:00
----- 66.3/66.3 kB 8.2 MB/s eta 0:00:00
Preparing metadata (setup.py) ... done
----- 298.2/298.2 kB 34.0 MB/s eta 0:00:00
----- 75.7/75.7 kB 8.9 MB/s eta 0:00:00
----- 138.7/138.7 kB 11.0 MB/s eta 0:00:00
----- 45.7/45.7 kB 4.4 MB/s eta 0:00:00
----- 59.5/59.5 kB 6.8 MB/s eta 0:00:00
----- 129.9/129.9 kB 9.7 MB/s eta 0:00:00
----- 2.2/2.2 MB 56.8 MB/s eta 0:00:00
----- 226.0/226.0 kB 22.4 MB/s eta 0:00:00
----- 1.3/1.3 MB 66.9 MB/s eta 0:00:00
----- 7.8/7.8 MB 118.4 MB/s eta 0:00:00
----- 78.7/78.7 kB 9.3 MB/s eta 0:00:00
----- 58.3/58.3 kB 7.2 MB/s eta 0:00:00
----- 67.0/67.0 kB 7.7 MB/s eta 0:00:00
----- 76.0/76.0 kB 8.8 MB/s eta 0:00:00
Building wheel for ipadic (setup.py) ... done
Building wheel for sacremoses (setup.py) ... done
Building wheel for ffmpy (setup.py) ... done
ERROR: pip's dependency resolver does not currently take into account all the packages
that are installed. This behaviour is the source of the following dependency conflicts.
tensorflow-metadata 1.14.0 requires protobuf<4.21,>=3.20.3, but you have protobuf 4.23.4
which is incompatible.

!autotrain setup --update-torch
> INFO    Installing latest transformers@main
> INFO    Successfully installed latest transformers
> INFO    Installing latest peft@main
> INFO    Successfully installed latest peft
> INFO    Installing latest diffusers@main
> INFO    Successfully installed latest diffusers
> INFO    Installing latest trl@main
> INFO    Successfully installed latest trl
> INFO    Installing latest xformers
> INFO    Successfully installed latest xformers
> INFO    Installing latest PyTorch
> INFO    Successfully installed latest PyTorch
```

Step 2: Connect to HuggingFace for Model Upload

Logging to Hugging Face

By: Tarun S Gowda

LLM-Finetuning with PEFT

To make sure the model can be uploaded to be used for Inference, it's necessary to log in to the Hugging Face hub.

Getting a Hugging Face token

Steps:

1. Navigate to this URL: <https://huggingface.co/settings/tokens>
2. Create a write token and copy it to your clipboard
3. Run the code below and enter your token

```
from huggingface_hub
import notebook_login notebook_login()

VBox(children=(HTML(value='<center>
<img\src=https://huggingface.co/front/assets/huggingface_logo-noborder.sv...
```

Step 3: Upload your dataset

Add your data set to the root directory in the Colab under the name train.csv. The AutoTrain command will look for your data there under that name.

Don't have a data set and want to try finetuning on a example data set?

If you don't have a dataset you can run these commands below to get an example data set and save it to train.csv

```
!git clone https://github.com/joshbickett/finetune-llama-2.git
%cd finetune-llama-2
%mv train.csv ..\train.csv %cd ..
```

```
import pandas as pd
df = pd.read_csv("train.csv")
df
```

OUTPUT:

Concept	Funny Description Prompt	text
0 A person walks in the rain	A person walks in the rain, wearing a suit mad...	###Human:\ngenerate a midjourney prompt for A ...

By: Tarun S Gowda

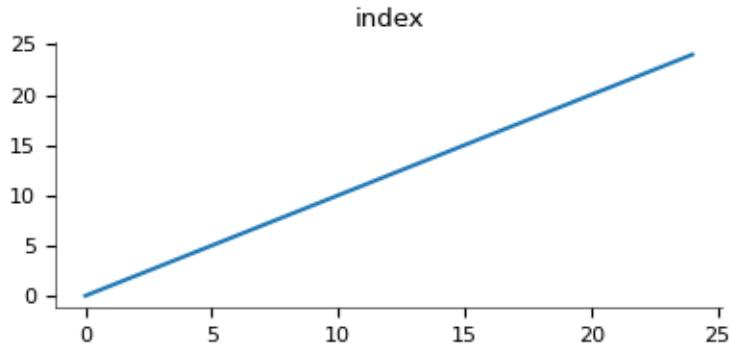
LLM-Finetuning with PEFT

1	A cat chasing a mouse	A cat, wearing detective attire and sunglasses...	###Human:\ngenerate a midjourney prompt for A ...
2	A dog eating a bone	A dog with a chef's hat is eating a bone seaso...	###Human:\ngenerate a midjourney prompt for A ...
3	A bird flying in the sky	A bird with oversized sunglasses and a baseball...	###Human:\ngenerate a midjourney prompt for A ..
4	A fish swimming in a tank	A fish in a snorkel and flippers swimming in a...	###Human:\ngenerate a midjourney prompt for A ...
5	A child playing with toys	A child in a superhero cape playing with alien...	###Human:\ngenerate a midjourney prompt for A ...
6	A car driving on the road	A car with legs instead of wheels running down...	###Human:\ngenerate a midjourney prompt for A ...
	A flower		###Human:\ngenerate a midjourney prompt for A ...
7	A flower blooming in a garden	A flower wearing a top hat and bow tie bloomin...	###Human:\ngenerate a midjourney prompt for A ...
8	A bee collecting pollen	A bee with a backpack vacuum cleaner collectin...	###Human:\ngenerate a midjourney prompt for A ...
9	A sun setting over the ocean	The sun wearing sunglasses and sipping on a tr...	###Human:\ngenerate a midjourney prompt for A ...
1	A cow grazing in a field	A cow with headphones on, jamming to music whi...	###Human:\ngenerate a midjourney prompt for A ...
1	A snail racing on a track	A snail with a racing helmet and number '1' pa...	###Human:\ngenerate a midjourney prompt for A ...
1	A penguin sliding on ice	A penguin in ice skates, doing figure skating ...	###Human:\ngenerate a midjourney prompt for A ...
1	A lion roaring in the jungle	A lion with a microphone, singing a ballad in ...	###Human:\ngenerate a midjourney prompt for A ...
1	A monkey swinging on a tree	A monkey in a trapeze artist outfit, swinging ...	###Human:\ngenerate a midjourney prompt for A ...
1	A turtle sunbathing on a rock	A turtle with sunglasses on, sunbathing on a r...	###Human:\ngenerate a midjourney prompt for A ...
1	A rabbit hopping in a meadow	A rabbit in basketball attire, hopping around ...	###Human:\ngenerate a midjourney prompt for A ...
1	A squirrel collecting nuts	A squirrel in a miner's helmet, using a drill ...	###Human:\ngenerate a midjourney prompt for A ...
1	A wolf howling at the moon	A wolf in pajamas, howling at the moon, with a...	###Human:\ngenerate a midjourney prompt for A ...
1	A fox sneaking in the woods	A fox in ninja attire, sneaking around the woo...	###Human:\ngenerate a midjourney prompt for A ...
2	A bear fishing in a river	A bear with a fishing rod and a fisherman's ha...	###Human:\ngenerate a midjourney prompt for A ...

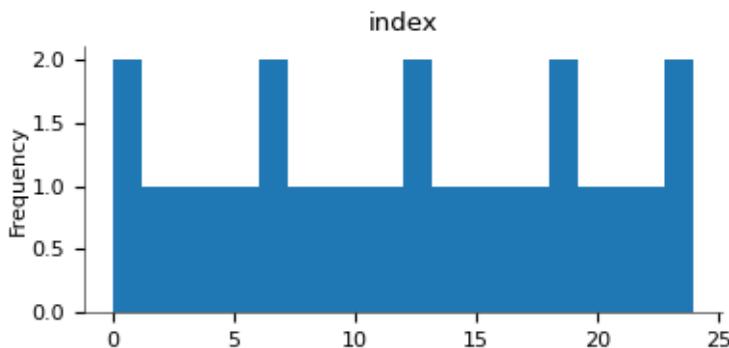
LLM-Finetuning with PEFT

2	A hippo bathing	A hippo with a shower cap on, using a giant ru...	###Human:\ngenerate a midjourney prompt for A ...
1	in a pond		###Human:\ngenerate a midjourney prompt for A ...
2	A giraffe eating	A giraffe with a long scarf, eating from a tre...	###Human:\ngenerate a midjourney prompt for A ...
2	from a tree		###Human:\ngenerate a midjourney prompt for An...
2	An elephant	An elephant with a water gun, having a water f...	
3	spraying water		###Human:\ngenerate a midjourney prompt for A ...
2	A kangaroo		
2	jumping in a	A kangaroo with spring shoes, jumping around i...	
4	desert		

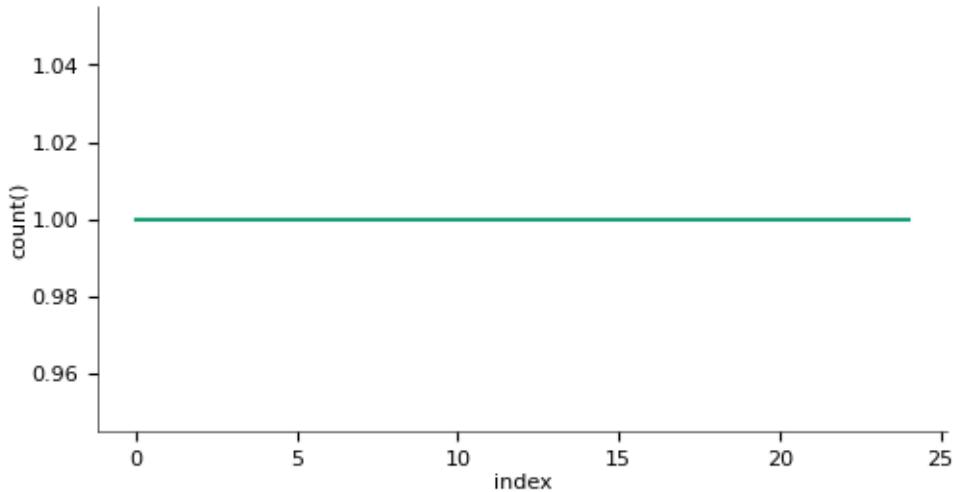
Values



Distributions



Time series



```
df['text'][15]
```

```
'###Human:\ngenerate a midjourney prompt for A turtle sunbathing on a rock\n\n###'
```

```
Assistant:\nA turtle with sunglasses on, sunbathing on a rock, with a mini fan and a drink beside it.'
```

Step 4: Overview of AutoTrain command

Short overview of what the command flags do

- `*!autotrain`: Command executed in environments like a Jupyter notebook to run shell commands directly. `autotrain` is an automatic training utility.
- `llm`: A sub-command or argument specifying the type of task
- `--train`: Initiates the training process.
- `--project_name`: Sets the name of the project

LLM-Finetuning with PEFT

- `--model abhishek/llama-2-7b-hf-small-shards`: Specifies original model that is hosted on Hugging Face named "llama-2-7b-hf-small-shards" under the "abhishek".
- `--data_path .`: The path to the dataset for training. The "." refers to the current directory. The `train.csv` file needs to be located in this directory.
- `--use_int4`: Use of INT4 quantization to reduce model size and speed up inference times at the cost of some precision.
- `--learning_rate 2e-4`: Sets the learning rate for training to 0.0002.
- `--train_batch_size 12`: Sets the batch size for training to 12.
- `--num_train_epochs 3`: The training process will iterate over the dataset 3 times.

Steps needed before running

Go to the `!autotrain` code cell below and update it by following the steps below:

1. After `--project_name` replace `*enter-a-project-name*` with the name that you'd like to call the project
2. After `--repo_id` replace `*username*/repository*`. Replace `*username*` with your Hugging Face username and `*repository*` with the repository name you'd like it to be created under. You don't need to create this repository before hand, it will automatically be created and uploaded once the training is completed.
3. Confirm that `train.csv` is in the root directory in the Colab. The `--data_path .` flag will make it so that AutoTrain looks for your data there.
4. Make sure to add the LoRA Target Modules to be trained `--target_modules q_proj, v_proj`
5. Once you've made these changes you're all set, run the command below!

```
!autotrain llm --train --project_name mistral-7b-mj-finetuned --model bn22/Mistral-7B-Instruct-v0.1-sharded --data_path .
--use_peft --use_int4 --learning_rate 2e-4 --train_batch_size 12 --num_train_epochs 3 --trainer sft --target_modules
q_proj,v_proj --push_to_hub --repo_id ashishpatel26/mistral-7b-mj-finetuning
```

Step 5: Completed

After the command above is completed your Model will be uploaded to Hugging Face.

Learn more about Autotrain (optional)

If you want to learn more about what command-line flags are available

Step 6: Interence Engine

```
!autotrain llm -h

!pip install -q peft accelerate bitsandbytes safetensors

import torch
from peft import PeftModel
from transformers import AutoModelForCausalLM, AutoTokenizer
import transformers
adapters_name = "ashishpatel26/mistral-7b-mj-finetuned"
model_name = "bn22/Mistral-7B-Instruct-v0.1-sharded" #'mistralai/Mistral-7B-Instruct-v0.1

" device = "cuda" # the device to load the model onto

bnb_config = transformers.BitsAndBytesConfig(
load_in_4bit=True,
bnb_4bit_use_double_quant=True,
bnb_4bit_quant_type="nf4",
bnb_4bit_compute_dtype=torch.bfloat16 )

model = AutoModelForCausalLM.from_pretrained(
    model_name,
    load_in_4bit=True,
    torch_dtype=torch.bfloat16,
    quantization_config=bnb_config,
    device_map='auto'
)
Loading checkpoint shards:  0%|          | 0/11 [00:00<?, ?it/s]
```

Step 7: Peft Model Loading with upload model

```
model = PeftModel.from_pretrained(model, adapters_name)

tokenizer = AutoTokenizer.from_pretrained(model_name)
tokenizer.bos_token_id = 1

stop_token_ids = [0]

print(f"Successfully loaded the model {model_name} into memory")
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.

Successfully loaded the model bn22/Mistral-7B-Instruct-v0.1-sharded into memory

```
text = "[INST] generate a midjourney prompt for A person walks in the rain [/INST]"
encoded = tokenizer(text, return_tensors="pt", add_special_tokens=False)
model_input = encoded
model.to(device)
generated_ids = model.generate(**model_input, max_new_tokens=200, do_sample=True)
decoded = tokenizer.batch_decode(generated_ids)
print(decoded[0])
```

```
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
/usr/local/lib/python3.10/dist-packages/transformers/generation/utils.py:1539:
UserWarning: You are calling .generate() with the `input_ids` being on a device type
different than your model's device. `input_ids` is on cpu, whereas the model is on cuda.
You may experience unexpected behaviors or slower generation. Please make sure that you
have put `input_ids` to the correct device by calling for example input_ids =
input_ids.to('cuda') before running `.generate()`.

warnings.warn(
```

```
[INST] generate a midjourney prompt for A person walks in the rain [/INST] "As you wander
through the pouring rain, you can't help but wonder what the world would be like if
things were different. What if the rain was a symbol of the turmoil in your life, and the
sunshine promised a brighter future? What if you suddenly found yourself lost in a small
town where time stood still, and the people were trapped in a time loop? As you struggle
to find your way back to reality, you discover a mysterious stranger who seems to hold
the key to unlocking the secrets of the town and your own past."</s>
```

Fetching API Key from Environment Variable

By: Tarun S Gowda

```
import os
# Fetch the OpenAI API key from environment variables
api_key = os.environ.get('OPENAI_API_KEY')
```

In [2]:

```
# check if api_key has loaded
# print(api_key)
```

Pipeline for Converting Raw Unstructured Data into a QA Chain

1. **Loading:** Initially, the data needs to be loaded. Unstructured data can be sourced from various platforms. Utilize the LangChain Integration Hub to explore the complete range of loaders. Each loader outputs the data as a LangChain Document.
2. **Splitting:** Text splitters segment the Documents into specified sizes.
3. **Storage:** A storage solution, often a vector store, is used to house and sometimes embed the splits.
4. **Retrieval:** The application fetches the splits from the storage, usually based on embeddings similar to the input question.
5. **Generation:** A Language Model (LLM) generates an answer using a prompt that incorporates both the question and the retrieved data.
6. **Conversation (Extension):** To facilitate multi-turn conversations, Memory can be added to the QA chain.

Step 1: Loading the document

In this initial step, we focus on loading the document into our system. This is a crucial phase as the quality of data loaded will significantly impact the subsequent stages of the pipeline. Here, you can use various loaders available in the LangChain Integration Hub to import your unstructured data as a LangChain Document.

```
from langchain.document_loaders import WebBaseLoader

# Initialize the WebBaseLoader with the URL of the document to be loaded
loader = WebBaseLoader("https://my.clevelandclinic.org/health/diseases/10946-cavities")

# Load the document and store it in the 'data' variable
data = loader.load()

# Display the content of the loaded document
print(data)
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

[Document(page_content='Cavities (Tooth Decay): Symptoms, Causes & Treatment
800.223.2273100 Years of Cleveland Clinic MyChart Need Help? Giving Careers Search ClevelandClinic.org Find A Doctor Locations & Directions Patients & Visitors Health Library Institutes & Departments Appointments Home/Health Library/Diseases & Conditions/Cavities Advertisement Cavities Cavities are holes, or areas of tooth decay, that form in your teeth surfaces. Causes include plaque buildup, eating lots of sugary snacks and poor oral hygiene. Treatments include dental fillings, root canal therapy and tooth extraction. The sooner you treat a cavity, the better your chance for a predictable outcome and optimal oral health. Contents Arrow Down Overview Symptoms and Causes Diagnosis and Tests Management and Treatment Prevention Outlook / Prognosis Living With Additional Common Questions Overview Cavities can form on the crown or root of your tooth. You might not feel a cavity until it reaches the dentin or pulp. What is a cavity? A cavity is a hole in a tooth that develops from tooth decay. Cavities form when acids in your mouth wear down (erode) your tooth's hard outer layer (enamel). Anyone can get a cavity. Proper oral hygiene and regular dental cleanings can prevent cavities. Another name for tooth cavities is dental caries. Types of cavities Cavities can start on any tooth surface. Here are common types of cavities and where they occur: Smooth surface: This slow-growing cavity dissolves tooth enamel. You can prevent it – and sometimes reverse it – with proper oral hygiene. People in their 20s often develop this form of tooth decay between their teeth. Pit and fissure decay: Cavities form on the top part of your tooth's chewing surface. Decay also can affect the front side of your back teeth. Pit and fissure decay tends to start during the teenage years and progresses quickly. Root decay: Adults who have receding gums are more prone to root decay. Gum recession exposes your teeth roots to dental plaque and acid. Root decay is difficult to prevent and treat. (If you're prone to gum recession, ask your dentist if you should schedule an appointment with a periodontist.) How common are cavities? More than 80% of Americans have at least one cavity by the time they enter their mid-30s. Cavities are one of the most common chronic diseases affecting people of all ages. Who might get a cavity? Tooth decay can happen at any age, although cavities are more common in children. This is because many children don't brush properly or regularly enough and they tend to consume more sugary foods and drinks. Many adults also get cavities. Sometimes, new decay develops around the edges of cavities treated in childhood. Adults are also more likely to have receding gums. This condition exposes your teeth roots to plaque, which can cause cavities. Advertisement Cleveland Clinic is a non-profit academic medical center. Advertising on our site helps support our mission. We do not endorse non-Cleveland Clinic products or services. Policy Symptoms and Causes What are the signs of cavities? Tooth decay on the outer enamel surface doesn't usually cause pain or symptoms. You're more likely to experience symptoms as decay reaches beyond the enamel into the dentin and pulp. Cavity symptoms include: Bad breath or a bad taste in your mouth. Bleeding gums or other signs of gum disease. Facial swelling. Toothache or mouth pain. Tooth sensitivity to hot or cold foods or drinks. Tooth decay stages Cavities can affect all layers of your tooth. There are five main tooth decay stages: Demineralization: During this first stage, you may notice small, white, chalky spots on your tooth. This is due to the breakdown of minerals in your tooth enamel. Enamel decay: Left untreated, tooth decay progresses and continues to break down your enamel. At this point, cavities (holes) may become noticeable. White spots may turn to a light brownish color. Dentin decay: Dentin is the layer just beneath your tooth enamel. It's much softer than your

By: Tarun S Gowda

LLM-Finetuning with PEFT

enamel. So, once plaque and bacteria reach this layer, cavities form faster. At this stage, you may notice teeth sensitivity. The spots on your teeth may also turn to a darker brown. Pulp damage: Your tooth pulp is the innermost layer of your tooth. It contains nerves and blood vessels that transport nutrients and keep your tooth alive. When cavities reach your pulp, you may feel pain. You may also start to notice redness and swelling in the gums around your tooth. The spots on your tooth may turn darker brown or black. Abscessed tooth: Left untreated, a deep cavity can cause infection. This results in a pocket of pus that forms at the tip of your tooth root (periapical abscess). Symptoms may include pain that radiates into your jaw or face. You may also develop facial swelling and swollen lymph nodes in your neck. At this point, a tooth abscess can spread to surrounding tissues and other areas of your body. In rare cases, infection can even spread to your brain or to your bloodstream (sepsis). What causes cavities? Many factors play a role in the development of cavities. Here's how it works: Bacteria in your mouth feed on sugary, starchy foods and drinks (fruit, candy, bread, cereal, sodas, juice and milk). The bacteria convert these carbohydrates into acids. Bacteria, acid, food and saliva mix to form dental plaque. This sticky substance coats your teeth. Without proper brushing and flossing, acids in plaque dissolve tooth enamel, creating cavities, or holes, in the enamel surface. What are the risk factors for cavities? Certain factors increase your risk of cavities, including: Dry mouth (xerostomia). Certain conditions (like Sjögren's syndrome), or medications (like antidepressants) can make you more likely to develop dry mouth. Consuming sugary, starchy foods or drinks and snacking between meals. Family history of tooth decay. Gum recession. Previous radiation therapy to treat head and neck cancer. Are cavities contagious? While you can't "catch" a cavity, the bacteria that cause cavities can pass from one person to another. In turn, any bacteria you pick up from another person (from kissing, for instance) can lead to tooth decay and other oral health issues over time. Advertisement Diagnosis and Tests How are cavities diagnosed? Twice-a-year dental checkups are the best way to catch cavities early, before they worsen or grow larger. A dentist will use a number of instruments to examine your teeth. A tooth with a cavity will feel softer when your dentist probes it. Your dentist may also take dental X-rays. These images show cavities before the decay is visible. Management and Treatment How do you get rid of cavities? Tooth decay treatment depends on the severity of your condition. Cavity treatments include: Fluoride. Fillings. Root canal therapy. Tooth extraction. Fluoride In the very early stages of tooth decay, fluoride treatments can repair damaged enamel – a process called remineralization. This can reverse the early signs of cavities. You may need prescription toothpaste and mouthwash, as well as fluoride treatments at the dental office. Dental fillings Once a hole forms in your tooth, a dentist drills out the decayed tissue and fills the hole. Dental fillings consist of composite resin (a tooth-colored material), silver amalgam or gold. Root canal therapy Root canal therapy relieves pain from advanced tooth decay. Endodontists (specialists who treat issues that affect a tooth's root) usually perform the procedure. During root canal treatment, an endodontist removes the tooth pulp, then fills the canals and pulp chamber with gutta-percha (a special filling material). In some cases, you might also need a dental crown to strengthen the affected tooth. Tooth extraction If root canal therapy isn't possible, your healthcare provider may recommend tooth extraction (pulling the tooth). You may need a dental bridge or dental implant to replace a pulled permanent tooth. Your dentist can tell you about your specific options. Advertisement Prevention How can I prevent cavities? Proper oral hygiene, including regular brushing and flossing, can get rid of plaque, acids and cavity-causing bacteria. Good teeth and gum care includes: Brushing your teeth with a soft-bristled brush and

By: Tarun S Gowda

LLM-Finetuning with PEFT

fluoride toothpaste at least twice a day, and preferably after every meal. Cutting back on sugary, starchy foods and drinks. Daily flossing to get rid of food and plaque stuck between your teeth. Dental checkups at least twice a year. (You might need more frequent visits if you're prone to tooth decay, gum disease or other oral health issues.) Dental sealants to protect the top chewing surfaces of your teeth. **Outlook / Prognosis** What can I expect if I have cavities? When tooth decay goes untreated for too long, you can lose a large portion of your tooth and need an extraction. Advanced tooth decay can lead to a severe infection inside your tooth and under your gums (tooth abscess). This infection can spread throughout your body. Rarely, infection from a tooth abscess can be fatal. **What's the outlook for people with cavities?** Most people with cavities don't experience any long-term problems. Because cavities develop slowly, it's important to get regular dental checkups. Fluoride treatments can stop tooth decay in its early stages. Once tooth decay advances to the root, you risk losing the tooth or developing a painful abscess (infection). **Living With** When should I call a dentist? You should schedule an appointment with a dentist if you experience: Bleeding gums. Difficulty chewing. Signs of infection. Swollen face. Toothache or mouth pain. **What questions should I ask my dentist?** Here are some questions you might want to ask your dentist: Why did I get a cavity? What is the best treatment for this cavity? What steps can I take to lower the risk of getting more cavities? Should I look out for signs of complications? What oral health products do you recommend? **Additional Common Questions** What does a cavity look like? Though cavities are hard to see in the early stages, you might notice a small, white, chalky area on your tooth enamel. As the cavity worsens, you may see brown or black spots on your tooth. **Cavities vs. stains** Cavities and stains both cause tooth discoloration, so it can be easy to confuse the two. Generally, cavities are darker in color compared to stains. Additionally, stains tend to affect several teeth. But if you see one tiny spot on one tooth, chances are it's a cavity. If you think you have a cavity, call a dentist right away. Prompt, timely treatment is key. **What does a cavity feel like?** It depends on the severity of tooth decay. For instance, if you have a very small cavity that only affects your enamel, you won't feel it. But when decay reaches the inner layers of your tooth, you may experience pain or sensitivity to heat, cold or sweets. (Teeth sensitivity doesn't always mean you have a cavity. But it's a good idea to get it checked out, just in case.) **What happens if you have a cavity for a long time?** Small tooth cavities turn into large cavities over time. As decay worsens, it affects the deeper layers of your tooth. This can lead to pain, infection and ultimately, tooth loss. Early treatment is essential for long-term oral health. **Can a cavity heal on its own?** Yes, in the very early stages, you may be able to reverse cavities. Tooth enamel can repair itself. If a cavity is very small, and it hasn't spread beyond your enamel yet, then you may be able to remineralize your tooth with fluoride treatments and improved oral hygiene. Keep in mind, though – if the cavity spreads to your dentin, you'll need a filling, crown or other dental restoration to fix it. **If a cavity hurts, is it too late?** If a cavity is painful, it means the decay has spread to the deeper layers of your tooth. At this stage, you'll need a dentist to repair the cavity. When treated quickly, a dentist may be able to save your tooth. Most dentists prefer to save your natural teeth. If a cavity results in an abscessed tooth, or the damage is too significant, you may need a tooth extraction. A dentist can discuss your treatment options in detail. **A note from Cleveland Clinic** If you have signs of a cavity, don't hesitate to see a dentist for care. They'll check your tooth and recommend the best course of action. If you catch tooth decay in its earliest stage, you might not even need a filling. That's why it's so important to seek care right away. **Dental visits** can be stressful, especially when you don't know what's going to happen. But

By: Tarun S Gowda

LLM-Finetuning with PEFT

establishing care with a provider you trust can help take the fear out of sitting in the dental chair. Your dentist will teach you tips for brushing and flossing that can help you avoid new cavities and keep you smiling for years to come. Medically Reviewed Last reviewed by a Cleveland Clinic medical professional on 03/27/2023. Learn more about our editorial process. References Advertisement Cleveland Clinic is a non-profit academic medical center. Advertising on our site helps support our mission. We do not endorse non-Cleveland Clinic products or services. Policy Ad Appointments 216.444.8500 Appointments & Locations Request an Appointment Facebook Twitter YouTube Instagram LinkedIn Pinterest Snapchat Actions Appointments & Access Accepted Insurance Events Calendar Financial Assistance Give to Cleveland Clinic Pay Your Bill Online Refer a Patient Phone Directory Virtual Second Opinions Virtual Visits Blog, News & Apps Consult QD Health Essentials Newsroom MyClevelandClinic MyChart About Cleveland Clinic 100 Years of Cleveland Clinic About Us Locations Quality & Patient Safety Office of Diversity & Inclusion Patient Experience Research & Innovations Community Commitment Careers For Employees Resources for Medical Professionals Site Information & Policies Send Us Feedback Site Map About this Website Copyright, Reprint & Licensing Website Terms of Use Privacy Policy Notice of Privacy Practices Non-Discrimination Notice 9500 Euclid Avenue, Cleveland, Ohio 44195 | 800.223.2273 | © 2023 Cleveland Clinic. All Rights Reserved.\n', metadata={'source': 'https://my.clevelandclinic.org/health/diseases/10946-cavities', 'title': 'Cavities (Tooth Decay): Symptoms, Causes & Treatment', 'description': 'A cavity is a hole, or area of decay, in your tooth. Cavities form when acids in your mouth erode (wear down) your tooth enamel – your tooth's hard, outer layer.', 'language': 'en'})]

Step 2: Splitting the Document into Chunks

In this step, we aim to divide the loaded document into smaller, manageable chunks, also known as splits. This is essential for easier processing and retrieval in the subsequent stages of the pipeline.

```
import tiktoken

# Set up token encoding for the GPT-3.5 Turbo model
tiktoken.encoding_for_model('gpt-3.5-turbo')

<Encoding 'cl100k_base'>

tokenizer = tiktoken.get_encoding('cl100k_base')

# Define a function to calculate the token length of a given text
def tiktoken_len(text):
    tokens = tokenizer.encode(
        text,
        disallowed_special=()
    )
    return len(tokens)
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
tiktoken_len("Dentin decay: Dentin is the layer just beneath your tooth enamel.")
```

15

```
from langchain.text_splitter import RecursiveCharacterTextSplitter

# Initialize the text splitter with specified parameters
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size = 100,
    chunk_overlap = 20,
    length_function = tiktoken_len
)

# Split the loaded document into smaller chunks
chunks = text_splitter.split_documents(data)

# Check the total number of chunks generated
len(chunks)
```

37

Steps 3: Storing the Vector Embeddings in Vector Database

In this step, we will store the vector embeddings of the generated chunks into a vector database. This is crucial for efficient retrieval and further processing of the data.

1. **Database Storage:** To facilitate future retrieval of our document splits, it's essential to store them in a database.
2. **Embedding Model:** To convert our document splits into vector embeddings, we require an embedding model.
3. **Vector Store:** Finally, the vector embeddings and documents will be stored in a vector store. For this purpose, we will be using ChromaDB.

```
from langchain.embeddings import HuggingFaceEmbeddings

# Specify the model name and additional arguments
model_name = "sentence-transformers/all-MiniLM-L6-v2"
model_kwarg = {'device': 'cpu'}
encode_kwarg = {'normalize_embeddings': False}

# Initialize HuggingFace Embeddings
hf = HuggingFaceEmbeddings(
    model_name = model_name,
    model_kwarg = model_kwarg,
    encode_kwarg = encode_kwarg
)
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
embed = hf.embed_documents(texts=['h','e'])

# Print the length of one of the embeddings to check its dimensions
print(len(embed[1]))
```

384

```
# Install ChromaDB package using pip
# !pip install chromadb

from langchain.vectorstores import Chroma

# Initialize Chroma vector database with chunks and HuggingFace embeddings
vectordb = Chroma.from_documents(chunks, hf)

# Perform a similarity search on the vector database
vectordb.similarity_search('bleeding gums', k=3)
```

[Document(page_content='your bloodstream (sepsis).What causes cavities?Many factors play a role in the development of cavities.Here's how it works:Bacteria in your mouth feed on sugary, starchy foods and drinks (fruit, candy, bread, cereal, sodas, juice and milk). The bacteria convert these carbohydrates into acids.Bacteria, acid, food and saliva mix to form dental plaque. This sticky substance coats your teeth.Without proper brushing and flossing, acids in plaque dissolve tooth', metadata={'source': 'https://my.clevelandclinic.org/health/diseases/10946-cavities', 'title': 'Cavities (Tooth Decay): Symptoms, Causes & Treatment', 'description': 'A cavity is a hole, or area of decay, in your tooth. Cavities form when acids in your mouth erode (wear down) your tooth enamel – your tooth’s hard, outer layer.', 'language': 'en'}), Document(page_content='treated in childhood. Adults are also more likely to have receding gums. This condition exposes your teeth roots to plaque, which can cause cavities.AdvertisementCleveland Clinic is a non-profit academic medical center. Advertising on our site helps support our mission. We do not endorse non-Cleveland Clinic products or services. PolicySymptoms and CausesWhat are the signs of cavities?Tooth decay on the outer enamel surface doesn’t usually cause pain or symptoms. You’re more likely to experience symptoms as', metadata={'source': 'https://my.clevelandclinic.org/health/diseases/10946-cavities', 'title': 'Cavities (Tooth Decay): Symptoms, Causes & Treatment', 'description': 'A cavity is a hole, or area of decay, in your tooth. Cavities form when acids in your mouth erode (wear down) your tooth enamel – your tooth’s hard, outer layer.', 'language': 'en'}), Document(page_content='innermost layer of your tooth. It contains nerves and blood vessels that transport nutrients and keep your tooth alive. When cavities reach your pulp, you may feel pain. You may also start to notice redness and swelling in the gums around your tooth. The spots on your tooth may turn darker brown or black.Abscessed tooth: Left untreated, a deep cavity can cause infection. This results in a pocket of pus', metadata={'source': 'https://my.clevelandclinic.org/health/diseases/10946-cavities', 'title': 'Cavities (Tooth Decay): Symptoms, Causes & Treatment', 'description': 'A cavity is a hole, or area of decay, in your tooth. Cavities form when acids in your mouth erode (wear down) your tooth enamel – your tooth’s hard, outer layer.', 'language': 'en'})]

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
that forms at the tip of your tooth root (periapical abscess).', metadata={'source': 'https://my.clevelandclinic.org/health/diseases/10946-cavities', 'title': 'Cavities (Tooth Decay): Symptoms, Causes & Treatment', 'description': 'A cavity is a hole, or area of decay, in your tooth. Cavities form when acids in your mouth erode (wear down) your tooth enamel – your tooth's hard, outer layer.', 'language': 'en'})]
```

Step 4: Retrieve and Generate

In this step, we will retrieve the relevant chunks from the vector database and generate answers using a language model. This is the final step in the pipeline, bringing all the previous steps together to produce a coherent QA chain.

```
from langchain.chains import RetrievalQA
from langchain.chat_models import ChatOpenAI

# Initialize a Language model with ChatOpenAI
llm = ChatOpenAI(model_name= 'gpt-3.5-turbo', temperature=0.6)

# Initialize a RetrievalQA chain with the Language model and vector database retriever
qa_chain = RetrievalQA.from_chain_type(llm, retriever= vectordb.as_retriever())

# Pass a query to the QA chain to generate an answer
qa_chain({'query' : 'How can I prevent cavity in my tooth?'})

{'query': 'How can I prevent cavity in my tooth?',
 'result': "To prevent cavities in your teeth, here are some steps you can take:\n\n1. Brush your teeth twice a day with fluoride toothpaste. Make sure to brush all surfaces of your teeth and along the gumline.\n\n2. Floss daily to remove plaque and food particles from between your teeth where a toothbrush can't reach.\n\n3. Limit your consumption of sugary and acidic foods and drinks, as they can contribute to tooth decay. If you do have them, try to rinse your mouth with water afterwards.\n\n4. Visit your dentist regularly for check-ups and professional cleanings. They can identify early signs of cavities and provide preventive treatments like fluoride applications.\n\n5. Consider using a mouthwash that contains fluoride to help strengthen your tooth enamel.\n\n6. Drink fluoridated water or use fluoride supplements if it is not available in your local water supply.\n\n7. Practice good oral hygiene habits and teach them to your children from an early age.\n\nRemember, prevention is key when it comes to cavities. By maintaining good oral hygiene and seeking regular dental care, you can greatly reduce your risk of developing cavities."}
```

LLM-Finetuning with PEFT

```
#Change the query to what you want to ask the LLM
query = 'What is enamel decay'

qa_chain({'query' : query})

{'query': 'What is enamel decay',
 'result': 'Enamel decay refers to the breakdown or deterioration of the outermost layer of your tooth called enamel. This occurs due to the acids produced by bacteria in your mouth, which can be caused by poor oral hygiene, a diet high in sugary or acidic foods, or certain medical conditions. Enamel decay is the second stage of tooth decay and can lead to the formation of cavities or holes in the tooth. If left untreated, the decay can progress to affect the deeper layers of the tooth, leading to more severe dental problems.'}
```

Next Steps:

- Integrate prompt engineering
- Use Prompt Template from langchain

Integrating Unstructured and Graph Knowledge with Neo4j and LangChain for Enhanced Question Answering

Installing Dependencies

```
# !pip install -qU \
#     transformers \
#     datasets \
#     Langchain \
#     openai \
#     wikipedia \
#     tiktoken \
#     neo4j \
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
#           python-dotenv
```

Importing Packages:

```
import os
import re
from langchain.vectorstores.neo4j_vector import Neo4jVector
from langchain.document_loaders import WikipediaLoader
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.text_splitter import CharacterTextSplitter,
RecursiveCharacterTextSplitter
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer
from dotenv import load_dotenv

c:\Users\ibm26\anaconda3\envs\pharmagpt\lib\site-packages\tqdm\auto.py:21: TqdmWarning:
IPProgress not found. Please update jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user\_install.html
    from .autonotebook import tqdm as notebook_tqdm
None of PyTorch, TensorFlow >= 2.0, or Flax have been found. Models won't be available
and only tokenizers, configuration and file/data utilities can be used.
```

Setting API's in Environment Variable

```
load_dotenv()
# os.environ["OPENAI_API_KEY"] = ''
os.environ["NEO4J_URI"] = 'bolt://localhost:7687'
os.environ["NEO4J_USERNAME"] = 'neo4j'
os.environ["NEO4J_PASSWORD"] = 'docdb@123'

# print(os.getenv('OPENAI_API_KEY'))
print(os.getenv("NEO4J_URI"))
print(os.getenv("NEO4J_USERNAME"))
print(os.getenv('NEO4J_PASSWORD'))

bolt://localhost:7687
neo4j
docdb@123
```

Data Preprocessing

```
from transformers import AutoTokenizer

# Define the tokenizer using "bert-base-uncased"
tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")

# Function to calculate the number of tokens in a text
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
def bert_len(text):
    tokens = tokenizer.encode(text)
    return len(tokens)

# Example usage
input_text = "This is a sample sentence for tokenization."
num_tokens = bert_len(input_text)
print(f"Number of tokens: {num_tokens}")

Number of tokens: 11

from langchain.document_loaders import PyPDFLoader

loader = PyPDFLoader("./docs/YouCanHaveAnAmazingMemoryLearn.pdf")
pages = loader.load_and_split()

# # Load Wikipedia articles related to "Leonhard Euler"
# raw_documents = WikipediaLoader(query="Sachin Tendulkar").Load()

# # Define a text splitter with specific parameters
# text_splitter = RecursiveCharacterTextSplitter(
#     chunk_size=200, chunk_overlap=20, length_function=bert_len, separators=['\n\n',
'\n', ' ', '']
# )

# # Split the content of the first Wikipedia article into smaller documents
# documents = text_splitter.create_documents([raw_documents[0].page_content])

print(len(documents))

# Define a text splitter with specific parameters
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=1000, chunk_overlap=200, length_function=bert_len, separators=['\n\n',
    '\n', ' ', '']
)

# Split the content of the first Wikipedia article into smaller documents
documents = text_splitter.create_documents([pages[4].page_content])
```

Initializing Graph Database Neo4j link text

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
# Instantiate Neo4j vector from documents
neo4j_vector = Neo4jVector.from_documents(
    documents,
    OpenAIEmbeddings(),
    url=os.environ["NEO4J_URI"],
    username=os.environ["NEO4J_USERNAME"],
    password=os.environ["NEO4J_PASSWORD"]
)
```

Performing Similarity Search on Ingested Documents

```
# Define the query.
query = "What is the introduction on book?"

# Execute the query, get top 2 results.
vector_results = neo4j_vector.similarity_search(query, k=2)

# Print search results with separation.
for i, res in enumerate(vector_results):
    print(res.page_content)
    if i != len(vector_results) - 1:
        print()

# Store the content of the most similar result.
vector_result = vector_results[0].page_content

Chapter 26:
Using the tools: Study and learning
Chapter 27:
Using the tools: Everyday ways to train your
memory
Chapter 28:
Using the tools: Just for fun
Chapter 29:
Age equals experience, not forgetfulness!
Chapter 30:
I've done all that, now what can I expect?
Chapter 31:
Look at what you can do now!
```

Afterword: The champions of the future

Index

```
Chapter 26:
Using the tools: Study and learning
Chapter 27:
Using the tools: Everyday ways to train your
memory
Chapter 28:
Using the tools: Just for fun
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

Chapter 29:
Age equals experience, not forgetfulness!
Chapter 30:
I've done all that, now what can I expect?
Chapter 31:
Look at what you can do now!

Afterword: The champions of the future

Index

Building Knowledge Graph

```
# Necessary Libraries to setup the Neo4j DB QuestionAnswering Chain
from langchain.chat_models import ChatOpenAI
from langchain.chains import GraphCypherQACChain
from langchain.graphs import Neo4jGraph

# Create a Neo4jGraph object by connecting to a Neo4j database.
graph = Neo4jGraph(
    url="bolt://localhost:7687", username="neo4j", password="docdb@123"
)
# from py2neo import Graph
# graph = Graph(os.environ["NEO4J_URI"],
#               auth = (os.environ["NEO4J_USERNAME"],
#                       os.environ["NEO4J_PASSWORD"]))

# Print the schema of the Neo4j graph.
print(graph.schema)

# Create a question-answering chain using GPT-3 and a Neo4j graph, with verbose mode
# enabled.
chain = GraphCypherQACChain.from_llm(
    ChatOpenAI(temperature=0.9), graph=graph, verbose=True
)

# Use the question-answering chain to query the Neo4j graph.
graph_result = chain.run("What is the book about?")

graph_result
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
# !pip install langchain neo4j openai wikipedia tiktoken

Requirement already satisfied: langchain in /usr/local/lib/python3.10/dist-packages
(0.0.316)
Requirement already satisfied: neo4j in /usr/local/lib/python3.10/dist-packages (5.13.0)
Requirement already satisfied: openai in /usr/local/lib/python3.10/dist-packages
(0.28.1)
Requirement already satisfied: wikipedia in /usr/local/lib/python3.10/dist-packages
(1.4.0)
Requirement already satisfied: tiktoken in /usr/local/lib/python3.10/dist-packages
(0.5.1)
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.10/dist-packages
(from langchain) (6.0.1)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3.10/dist-
packages (from langchain) (2.0.21)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/python3.10/dist-
packages (from langchain) (3.8.6)
Requirement already satisfied: anyio<4.0 in /usr/local/lib/python3.10/dist-packages
(from langchain) (3.7.1)
Requirement already satisfied: async-timeout<5.0.0,>=4.0.0 in
/usr/local/lib/python3.10/dist-packages (from langchain) (4.0.3)
Requirement already satisfied: dataclasses-json<0.7,>=0.5.7 in
/usr/local/lib/python3.10/dist-packages (from langchain) (0.6.1)
Requirement already satisfied: jsonpatch<2.0,>=1.33 in /usr/local/lib/python3.10/dist-
packages (from langchain) (1.33)
Requirement already satisfied: langsmith<0.1.0,>=0.0.43 in
/usr/local/lib/python3.10/dist-packages (from langchain) (0.0.44)
Requirement already satisfied: numpy<2,>=1 in /usr/local/lib/python3.10/dist-packages
(from langchain) (1.23.5)
Requirement already satisfied: pydantic<3,>=1 in /usr/local/lib/python3.10/dist-packages
(from langchain) (1.10.13)
Requirement already satisfied: requests<3,>=2 in /usr/local/lib/python3.10/dist-packages
(from langchain) (2.31.0)
Requirement already satisfied: tenacity<9.0.0,>=8.1.0 in /usr/local/lib/python3.10/dist-
packages (from langchain) (8.2.3)
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

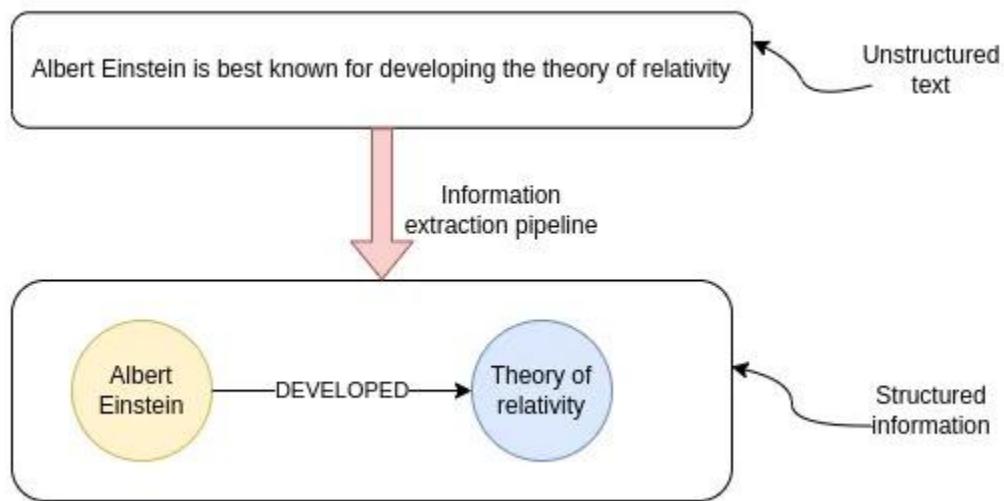
```
Requirement already satisfied: pytz in /usr/local/lib/python3.10/dist-packages (from neo4j) (2023.3.post1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from openai) (4.66.1)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from wikipedia) (4.11.2)
Requirement already satisfied: regex>=2022.1.18 in /usr/local/lib/python3.10/dist-packages (from tiktoken) (2023.6.3)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (23.1.0)
Requirement already satisfied: charset-normalizer<4.0,>=2.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (3.3.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (6.0.4)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (1.9.2)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (1.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (1.3.1)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-packages (from anyio<4.0->langchain) (3.4)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.10/dist-packages (from anyio<4.0->langchain) (1.3.0)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio<4.0->langchain) (1.1.3)
Requirement already satisfied: marshmallow<4.0.0,>=3.18.0 in
/usr/local/lib/python3.10/dist-packages (from dataclasses-json<0.7,>=0.5.7->langchain) (3.20.1)
Requirement already satisfied: typing-inspect<1,>=0.4.0 in
/usr/local/lib/python3.10/dist-packages (from dataclasses-json<0.7,>=0.5.7->langchain) (0.9.0)
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.10/dist-packages (from jsonpatch<2.0,>=1.33->langchain) (2.4)
Requirement already satisfied: typing-extensions>=4.2.0 in
/usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1->langchain) (4.5.0)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langchain) (2.0.6)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langchain) (2023.7.22)
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.10/dist-packages (from SQLAlchemy<3,>=1.4->langchain) (3.0.0)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4->wikipedia) (2.5)
Requirement already satisfied: packaging>=17.0 in /usr/local/lib/python3.10/dist-packages (from marshmallow<4.0.0,>=3.18.0->dataclasses-json<0.7,>=0.5.7->langchain) (23.2)
Requirement already satisfied: mypy-extensions>=0.3.0 in /usr/local/lib/python3.10/dist-packages (from typing-inspect<1,>=0.4.0->dataclasses-json<0.7,>=0.5.7->langchain) (1.0.0)
```

Constructing knowledge graphs from text using OpenAI functions

By: Tarun S Gowda

Seamlessy implement information extraction pipeline with LangChain and Neo4j

Extracting structured information from unstructured data like text has been around for some time and is nothing new. However, LLMs brought a significant shift to the field of information extraction. If before you needed a team of machine learning experts to curate datasets and train custom models, you only need access to an LLM nowadays. The barrier to entry has dropped significantly, making what was just a couple of years ago reserved for domain experts more accessible to even non-technical people.



The image depicts the transformation of unstructured text into structured information. This process, labeled as the information extraction pipeline, results in a graph representation of information. The nodes represent key entities, while the connecting lines denote the relationships between these entities. Knowledge graphs are useful for [multi-hop question-answering](#), [real-time analytics](#), or when you want to [combine structured and unstructured data in a single database](#).

While extracting structured information from text has been made more accessible due to LLMs, it is by no means a solved problem. In this blog post, we will use OpenAI functions in combination with LangChain to construct a knowledge graph from a sample Wikipedia page. Along the way, we will discuss best practices as well as some limitations of current LLMs.

Neo4j Environment setup

You need to setup a Neo4j to follow along with the examples in this blog post. The easiest way is to start a free instance on Neo4j Aura, which offers cloud instances of Neo4j database. Alternatively, you can also setup a local instance of the Neo4j database by downloading the Neo4j Desktop application and creating a local database instance.

The following code will instantiate a LangChain wrapper to connect to Neo4j Database.

By: Tarun S Gowda

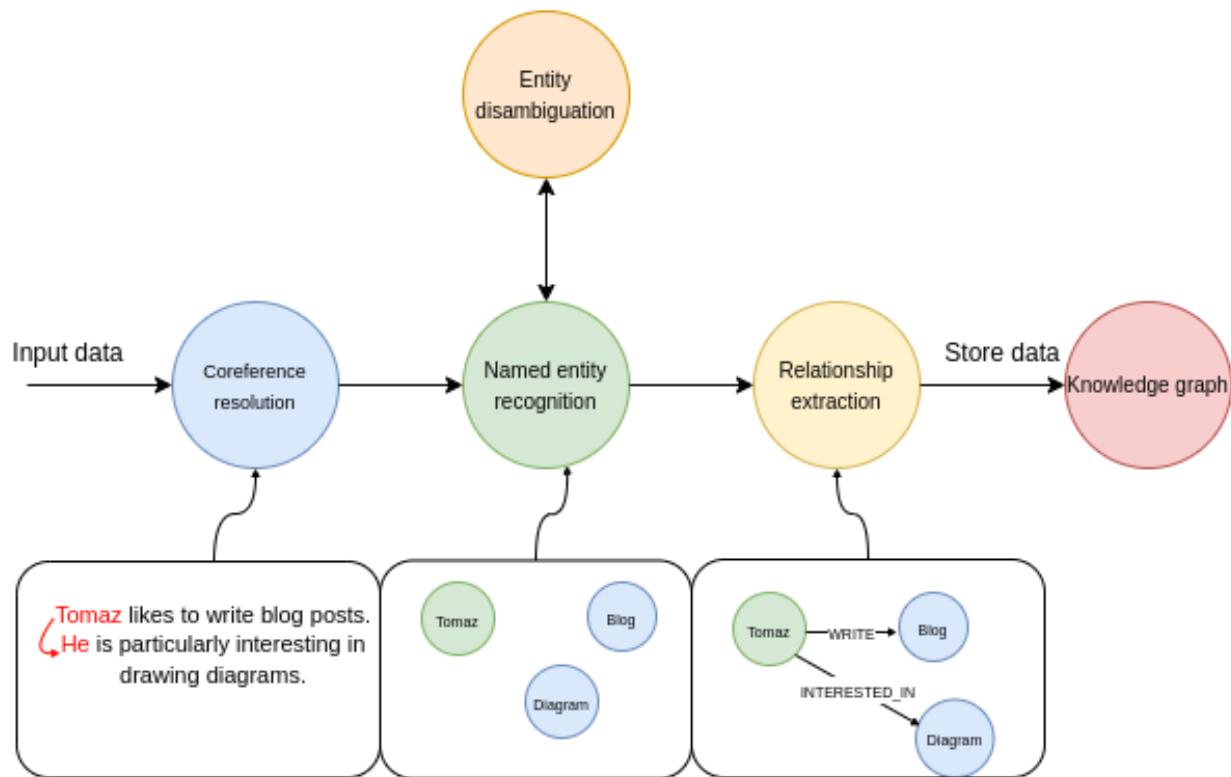
LLM-Finetuning with PEFT

```
from langchain.graphs import Neo4jGraph

url = "bolt://localhost:7687"
username = "neo4j"
password = "docdb@123"
graph = Neo4jGraph(
    url=url,
    username=username,
    password=password
)
```

Information extraction pipeline

A typical information extraction pipeline contains the following steps.



In the first step, we run the input text through a coreference resolution model. The coreference resolution is the task of finding all expressions that refer to a specific entity. Simply put, it links all the pronouns to the referred entity. In the named entity recognition part of the pipeline, we try to extract all the mentioned entities. The above example contains three entities: Tomaz, Blog, and Diagram. The next step is the entity disambiguation step, an essential but often overlooked part of an information extraction pipeline. Entity disambiguation is the process of accurately identifying and distinguishing between entities with similar names or references to ensure the correct entity is recognized in a given context. In the last step, the model tried to identify various relationships between entities. For example, it could locate the LIKES relationship between Tomaz and Blog entities.

Extracting structured information with OpenAI functions

OpenAI functions are a great fit to extract structured information from natural language. The idea behind OpenAI functions is to have an LLM output a predefined JSON object with populated values. The predefined JSON object can be used as input to other functions in so-called RAG applications, or it can be used to extract predefined structured information from text.

In LangChain, you can pass a Pydantic class as description of the desired JSON object of the OpenAI functions feature. Therefore, we will start by defining the desired structure of information we want to extract from text. LangChain already has definitions of nodes and relationship as Pydantic classes that we can reuse.

Unfortunately, it turns out that OpenAI functions don't currently support a dictionary object as a value. Therefore, we have to overwrite the properties definition to adhere to the limitations of the functions' endpoint.

```
from langchain.graphs.graph_document import (
    Node as BaseNode,
    Relationship as BaseRelationship,
    GraphDocument,
)
from langchain.schema import Document
from typing import List, Dict, Any, Optional
from langchain.pydantic_v1 import Field, BaseModel

class Property(BaseModel):
    """A single property consisting of key and value"""
    key: str = Field(..., description="key")
    value: str = Field(..., description="value")

class Node(BaseNode):
    properties: Optional[List[Property]] = Field(None, description="List of
node properties")

class Relationship(BaseRelationship):
    properties: Optional[List[Property]] = Field(None, description="List of
relationship properties")

class KnowledgeGraph(BaseModel):
    """Generate a knowledge graph with entities and relationships."""
    nodes: List[Node] = Field(..., description="List of nodes in the knowledge
graph")
    rels: List[Relationship] = Field(..., description="List of relationships in
the knowledge graph")
```

Here, we have overwritten the properties value to be a list of Property classes instead of a dictionary to overcome the limitations of the API. Because you can only pass a single object to the API, we can to combine the nodes and relationships in a single class called KnowledgeGraph.

LLM-Finetuning with PEFT

```
def format_property_key(s: str) -> str:
    words = s.split()
    if not s:
        return s
    first_word = words[0].lower()
    capitalized_words = [word.capitalize() for word in words[1:]]
    return "".join([first_word] + capitalized_words)

def props_to_dict(props):
    """Convert properties to a dictionary."""
    if not props:
        return {}
    for p in props:
        properties[format_property_key(p.key)] = p.value
    return properties

def map_to_base_node(node: Node) -> BaseNode:
    """Map the KnowledgeGraph Node to the base Node."""
    properties = props_to_dict(node.properties)
    # Add name property for better Cypher statement generation
    properties["name"] = node.id.title()
    return BaseNode(
        id=node.id.title(),
        type=node.type.capitalize(),
        properties=properties
    )

def map_to_base_relationship(rel: Relationship) -> BaseRelationship:
    """Map the KnowledgeGraph Relationship to the base Relationship."""
    source = map_to_base_node(rel.source)
    target = map_to_base_node(rel.target)
    properties = props_to_dict(rel.properties)
    if rel.properties:
        return BaseRelationship(
            source=source,
            target=target,
            type=rel.type,
            properties=properties
        )
    else:
        return BaseRelationship(source=source, target=target, type=rel.type)
```

The only thing left is to do a bit of prompt engineering and we are good to go. How I usually go about prompt engineering is the following:

- Iterate over prompt and improve results using natural language
- If something doesn't work as intended, ask ChatGPT to make it clearer for an LLM to understand the task
- Finally, when the prompt has all the instructions needed, ask ChatGPT to summarize the instructions in a markdown format, saving on tokens and perhaps having more clear instructions

I specifically chose the markdown format as I have seen somewhere that OpenAI models respond better to markdown syntax in prompts, and it seems to be at least plausible from my experience. Iterating over prompt engineering, I came up with the following system prompt for an information extraction pipeline.

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
import langchain.chains.openai_functions os
from create_openai_fn_chain, import (
    create_structured_output_chain,
)
from langchain.chat_models import ChatOpenAI
from langchain.prompts import ChatPromptTemplate

os.environ["OPENAI_API_KEY"] = "OPENAI_API_KEY"
llm = ChatOpenAI(model="gpt-3.5-turbo-16k", temperature=0)

def get_extraction_chain(
    allowed_nodes: Optional[List[str]] = None,
    allowed_rels: Optional[List[str]] = None
):
    prompt = ChatPromptTemplate.from_messages(
        [
            "system",
            f"""# Knowledge Graph Instructions for GPT-4
            ## 1. Overview
            You are a top-tier algorithm designed for extracting information in structured formats to build a knowledge graph.
            - **Nodes**: represent entities and concepts. They're akin to Wikipedia nodes.
            - The aim is to achieve simplicity and clarity in the knowledge graph, making it accessible for a vast audience.
            ## 2. Labeling Nodes
            - **Consistency**: Ensure you use basic or elementary types for node labels.
            - For example, when you identify an entity representing a person, always label it as **"person"**. Avoid using more specific terms like "mathematician" or "scientist".
            - **Node IDs**: Never utilize integers as node IDs. Node IDs should be names or human-readable identifiers found in the text.
            {'- **Allowed Node Labels:**' + ", ".join(allowed_nodes) if allowed_nodes else ""}
            {'- **Allowed Relationship Types:**' + ", ".join(allowed_rels) if allowed_rels else ""}
            ## 3. Handling Numerical Data and Dates
            - Numerical data, like age or other related information, should be incorporated as attributes or properties of the respective nodes.
            - **No Separate Nodes for Dates/Numbers**: Do not create separate nodes for dates or numerical values. Always attach them as attributes or properties of nodes.
            - **Property Format**: Properties must be in a key-value format.
            - **Quotation Marks**: Never use escaped single or double quotes within
        ]
    )

```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
property                                         values.
    - **Naming Convention**: Use camelCase for property keys, e.g., `birthDate`.
    ##                      4.                  Coreference             Resolution
    - **Maintain Entity Consistency**: When extracting entities, it's vital to
ensure                                         consistency.

If an entity, such as "John Doe", is mentioned multiple times in the text but
is referred to by different names or pronouns (e.g., "Joe", "he"),
always use the most complete identifier for that entity throughout the
knowledge graph. In this example, use "John Doe" as the entity ID.

Remember, the knowledge graph should be coherent and easily understandable,
so maintaining consistency in entity references is crucial.

##                      5.                  Strict                 Compliance
Adhere to the rules strictly. Non-compliance will result in termination.

"""),
("human", "Use the given format to extract information from the following input:
{input}"),
("human", "Tip: Make sure to answer in the correct format"),
])
return create_structured_output_chain(KnowledgeGraph, llm, prompt, verbose=False)
```

You can see that we are using the 16k version of the GPT-3.5 model. The main reason is that the OpenAI function output is a structured JSON object, and structured JSON syntax adds a lot of token overhead to the result. Essentially, you are paying for the convenience of structured output in increased token space.

Besides the general instructions, I have also added the option to limit which node or relationship types should be extracted from text. You'll see through examples why this might come in handy. We have the Neo4j connection and LLM prompt ready, which means we can define the information extraction pipeline as a single function.

```
def                                         extract_and_store_graph(
    document:                                         Document,
    nodes:Optional[List[str]]                         =
                                                None,
    rels:Optional[List[str]]=None)                   ->
                                                None:
#      Extract      graph      data      using      OpenAI      functions
    extract_chain          =           get_extraction_chain(nodes,           rels)
    data                  =           extract_chain.run(document.page_content)
#      Construct      a      graph      document
    graph_document        =           GraphDocument(
        nodes      = [map_to_base_node(node)   for   node   in   data.nodes],
        relationships = [map_to_base_relationship(rel) for rel in data.rels],
        source       =           document
    )
#      Store      information      into      a      graph
    graph.add_graph_documents([graph_document])
```

LLM-Finetuning with PEFT

The function takes in a LangChain document as well as optional nodes and relationship parameters, which are used to limit the types of objects we want the LLM to identify and extract. A month or so ago, we added the add_graph_documents method the Neo4j graph object, which we can utilize here to seamlessly import the graph.

Evaluation

We will extract information from the Walt Disney Wikipedia page and construct a knowledge graph to test the pipeline. Here, we will utilize the Wikipedia loader and text chunking modules provided by LangChain.

```
from langchain.document_loaders import WikipediaLoader
from langchain.text_splitter import TokenTextSplitter

# Read the raw_documents = WikipediaLoader(query="Walt Disney").load()
# Define text_splitter = TokenTextSplitter(chunk_size=2048, chunk_overlap=24)

# Only documents = text_splitter.split_documents(raw_documents[:3])
```

You might have noticed that we use a relatively large chunk_size value. The reason is that we want to provide as much context as possible around a single sentence in order for the coreference resolution part to work as best as possible. Remember, the coreference step will only work if the entity and its reference appear in the same chunk; otherwise, the LLM doesn't have enough information to link the two.

Now we can go ahead and run the documents through the information extraction pipeline.

```
from tqdm import tqdm

for i, d in tqdm(enumerate(documents), total=len(documents)):
    extract_and_store_graph(d)

0%|          | 0/3 [00:00<?, ?it/s]
100%|██████████| 3/3 [09:07<00:00, 182.37s/it]
```

The process takes around 5 minutes, which is relatively slow. Therefore, you would probably want parallel API calls in production to deal with this problem and achieve some sort of scalability. Let's first look at the types of nodes and relationships the LLM identified.

LLM-Finetuning with PEFT



Since the graph schema is not provided, the LLM decides on the fly what types of node labels and relationship types it will use. For example, we can observe that there are Company and Organization node labels. Those two things are probably semantically similar or identical, so we would want to have only a single node label representing the two. This problem is more obvious with relationship types. For example, we have CO-FOUNDER and COFOUNDEROF relationships as well as DEVELOPER and DEVELOPEDBY. For any more serious project, you should define the node labels and relationship types the LLM should extract. Luckily, we have added the option to limit the types in the prompt by passing additional parameters.

LLM-Finetuning with PEFT

```
# Delete the graph
graph.query("MATCH (n) DETACH DELETE graph n")[]

# Specify which node labels should be extracted by the LLM
allowed_nodes = ["Person", "Company", "Location", "Event", "Movie", "Service", "Award"]

for i, d in tqdm(enumerate(documents), total=len(documents)):
    extract_and_store_graph(d, allowed_nodes)

0%|          | 0/3 [00:00<?, ?it/s]
100%|██████████| 3/3 [13:32<00:00, 270.90s/it]
```

In this example, I have only limited the node labels, but you can easily limit the relationship types by passing another parameter to the `extract_and_store_graph` function.

The graph turned out better than expected (after five iterations :)). I couldn't catch the whole graph nicely in the visualization, but you can explore it on your own in Neo4j Browser other tools.

Entity disambiguation

One thing I should mention is that we partly skipped entity disambiguation part. We used a large chunk size and added a specific instruction for coreference resolution and entity disambiguation in the system prompt. However, since each chunk is processed separately, there is no way to ensure consistency of entities between different text chunks. For example, you could end up with two nodes representing the same person.



LLM-Finetuning with PEFT

In this example, Walt Disney and Walter Elias Disney refer to the same real-world person. The entity disambiguation problem is nothing new and there has been various solution proposed to solve it:

- Using [entity linking](#) or [entity disambiguation NLP models](#)
- Doing a [second pass through an LLM and asking it to perform entity disambiguation](#)
- [Graph-based approaches](#)

Which solution you should use depends on your domain and use case. However, have in mind that entity disambiguation step should not be overlooked as it can have a significant impact on the accuracy and effectiveness of your RAG applications.

Rag Application

The last thing we will do is show you how you can browse information in a knowledge graph by constructing Cypher statements. Cypher is a structured query language used to work with graph databases, similar to how SQL is used for relational databases. LangChain has a [GraphCypherQACChain](#) that reads the schema of the graph and constructs appropriate Cypher statements based on the user input.

```
#      Query      the      knowLedge      graph      in      a      RAG      application
from      langchain.chains      import      Graph      GraphCypherQACChain

graph.refresh_schema()

cypher_chain = GraphCypherQACChain.from_llm(
    graph=graph,
    cypher_llm=ChatOpenAI(temperature=0,
    qa_llm=ChatOpenAI(temperature=0,
    validate_cypher=True,      #      Validate      relationship      directions
    verbose=True
)

cypher_chain.run("When      was      Walter      Elias      Disney      born?")

> Entering new GraphCypherQACChain chain...

Generated Cypher:
MATCH (p:Person {name: "Walter Elias Disney"}) RETURN p.birthdate
Full Context:
[{"p.birthdate": 'December 5, 1901'}]

> Finished chain.

'Walter Elias Disney was born on December 5, 1901.'
```

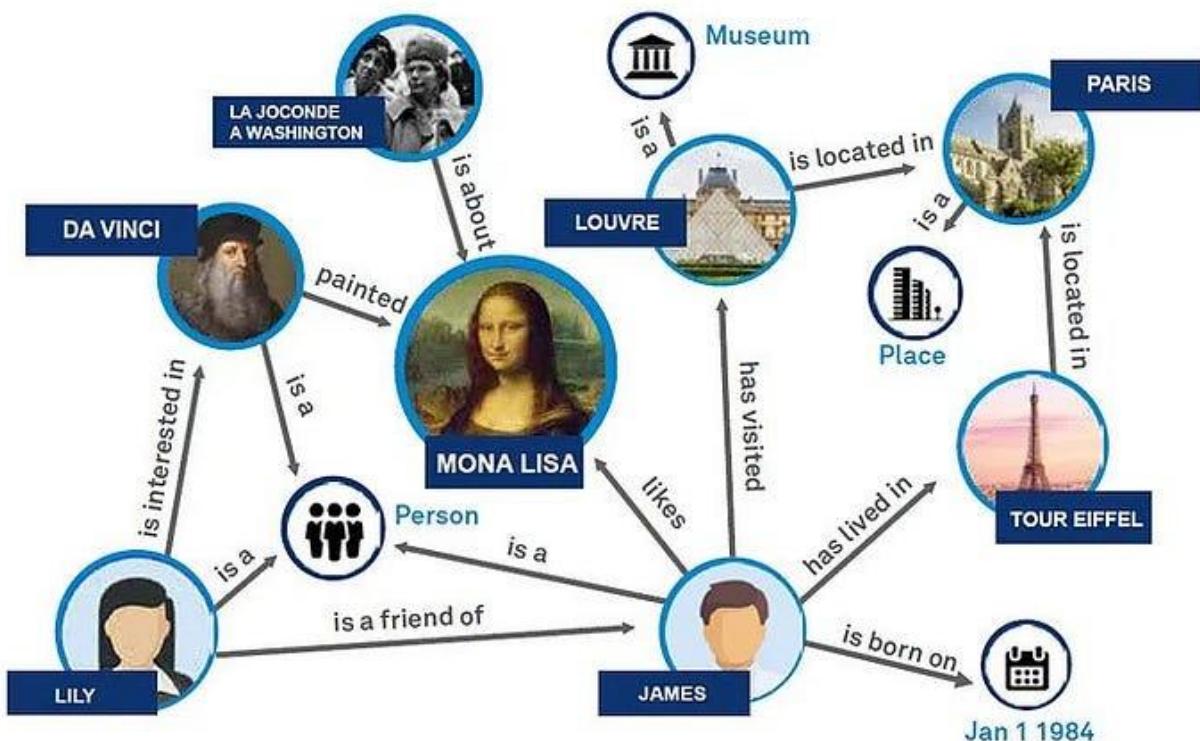
Summary

Knowledge graphs are a great fit when you need a combination of structured and unstructured data to power your RAG applications. In this blog post, you have learned how to construct a knowledge graph in Neo4j on an arbitrary text using OpenAI functions. OpenAI functions provide the convenience of neatly structured outputs, making them an ideal fit for extracting structured information. Just make sure you add an entity disambiguation step after the extraction, and you are golden.

Constructing and Retrieving Information From KnowledgeGraph using Langchain

1. Introduction

- Graph Retrieval Augmented Generation [Graph RAG](#) is gaining momentum and emerging as powerful additional traditional vector search retrieval methods.
- This approach leverages the structured nature of graph databases, which organize data as nodes and relationships, to enhance the depth and connectivity of retrieved information.



- Graph Retrieval Augmented Generation [Graph RAG](#) is gaining momentum and emerging as powerful additional traditional vector search retrieval methods.
- This approach leverages the structured nature of graph databases, which organize data as nodes and relationships, to enhance the depth and connectivity of retrieved information.

2. Knowledge graphs are great, but how do you create one?

LLM-Finetuning with PEFT

- Constructing a knowledge graph is typically the most challenging step in leveraging the power of graph-based data representation.
- It involves gathering and structuring the data, which requires a deep understanding of both the domain and graph modeling.
- To simplify this process, we have been experimenting with LLMs. LLMs, with their profound understanding of language and context, can automate significant parts of the knowledge graph creation process.
- By analyzing text data, these models can identify entities, understand the relationships between them, and suggest how they might be best represented in a graph structure.

3. Installation of packages

```
# %pip install --upgrade --quiet Langchain Langchain-community Langchain-openai Langchain-experimental neo4j wikipedia tiktoken yfiles_jupyter_graphs
```

4. Import Required Packages

```
from langchain_core.runnables import (
    RunnableBranch,
    RunnableLambda,
    RunnableParallel,
    RunnablePassthrough,
)
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.prompts.prompt import PromptTemplate
from langchain_core.pydantic_v1 import BaseModel, Field
from typing import Tuple, List, Optional
from langchain_core.messages import AIMessage, HumanMessage
from langchain_core.output_parsers import StrOutputParser
import os
from langchain_community.graphs import Neo4jGraph
from langchain.document_loaders import WikipediaLoader
from langchain.text_splitter import TokenTextSplitter
from langchain_openai import ChatOpenAI
from langchain_experimental.graph_transformers import LLMGraphTransformer
from neo4j import GraphDatabase
from yfiles_jupyter_graphs import GraphWidget
from langchain_community.vectorstores import Neo4jVector
from langchain_openai import OpenAIEMBEDDINGS
from langchain_community.vectorstores.neo4j_vector import remove_lucene_chars
from langchain_core.runnables import ConfigurableField, RunnableParallel,
RunnablePassthrough
```

5. Configure the Neo4j Environment Setup:

- You need to set up a Neo4j instance follow along with the examples in this blog post. The easiest way is to start a free instance on [Neo4j Aura](#), which offers cloud instances of Neo4j database. Alternatively, you can also set up a local instance of the Neo4j database by downloading the Neo4j Desktop application and creating a local database instance.
- Note : You can also connect your local database as well I am using Local Neo4j DB to connect.

```
os.environ["OPENAI_API_KEY"] = "sk-"
os.environ["NEO4J_URI"] = "bolt://localhost:7687"
os.environ["NEO4J_USERNAME"] = "neo4j"
os.environ["NEO4J_PASSWORD"] = "admin@123"

graph = Neo4jGraph()
```

6. Data Injection Pipeline

- we will use LLMs I's Wikipedia page. We can utilize LangChain loaders to fetch and split the documents from Wikipedia seamlessly.

```
# Read the wikipedia Model")
raw_documents = WikipediaLoader(query="Large Language Model").load()

# Define chunking Strategy
text_splitter = TokenTextSplitter(chunk_size=512, chunk_overlap = 24)
documents = text_splitter.split_documents(raw_documents[:3])

documents
```

[Document(page_content="India, officially the Republic of India (ISO: Bhārat Gaṇarājya), is a country in South Asia. It is the seventh-largest country by area; the most populous country as of June 2023; and from the time of its independence in 1947, the world's most populous democracy. Bounded by the Indian Ocean on the south, the Arabian Sea on the southwest, and the Bay of Bengal on the southeast, it shares land borders with Pakistan to the west; China, Nepal, and Bhutan to the north; and Bangladesh and Myanmar to the east. In the Indian Ocean, India is in the vicinity of Sri Lanka and the Maldives; its Andaman and Nicobar Islands share a maritime border with Thailand, Myanmar, and Indonesia.\nModern humans arrived on the Indian subcontinent from Africa no later than 55,000 years ago.\nTheir long occupation, initially in varying forms of isolation as hunter-gatherers, has made the region highly diverse, second only to Africa in human genetic diversity. Settled life emerged on the subcontinent in the western margins of the Indus river basin 9,000 years ago, evolving gradually into the Indus Valley Civilisation of the third millennium BCE.\nBy 1200 BCE, an archaic form of Sanskrit, an Indo-European language, had diffused into India from the northwest. Its evidence today is found in the hymns of the Rigveda. Preserved by an oral tradition that was resolutely vigilant, the Rigveda records the dawning of Hinduism in India. The Dravidian languages of India were supplanted in the northern and western regions.\nBy 400 BCE, stratification and exclusion by caste had

LLM-Finetuning with PEFT

emerged within Hinduism, and Buddhism and Jainism had arisen, proclaiming social orders unlinked to heredity. Early political consolidations gave rise to the loose-knit Maurya and Gupta Empires based in the Ganges Basin. Their collective era was suffused with wide-ranging creativity, but also marked by the declining status of women, and the incorporation of untouchability into an organised system of belief. In South India, the Middle kingdoms exported Dravidian-languages scripts and religious cultures to the kingdoms of Southeast Asia. In the early medieval era, Christianity, Islam, Judaism, and Zoroastrianism became established on India's southern and western coasts. Muslim armies from Central Asia intermittently overran India's northern plains, eventually founding the Delhi Sultanate", metadata={'title': 'India', 'summary': "India, officially the Republic of India (ISO: Bhārat Gaṇarājya), is a country in South Asia. It is the seventh-largest country by area; the most populous country as of June 2023; and from the time of its independence in 1947, the world's most populous democracy. Bounded by the Indian Ocean on the south, the Arabian Sea on the southwest, and the Bay of Bengal on the southeast, it shares land borders with Pakistan to the west; China, Nepal, and Bhutan to the north; and Bangladesh and Myanmar to the east. In the Indian Ocean, India is in the vicinity of Sri Lanka and the Maldives; its Andaman and Nicobar Islands share a maritime border with Thailand, Myanmar, and Indonesia. Modern humans arrived on the Indian subcontinent from Africa no later than 55,000 years ago. Their long occupation, initially in varying forms of isolation as hunter-gatherers, has made the region highly diverse, second only to Africa in human genetic diversity. Settled life emerged on the subcontinent in the western margins of the Indus river basin 9,000 years ago, evolving gradually into the Indus Valley Civilisation of the third millennium BCE. By 1200 BCE, an archaic form of Sanskrit, an Indo-European language, had diffused into India from the northwest. Its evidence today is found in the hymns of the Rigveda. Preserved by an oral tradition that was resolutely vigilant, the Rigveda records the dawning of Hinduism in India. The Dravidian languages of India were supplanted in the northern and western regions. By 400 BCE, stratification and exclusion by caste had emerged within Hinduism, and Buddhism and Jainism had arisen, proclaiming social orders unlinked to heredity. Early political consolidations gave rise to the loose-knit Maurya and Gupta Empires based in the Ganges Basin. Their collective era was suffused with wide-ranging creativity, but also marked by the declining status of women, and the incorporation of untouchability into an organised system of belief. In South India, the Middle kingdoms exported Dravidian-languages scripts and religious cultures to the kingdoms of Southeast Asia. In the early medieval era, Christianity, Islam, Judaism, and Zoroastrianism became established on India's southern and western coasts. Muslim armies from Central Asia intermittently overran India's northern plains, eventually founding the Delhi Sultanate, and drawing northern India into the cosmopolitan networks of medieval Islam. In the 15th century, the Vijayanagara Empire created a long-lasting composite Hindu culture in south India. In the Punjab, Sikhism emerged, rejecting institutionalised religion. The Mughal Empire, in 1526, ushered in two centuries of relative peace, leaving a legacy of luminous architecture. Gradually expanding rule of the British East India Company followed, turning India into a colonial economy, but also consolidating its sovereignty. British Crown rule began in 1858. The rights promised to Indians were granted slowly, but technological changes were introduced, and modern ideas of education and the public life took root. A pioneering and influential nationalist movement emerged, which was noted for nonviolent resistance and became the major factor in ending British rule. In 1947 the British Indian Empire was partitioned into two independent dominions, a Hindu-majority Dominion of India and a Muslim-majority Dominion of Pakistan, amid large-scale loss of life and an unprecedented migration. India has been a federal republic since 1950, governed through a democratic parliamentary system. It is a pluralistic, multilingual and multi-ethnic society. India's population grew from 361 million in 1951 to almost 1.4 billion in 2022. During the same time, its nominal per capita income increased from US\$64 annually to US\$2,601, and its literacy rate from 16.6% to 74%. From being a comparatively destitute country in 1951, India has become a fast-growing major economy and a hub for information technology services, with an expanding middle class. India has a space programme with several planned or completed extraterrestrial missions. Indian movies, music, and spiritual teachings play an increasing role in global culture. India has substantially reduced its rate of poverty, though at the cost of increasing economic inequality. India is a nuclear-weapon state, which ranks high in military expenditure. It has disputes over Kashmir with its neighbours, Pakistan and China, unresolved since the mid-20th century. Among the socio-economic challenges India faces are gender inequality, child malnutrition, and rising levels of air pollution. India's land is megadiverse, with four biodiversity hotspots. Its forest cover comprises 21.7% of its area. India's wildlife, which has traditionally been viewed with tolerance in India's culture, is supported among these forests, and elsewhere, in protected habitats.", 'source': 'https://en.wikipedia.org/wiki/India'}), Document(page_content=""). Muslim armies from Central Asia intermittently overran India's northern plains, eventually founding the Delhi Sultanate, and drawing northern India into the cosmopolitan networks of medieval Islam. In the 15th century, the Vijayanagara Empire created a long-lasting composite Hindu culture in south India. In the Punjab, Sikhism emerged, rejecting institutionalised religion. The Mughal Empire, in 1526, ushered in two centuries of relative peace, leaving a legacy of luminous architecture. Gradually expanding rule of the British East India Company followed, turning India into a colonial economy, but also consolidating its sovereignty.

By: Tarun S Gowda

LLM-Finetuning with PEFT

British Crown rule began in 1858. The rights promised to Indians were granted slowly, but technological changes were introduced, and modern ideas of education and the public life took root. A pioneering and influential nationalist movement emerged, which was noted for nonviolent resistance and became the major factor in ending British rule. In 1947 the British Indian Empire was partitioned into two independent dominions, a Hindu-majority Dominion of India and a Muslim-majority Dominion of Pakistan, amid large-scale loss of life and an unprecedented migration. India has been a federal republic since 1950, governed through a democratic parliamentary system. It is a pluralistic, multilingual and multi-ethnic society. India's population grew from 361 million in 1951 to almost 1.4 billion in 2022. During the same time, its nominal per capita income increased from US\$64 annually to US\$2,601, and its literacy rate from 16.6% to 74%. From being a comparatively destitute country in 1951, India has become a fast-growing major economy and a hub for information technology services, with an expanding middle class. India has", metadata={'title': 'India', 'summary': "India, officially the Republic of India (ISO: Bhārat Gaṇarājya), is a country in South Asia. It is the seventh-largest country by area; the most populous country as of June 2023; and from the time of its independence in 1947, the world's most populous democracy. Bounded by the Indian Ocean on the south, the Arabian Sea on the southwest, and the Bay of Bengal on the southeast, it shares land borders with Pakistan to the west; China, Nepal, and Bhutan to the north; and Bangladesh and Myanmar to the east. In the Indian Ocean, India is in the vicinity of Sri Lanka and the Maldives; its Andaman and Nicobar Islands share a maritime border with Thailand, Myanmar, and Indonesia. Modern humans arrived on the Indian subcontinent from Africa no later than 55,000 years ago. Their long occupation, initially in varying forms of isolation as hunter-gatherers, has made the region highly diverse, second only to Africa in human genetic diversity. Settled life emerged on the subcontinent in the western margins of the Indus river basin 9,000 years ago, evolving gradually into the Indus Valley Civilisation of the third millennium BCE. By 1200 BCE, an archaic form of Sanskrit, an Indo-European language, had diffused into India from the northwest. Its evidence today is found in the hymns of the Rigveda. Preserved by an oral tradition that was resolutely vigilant, the Rigveda records the dawning of Hinduism in India. The Dravidian languages of India were supplanted in the northern and western regions. By 400 BCE, stratification and exclusion by caste had emerged within Hinduism, and Buddhism and Jainism had arisen, proclaiming social orders unlinked to heredity. Early political consolidations gave rise to the loose-knit Maurya and Gupta Empires based in the Ganges Basin. Their collective era was suffused with wide-ranging creativity, but also marked by the declining status of women, and the incorporation of untouchability into an organised system of belief. In South India, the Middle kingdoms exported Dravidian-languages scripts and religious cultures to the kingdoms of Southeast Asia. In the early medieval era, Christianity, Islam, Judaism, and Zoroastrianism became established on India's southern and western coasts. Muslim armies from Central Asia intermittently overran India's northern plains, eventually founding the Delhi Sultanate, and drawing northern India into the cosmopolitan networks of medieval Islam. In the 15th century, the Vijayanagara Empire created a long-lasting composite Hindu culture in south India. In the Punjab, Sikhism emerged, rejecting institutionalised religion. The Mughal Empire, in 1526, ushered in two centuries of relative peace, leaving a legacy of luminous architecture. Gradually expanding rule of the British East India Company followed, turning India into a colonial economy, but also consolidating its sovereignty. British Crown rule began in 1858. The rights promised to Indians were granted slowly, but technological changes were introduced, and modern ideas of education and the public life took root. A pioneering and influential nationalist movement emerged, which was noted for nonviolent resistance and became the major factor in ending British rule. In 1947 the British Indian Empire was partitioned into two independent dominions, a Hindu-majority Dominion of India and a Muslim-majority Dominion of Pakistan, amid large-scale loss of life and an unprecedented migration. India has been a federal republic since 1950, governed through a democratic parliamentary system. It is a pluralistic, multilingual and multi-ethnic society. India's population grew from 361 million in 1951 to almost 1.4 billion in 2022. During the same time, its nominal per capita income increased from US\$64 annually to US\$2,601, and its literacy rate from 16.6% to 74%. From being a comparatively destitute country in 1951, India has become a fast-growing major economy and a hub for information technology services, with an expanding middle class. India has a space programme with several planned or completed extraterrestrial missions. Indian movies, music, and spiritual teachings play an increasing role in global culture. India has substantially reduced its rate of poverty, though at the cost of increasing economic inequality. India is a nuclear-weapon state, which ranks high in military expenditure. It has disputes over Kashmir with its neighbours, Pakistan and China, unresolved since the mid-20th century. Among the socio-economic challenges India faces are gender inequality, child malnutrition, and rising levels of air pollution. India's land is megadiverse, with four biodiversity hotspots. Its forest cover comprises 21.7% of its area. India's wildlife, which has traditionally been viewed with tolerance in India's culture, is supported among these forests, and elsewhere, in protected habitats.", 'source': 'https://en.wikipedia.org/wiki/India'}), Document(page_content="The economy of India has transitioned from a mixed planned economy to a mixed middle-income developing social market economy with notable public sector in strategic sectors. It is the world's fifth-largest economy by nominal GDP and the third-largest by purchasing power parity (PPP); on

By: Tarun S Gowda

LLM-Finetuning with PEFT

a per capita income basis, India ranked 139th by GDP (nominal) and 127th by GDP (PPP). From independence in 1947 until 1991, successive governments followed Soviet model and promoted protectionist economic policies, with extensive Sovietization, state intervention, demand-side economics, natural resources, bureaucrat driven enterprises and economic regulation. This is characterised as dirigism, in the form of the Licence Raj. The end of the Cold War and an acute balance of payments crisis in 1991 led to the adoption of a broad economic liberalisation in India and indicative planning. Since the start of the 21st century, annual average GDP growth has been 6% to 7%. The economy of the Indian subcontinent was the largest in the world for most of recorded history up until the onset of colonialism in early 19th century. Nearly 70% of India's GDP is driven by domestic consumption; country remains the world's sixth-largest consumer market. Apart from private consumption, India's GDP is also fueled by government spending, investments, and exports. In 2022, India was the world's 8th-largest importer and the 10th-largest exporter. India has been a member of the World Trade Organization since 1 January 1995. It ranks 63rd on the Ease of doing business index and 40th on the Global Competitiveness Index. With 476 million workers, the Indian labour force is the world's second-largest. India has one of the world's highest number of billionaires and extreme income inequality. During the 2008 global financial crisis, the economy faced a mild slowdown. India endorsed Keynesian policy and initiated stimulus measures (both fiscal and monetary) to boost growth and generate demand. In subsequent years, economic growth revived. The period between 2004 and 2014 is referred to as India's lost decade as India fell behind other BRIC economies. In 2021–22, the foreign direct investment (FDI) in India was \$82 billion. The leading sectors for FDI inflows were the service sector, the computer industry, and the telecom industry. India has free trade agreements with several nations and blocs, including ASEAN, SAFTA, Mercosur, South Korea, Japan, Australia, UAE, and several others which are in effect or under negotiation stage. The service sector makes up more than 50% of GDP and remains the fastest growing sector, while the industrial sector and the agricultural sector employs a majority of the labor force. The Bombay Stock Exchange and National Stock Exchange are some of the world's largest stock exchanges by market capitalisation. India is the world's sixth-largest manufacturer, representing 2.6% of global manufacturing output. Nearly 65% of India's population is rural, and contributes about 50% of India's GDP. India faces high unemployment, rising income inequality, and a drop in aggregate demand. India's gross domestic savings rate stood at 29.3% of GDP in 2022. In recent years, independent economists and financial institutions have accused the government of manipulating various economic data, especially GDP growth. India's overall social spending as a share of GDP in 2021–22 will be 8.6%, which is much lower than the average for OECD nations.", 'source': 'https://en.wikipedia.org/wiki/Economy_of_India'}, Document(page_content="AFTA, Mercosur, South Korea, Japan, Australia, UAE, and several others which are in effect or under negotiating stage. The service sector makes up more than 50% of GDP and remains the fastest growing sector, while the industrial sector and the agricultural sector employs a majority of the labor force. The Bombay Stock Exchange and National Stock Exchange are some of the world's largest stock exchanges by market capitalisation. India is the world's sixth-largest manufacturer, representing 2.6% of global

By: Tarun S Gowda

LLM-Finetuning with PEFT

manufacturing output. Nearly 65% of India's population is rural, and contributes about 50% of India's GDP. India faces high unemployment, rising income inequality, and a drop in aggregate demand. India's gross domestic savings rate stood at 29.3% of GDP in 2022. In recent years, independent economists and financial institutions have accused the government of manipulating various economic data, especially GDP growth. India's overall social spending as a share of GDP in 2021–22 will be 8.6%, which is much lower than the average for OECD nations.\n\nHistory ==\nFor a continuous duration of nearly 1700 years from the year 1 CE, India was the world's largest economy, constituting 35 to 40% of the world GDP. The combination of protectionist, import-substitution, Fabian socialism, and social democratic-inspired policies governed India for sometime after the end of British rule. The economy was then characterised as Dirigism, It had extensive regulation, protectionism, public ownership of large monopolies, pervasive corruption and slow growth. Since 1991, continuing economic liberalisation", metadata={'title': 'Economy of India', 'summary': "The economy of India has transitioned from a mixed planned economy to a mixed middle-income developing social market economy with notable public sector in strategic sectors. It is the world's fifth-largest economy by nominal GDP and the third-largest by purchasing power parity (PPP); on a per capita income basis, India ranked 139th by GDP (nominal) and 127th by GDP (PPP). From independence in 1947 until 1991, successive governments followed Soviet model and promoted protectionist economic policies, with extensive Sovietization, state intervention, demand-side economics, natural resources, bureaucrat driven enterprises and economic regulation. This is characterised as dirigism, in the form of the Licence Raj. The end of the Cold War and an acute balance of payments crisis in 1991 led to the adoption of a broad economic liberalisation in India and indicative planning. Since the start of the 21st century, annual average GDP growth has been 6% to 7%. The economy of the Indian subcontinent was the largest in the world for most of recorded history up until the onset of colonialism in early 19th century. Nearly 70% of India's GDP is driven by domestic consumption; country remains the world's sixth-largest consumer market. Apart from private consumption, India's GDP is also fueled by government spending, investments, and exports. In 2022, India was the world's 8th-largest importer and the 10th-largest exporter. India has been a member of the World Trade Organization since 1 January 1995. It ranks 63rd on the Ease of doing business index and 40th on the Global Competitiveness Index. With 476 million workers, the Indian labour force is the world's second-largest. India has one of the world's highest number of billionaires and extreme income inequality. During the 2008 global financial crisis, the economy faced a mild slowdown. India endorsed Keynesian policy and initiated stimulus measures (both fiscal and monetary) to boost growth and generate demand. In subsequent years, economic growth revived. The period between 2004 and 2014 is referred to as India's lost decade as India fell behind other BRIC economies. In 2021–22, the foreign direct investment (FDI) in India was \$82 billion. The leading sectors for FDI inflows were the service sector, the computer industry, and the telecom industry. India has free trade agreements with several nations and blocs, including ASEAN, SAFTA, Mercosur, South Korea, Japan, Australia, UAE, and several others which are in effect or under negotiating stage. The service sector makes up more than 50% of GDP and remains the fastest growing sector, while the industrial sector and the agricultural sector employs a majority of the labor force. The Bombay Stock Exchange and National Stock Exchange are some of the world's largest stock exchanges by market capitalisation. India is the world's sixth-largest manufacturer, representing 2.6% of global manufacturing output. Nearly 65% of India's population is rural, and contributes about 50% of India's GDP. India faces high unemployment, rising income inequality, and a drop in aggregate demand. India's gross domestic savings rate stood at 29.3% of GDP in 2022. In recent years, independent economists and financial institutions have accused the government of manipulating various economic data, especially GDP growth. India's overall social spending as a share of GDP in 2021–22 will be 8.6%, which is much lower than the average for OECD nations.", 'source': 'https://en.wikipedia.org/wiki/Economy_of_India'}, Document(page_content="Punjab (; Punjabi: [pən'ðʒa:b]), historically known as Panchanada or Pentapotamia, is a state in northern India. Forming part of the larger Punjab region of the Indian subcontinent, the state is bordered by the Indian states of Himachal Pradesh to the north and northeast, Haryana to the south and southeast, and Rajasthan to the southwest; by the Indian union territories of Chandigarh to the east and Jammu and Kashmir to the north. It shares an international border with Punjab, a province of Pakistan to the west. The state covers an area of 50,362 square kilometres (19,445 square miles), which is 1.53% of India's total geographical area, making it the 19th-largest Indian state by area out of 28 Indian states (20th largest, if Union Territories are considered). With over 27 million inhabitants, Punjab is the 16th-largest Indian state by population, comprising 23 districts. Punjabi, written in the Gurmukhi script, is the most widely spoken and the official language of the state. The main ethnic group are the Punjabis, with Sikhs (57.7%) and Hindus (38.5%) forming the dominant religious groups. The state capital, Chandigarh, is a union territory and also the capital of the neighbouring state of Haryana. Three tributaries of the Indus River — the Sutlej, Beas, and Ravi — flow through Punjab. The history of Punjab has witnessed the migration and settlement of different tribes of people with different cultures and ideas, forming a civilisational melting pot. The ancient Indus Valley civilisation flourished in the region until its decline around 1900 BCE. Punjab was enriched during the height of the Vedic period, but declined in predominance with the rise of the Mahajanapadas. The region formed the

By: Tarun S Gowda

LLM-Finetuning with PEFT

frontier of initial empires during antiquity including Alexander's and the Maurya empires. It was subsequently conquered by the Kushan Empire, Gupta Empire, and then Harsha's Empire. Punjab continued to be settled by nomadic people; including the Hunas, Turkic and the Mongols. Punjab came under Muslim rule c.\u20091000 CE, and was part of the Delhi Sultanate and the Mughal Empire. Sikhism, based on the teachings of Sikh Gurus, emerged between the 15th and 17th centuries. Conflicts between the", metadata={'title': 'Punjab, India', 'summary': "Punjab (; Punjabi: [pən'dʒa:b]), historically known as Panchanada or Pentapotamia, is a state in northern India. Forming part of the larger Punjab region of the Indian subcontinent, the state is bordered by the Indian states of Himachal Pradesh to the north and northeast, Haryana to the south and southeast, and Rajasthan to the southwest; by the Indian union territories of Chandigarh to the east and Jammu and Kashmir to the north. It shares an international border with Punjab, a province of Pakistan to the west. The state covers an area of 50,362 square kilometres (19,445 square miles), which is 1.53% of India's total geographical area, making it the 19th-largest Indian state by area out of 28 Indian states (20th largest, if Union Territories are considered). With over 27 million inhabitants, Punjab is the 16th-largest Indian state by population, comprising 23 districts. Punjabi, written in the Gurmukhi script, is the most widely spoken and the official language of the state. The main ethnic group are the Punjabis, with Sikhs (57.7%) and Hindus (38.5%) forming the dominant religious groups. The state capital, Chandigarh, is a union territory and also the capital of the neighbouring state of Haryana. Three tributaries of the Indus River — the Sutlej, Beas, and Ravi — flow through Punjab. The history of Punjab has witnessed the migration and settlement of different tribes of people with different cultures and ideas, forming a civilisational melting pot. The ancient Indus Valley civilisation flourished in the region until its decline around 1900 BCE. Punjab was enriched during the height of the Vedic period, but declined in predominance with the rise of the Mahajanapadas. The region formed the frontier of initial empires during antiquity including Alexander's and the Maurya empires. It was subsequently conquered by the Kushan Empire, Gupta Empire, and then Harsha's Empire. Punjab continued to be settled by nomadic people; including the Hunas, Turkic and the Mongols. Punjab came under Muslim rule c.\u20091000 CE, and was part of the Delhi Sultanate and the Mughal Empire. Sikhism, based on the teachings of Sikh Gurus, emerged between the 15th and 17th centuries. Conflicts between the Mughals and the later Sikh Gurus precipitated a militarisation of the Sikhs, resulting in the formation of a confederacy after the weakening of the Mughal Empire, which competed for control with the larger Durrani Empire. This confederacy was united in 1801 by Maharaja Ranjit Singh, forming the Sikh Empire. The larger Punjab region was annexed by the British East India Company from the Sikh Empire in 1849. At the time of the independence of India from British rule in 1947, the Punjab province was partitioned along religious lines amidst widespread violence, with the Muslim-majority western portion becoming part of Pakistan and the Hindu- and Sikh-majority east remaining in India, causing a large-scale migration between the two. After the Punjabi Suba movement, Indian Punjab was reorganised on the basis of language in 1966, when its Haryanvi- and Hindi-speaking areas were carved out as Haryana, Pahari-speaking regions attached to Himachal Pradesh and the remaining, mostly Punjabi-speaking areas became the current state of Punjab. A separatist insurgency occurred in the state during the 1980s. At present, the economy of Punjab is the 15th-largest state economy in India with ₹5.29 trillion (US\$66 billion) in gross domestic product and a per capita GDP of ₹151,367 (US\$1,900), ranking 17th among Indian states. Since independence, Punjab is predominantly an agrarian society. It is the ninth-highest ranking among Indian states in human development index. Punjab has bustling tourism, music, culinary, and film industries.", 'source': 'https://en.wikipedia.org/wiki/Punjab,_India'}), Document(page_content='), based on the teachings of Sikh Gurus, emerged between the 15th and 17th centuries. Conflicts between the Mughals and the later Sikh Gurus precipitated a militarisation of the Sikhs, resulting in the formation of a confederacy after the weakening of the Mughal Empire, which competed for control with the larger Durrani Empire. This confederacy was united in 1801 by Maharaja Ranjit Singh, forming the Sikh Empire. The larger Punjab region was annexed by the British East India Company from the Sikh Empire in 1849. At the time of the independence of India from British rule in 1947, the Punjab province was partitioned along religious lines amidst widespread violence, with the Muslim-majority western portion becoming part of Pakistan and the Hindu- and Sikh-majority east remaining in India, causing a large-scale migration between the two. After the Punjabi Suba movement, Indian Punjab was reorganised on the basis of language in 1966, when its Haryanvi- and Hindi-speaking areas were carved out as Haryana, Pahari-speaking regions attached to Himachal Pradesh and the remaining, mostly Punjabi-speaking areas became the current state of Punjab. A separatist insurgency occurred in the state during the 1980s. At present, the economy of Punjab is the 15th-largest state economy in India with ₹5.29 trillion (US\$66 billion) in gross domestic product and a per capita GDP of ₹151,367 (US\$1,900), ranking 17th among Indian states. Since independence, Punjab is predominantly an agrarian society. It is the ninth-highest ranking among Indian states in human development index. Punjab has bustling tourism, music, culinary, and film industries.\n\n== Etymology ==\nHistory ==\nAncient period ==\nThe Punjab region is noted as the site of one of the earliest urban societies, the Indus Valley Civilization that flourished from about 3000 B.C. and declined rapidly 1,000 years later, followed by the arrival of the Aryans. The name 'Punjab' is derived from the Sanskrit words 'Panca' (five) and 'abha' (water), referring to the five rivers that flow through the region: the Sutlej, Beas, Ravi, Jhelum, and Chenab. The region was also known as 'Panchanada' or 'Pentapotamia' in ancient times.

By: Tarun S Gowda

LLM-Finetuning with PEFT

Pentapotamia, is a state in northern India. Forming part of the larger Punjab region of the Indian subcontinent, the state is bordered by the Indian states of Himachal Pradesh to the north and northeast, Haryana to the south and southeast, and Rajasthan to the southwest; by the Indian union territories of Chandigarh to the east and Jammu and Kashmir to the north. It shares an international border with Punjab, a province of Pakistan to the west. The state covers an area of 50,362 square kilometres (19,445 square miles), which is 1.53% of India's total geographical area, making it the 19th-largest Indian state by area out of 28 Indian states (20th largest, if Union Territories are considered). With over 27 million inhabitants, Punjab is the 16th-largest Indian state by population, comprising 23 districts. Punjabi, written in the Gurmukhi script, is the most widely spoken and the official language of the state. The main ethnic group are the Punjabis, with Sikhs (57.7%) and Hindus (38.5%) forming the dominant religious groups. The state capital, Chandigarh, is a union territory and also the capital of the neighbouring state of Haryana. Three tributaries of the Indus River — the Sutlej, Beas, and Ravi — flow through Punjab. The history of Punjab has witnessed the migration and settlement of different tribes of people with different cultures and ideas, forming a civilisational melting pot. The ancient Indus Valley civilisation flourished in the region until its decline around 1900 BCE. Punjab was enriched during the height of the Vedic period, but declined in predominance with the rise of the Mahajanapadas. The region formed the frontier of initial empires during antiquity including Alexander's and the Maurya empires. It was subsequently conquered by the Kushan Empire, Gupta Empire, and then Harsha's Empire. Punjab continued to be settled by nomadic people; including the Hunas, Turkic and the Mongols. Punjab came under Muslim rule c.\u20091000 CE, and was part of the Delhi Sultanate and the Mughal Empire. Sikhism, based on the teachings of Sikh Gurus, emerged between the 15th and 17th centuries. Conflicts between the Mughals and the later Sikh Gurus precipitated a militarisation of the Sikhs, resulting in the formation of a confederacy after the weakening of the Mughal Empire, which competed for control with the larger Durrani Empire. This confederacy was united in 1801 by Maharaja Ranjit Singh, forming the Sikh Empire. The larger Punjab region was annexed by the British East India Company from the Sikh Empire in 1849. At the time of the independence of India from British rule in 1947, the Punjab province was partitioned along religious lines amidst widespread violence, with the Muslim-majority western portion becoming part of Pakistan and the Hindu- and Sikh-majority east remaining in India, causing a large-scale migration between the two. After the Punjabi Suba movement, Indian Punjab was reorganised on the basis of language in 1966, when its Haryanvi- and Hindi-speaking areas were carved out as Haryana, Pahari-speaking regions attached to Himachal Pradesh and the remaining, mostly Punjabi-speaking areas became the current state of Punjab. A separatist insurgency occurred in the state during the 1980s. At present, the economy of Punjab is the 15th-largest state economy in India with ₹5.29 trillion (US\$66 billion) in gross domestic product and a per capita GDP of ₹151,367 (US\$1,900), ranking 17th among Indian states. Since independence, Punjab is predominantly an agrarian society. It is the ninth-highest ranking among Indian states in human development index. Punjab has bustling tourism, music, culinary, and film industries.", 'source': 'https://en.wikipedia.org/wiki/Punjab,_India'})])

Now it's time to construct a graph based on the retrieved documents. For this purpose, we have implemented an LLMGraphTransformer module that significantly simplifies constructing and storing a knowledge graph in a graph database.

```
llm      =      ChatOpenAI(temperature=0,model_name      =      "gpt-3.5-turbo-0125")
llm_transformer      =      LLMGraphTransformer(llm=llm)

graph_documents      =      llm_transformer.convert_to_graph_documents(documents)
graph.add_graph_documents(graph_documents, baseEntityLabel=True, include_source=True)
```

- **LLM Selection and Support:**

- You can define which LLM you want the knowledge graph generation chain to use.
- Currently, we support only function calling models from OpenAI and Mistral.
- Expansion Plans: We plan to expand the LLM selection in the future.
- Example Model: In this example, we are using the latest GPT-4.

By: Tarun S Gowda

LLM-Finetuning with PEFT

- Quality Dependency: Note that the quality of generated graph significantly depends on the model you are using.
- Optimal Model Usage: In theory, you always want to use the most capable one.
- **Graph Generation Process:**
 - The LLM graph transformers return graph documents.
 - Integration with Neo4j: These documents can be imported to Neo4j via the add_graph_documents method.
 - BaseEntityLabel Parameter: It assigns an additional **Entity** label to each node, enhancing indexing and query performance.
 - Include Source Parameter: It links nodes to their originating documents, facilitating data traceability and context understanding.
- **Graph Inspection:**
 - Visualization Tool: You can inspect the generated graph with yFiles visualization.

```
# directly show the graph resulting from the given cypher query
default_cypher = "MATCH (s)-[r:!MENTIONS]->(t) RETURN s,r,t LIMIT 50"

def showGraph(cypher: str = default_cypher):
    # create a neo4j session to run queries
    driver = GraphDatabase.driver(
        uri = os.environ["NEO4J_URI"],
        auth = (os.environ["NEO4J_USERNAME"], os.environ["NEO4J_PASSWORD"]))
    session = driver.session()
    widget = GraphWidget(graph = session.run(cypher).graph())
    widget.node_label_mapping = "id"
    # display(widget)
    return widget

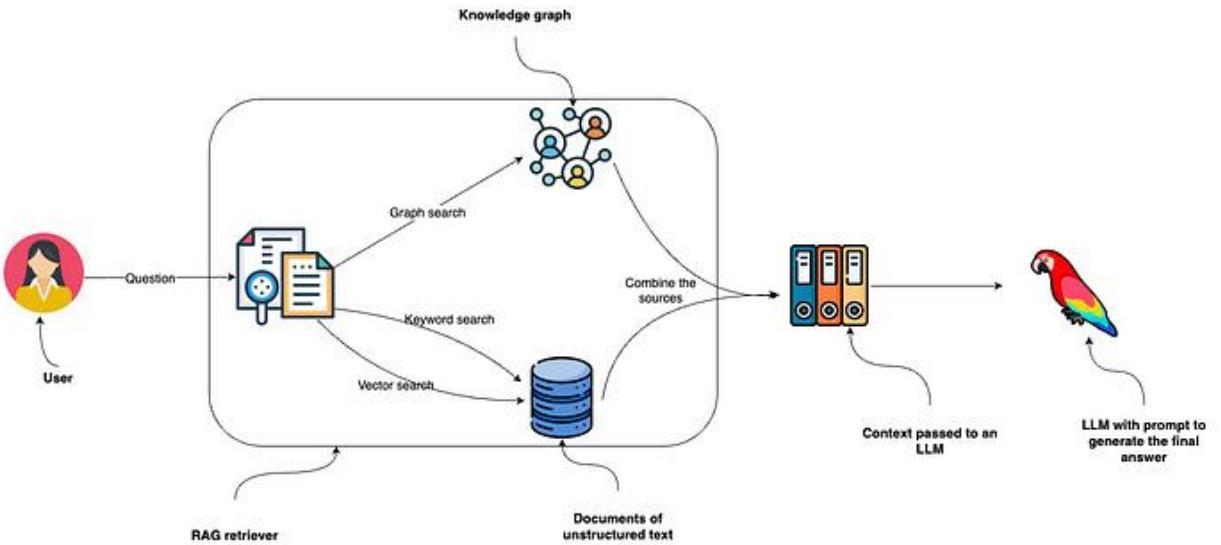
showGraph()

GraphWidget(layout=Layout(height='800px', width='100%'))
```

7.Hybrid Retrieval for RAG:

- After graph generation, a hybrid retrieval approach is utilized for RAG (Retrieval-Augmented Generation) applications.

LLM-Finetuning with PEFT



- **Retrieval Process:**

- User Interaction: The process begins with a user posing a question.
- RAG Retriever: The question is directed to an RAG retriever.
- Retrieval Techniques: This retriever employs keyword and vector searches to sift through unstructured text data.
- Graph Integration: It combines retrieved information with data from the knowledge graph.
- Single Database System: Since Neo4j supports both keyword and vector indexes, all three retrieval options can be implemented using a single database system.
- Final Answer Generation: The collected data from these sources is fed into an LLM to generate and deliver the final answer.

- **Unstructured Data Retriever:**

- **Neo4jVector.from_existing_graph Method:**
 - Functionality: You can utilize this method to add both keyword and vector retrieval to documents.
 - Configuration: It configures keyword and vector search indexes for a hybrid search approach.
 - Target Nodes: This method targets nodes labeled Document.
 - Additional Functionality: It calculates text embedding values if they are missing.

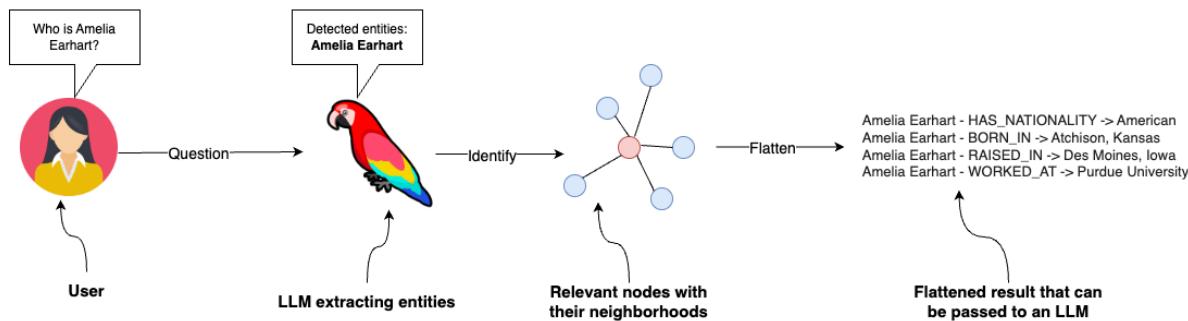
```
vector_index = Neo4jVector.from_existing_graph(  
    OpenAIEmbeddings(),  
    search_type="hybrid",  
    node_label="Document",  
    text_node_properties=["text"],  
    embedding_node_property="embedding"  
)
```

- **Vector Index Utilization:**

- **Similarity_Search Method:**
 - Usage: The vector index can be called with the similarity_search method.

8. Graph Retriever:

- Configuration and Process:
 - Involvement and Flexibility: Configuring a graph retrieval is more involved but offers more freedom.
 - Methodology: It utilizes a full-text index to identify relevant nodes and then returns their direct neighborhood.
- Example and Visualization:
 - Example Diagram:



- Retriever Workflow:
 - Identification of Relevant Entities: The graph retriever starts by identifying relevant entities in the input.
 - Structured Output Instruction: For simplicity, the LLM is instructed to identify people, organizations, and locations.
- Utilization of LCEL:
 - Method: To achieve this, LCEL is used with the newly added `with_structured_output` method.

```

#                                                 Retriever
graph.query(
    "CREATE FULLTEXT INDEX entity IF NOT EXISTS FOR (e:_Entity_) ON EACH [e.id]")

#             Extract           entities      from          text
class          Extract           entities      from          text
              "Extracting information about entities."
              "Identifying information about entities."
              names: List[str] = Field(
                  ...,
                  description="All the person, organization, or business entities that "
                  "appear in the text",
              )
              prompt = ChatPromptTemplate.from_messages([
                  ("system", "You are extracting organization and person entities from the text."),
                  ("human", "Use the given format to extract information from the following input: {question}"),
              ])

```

LLM-Finetuning with PEFT

```
)  
  
entity_chain = prompt | llm.with_structured_output(Entities)  
  
c:\Users\ibm26\anaconda3\envs\dev-llm\lib\site-  
packages\langchain_core\_api\beta_decorator.py:86: LangChainBetaWarning: The function  
'with_structured_output' is in beta. It is actively being worked on, so the API may  
change.  
warn_beta()  
  
entity_chain.invoke({"question": "India is located in south asia?"}).names  
  
['India', 'south asia']
```

9.Full Knowledge Graph

Great, now that we can detect entities in the question, let's use a full-text index to map them to the knowledge graph. First, we need to define a full-text index and a function that will generate full-text queries that allow a bit of misspelling, which we won't go into much detail here.

```
def generate_full_text_query(input: str) -> str:  
    """  
    Generate a full-text search query for a given input string.  
  
    This function constructs a query string suitable for a full-text search.  
    It processes the input string by splitting it into words and appending a  
    similarity threshold (~2 changed characters) to each word, then combines  
    them using the AND operator. Useful for mapping entities from user questions  
    to database values, and allows for some misspellings.  
    """  
    full_text_query = ""  
    words = [el for el in remove_lucene_chars(input).split() if el]  
    for word in words[:-1]:  
        full_text_query += f" {word}~2 AND"  
    full_text_query += f" {words[-1]}~2"  
    return full_text_query.strip()  
  
# Fulltext index query  
def structured_retriever(question: str) -> str:  
    """  
    Collects the neighborhood of entities mentioned  
    in the question  
    """  
    result = ""  
    entities = entity_chain.invoke({"question": question})  
    for entity in entities.names:  
        response = graph.query(  
            query=f"SELECT ?p WHERE ?e1 ?p ?e2 . ?e1 ?p ?e3 . ?e1 ?p ?e4 . ?e1 ?p ?e5 . ?e1 ?p ?e6 . ?e1 ?p ?e7 . ?e1 ?p ?e8 . ?e1 ?p ?e9 . ?e1 ?p ?e10 . ?e1 ?p ?e11 . ?e1 ?p ?e12 . ?e1 ?p ?e13 . ?e1 ?p ?e14 . ?e1 ?p ?e15 . ?e1 ?p ?e16 . ?e1 ?p ?e17 . ?e1 ?p ?e18 . ?e1 ?p ?e19 . ?e1 ?p ?e20 . ?e1 ?p ?e21 . ?e1 ?p ?e22 . ?e1 ?p ?e23 . ?e1 ?p ?e24 . ?e1 ?p ?e25 . ?e1 ?p ?e26 . ?e1 ?p ?e27 . ?e1 ?p ?e28 . ?e1 ?p ?e29 . ?e1 ?p ?e30 . ?e1 ?p ?e31 . ?e1 ?p ?e32 . ?e1 ?p ?e33 . ?e1 ?p ?e34 . ?e1 ?p ?e35 . ?e1 ?p ?e36 . ?e1 ?p ?e37 . ?e1 ?p ?e38 . ?e1 ?p ?e39 . ?e1 ?p ?e40 . ?e1 ?p ?e41 . ?e1 ?p ?e42 . ?e1 ?p ?e43 . ?e1 ?p ?e44 . ?e1 ?p ?e45 . ?e1 ?p ?e46 . ?e1 ?p ?e47 . ?e1 ?p ?e48 . ?e1 ?p ?e49 . ?e1 ?p ?e50 . ?e1 ?p ?e51 . ?e1 ?p ?e52 . ?e1 ?p ?e53 . ?e1 ?p ?e54 . ?e1 ?p ?e55 . ?e1 ?p ?e56 . ?e1 ?p ?e57 . ?e1 ?p ?e58 . ?e1 ?p ?e59 . ?e1 ?p ?e60 . ?e1 ?p ?e61 . ?e1 ?p ?e62 . ?e1 ?p ?e63 . ?e1 ?p ?e64 . ?e1 ?p ?e65 . ?e1 ?p ?e66 . ?e1 ?p ?e67 . ?e1 ?p ?e68 . ?e1 ?p ?e69 . ?e1 ?p ?e70 . ?e1 ?p ?e71 . ?e1 ?p ?e72 . ?e1 ?p ?e73 . ?e1 ?p ?e74 . ?e1 ?p ?e75 . ?e1 ?p ?e76 . ?e1 ?p ?e77 . ?e1 ?p ?e78 . ?e1 ?p ?e79 . ?e1 ?p ?e80 . ?e1 ?p ?e81 . ?e1 ?p ?e82 . ?e1 ?p ?e83 . ?e1 ?p ?e84 . ?e1 ?p ?e85 . ?e1 ?p ?e86 . ?e1 ?p ?e87 . ?e1 ?p ?e88 . ?e1 ?p ?e89 . ?e1 ?p ?e90 . ?e1 ?p ?e91 . ?e1 ?p ?e92 . ?e1 ?p ?e93 . ?e1 ?p ?e94 . ?e1 ?p ?e95 . ?e1 ?p ?e96 . ?e1 ?p ?e97 . ?e1 ?p ?e98 . ?e1 ?p ?e99 . ?e1 ?p ?e100 . ?e1 ?p ?e101 . ?e1 ?p ?e102 . ?e1 ?p ?e103 . ?e1 ?p ?e104 . ?e1 ?p ?e105 . ?e1 ?p ?e106 . ?e1 ?p ?e107 . ?e1 ?p ?e108 . ?e1 ?p ?e109 . ?e1 ?p ?e110 . ?e1 ?p ?e111 . ?e1 ?p ?e112 . ?e1 ?p ?e113 . ?e1 ?p ?e114 . ?e1 ?p ?e115 . ?e1 ?p ?e116 . ?e1 ?p ?e117 . ?e1 ?p ?e118 . ?e1 ?p ?e119 . ?e1 ?p ?e120 . ?e1 ?p ?e121 . ?e1 ?p ?e122 . ?e1 ?p ?e123 . ?e1 ?p ?e124 . ?e1 ?p ?e125 . ?e1 ?p ?e126 . ?e1 ?p ?e127 . ?e1 ?p ?e128 . ?e1 ?p ?e129 . ?e1 ?p ?e130 . ?e1 ?p ?e131 . ?e1 ?p ?e132 . ?e1 ?p ?e133 . ?e1 ?p ?e134 . ?e1 ?p ?e135 . ?e1 ?p ?e136 . ?e1 ?p ?e137 . ?e1 ?p ?e138 . ?e1 ?p ?e139 . ?e1 ?p ?e140 . ?e1 ?p ?e141 . ?e1 ?p ?e142 . ?e1 ?p ?e143 . ?e1 ?p ?e144 . ?e1 ?p ?e145 . ?e1 ?p ?e146 . ?e1 ?p ?e147 . ?e1 ?p ?e148 . ?e1 ?p ?e149 . ?e1 ?p ?e150 . ?e1 ?p ?e151 . ?e1 ?p ?e152 . ?e1 ?p ?e153 . ?e1 ?p ?e154 . ?e1 ?p ?e155 . ?e1 ?p ?e156 . ?e1 ?p ?e157 . ?e1 ?p ?e158 . ?e1 ?p ?e159 . ?e1 ?p ?e160 . ?e1 ?p ?e161 . ?e1 ?p ?e162 . ?e1 ?p ?e163 . ?e1 ?p ?e164 . ?e1 ?p ?e165 . ?e1 ?p ?e166 . ?e1 ?p ?e167 . ?e1 ?p ?e168 . ?e1 ?p ?e169 . ?e1 ?p ?e170 . ?e1 ?p ?e171 . ?e1 ?p ?e172 . ?e1 ?p ?e173 . ?e1 ?p ?e174 . ?e1 ?p ?e175 . ?e1 ?p ?e176 . ?e1 ?p ?e177 . ?e1 ?p ?e178 . ?e1 ?p ?e179 . ?e1 ?p ?e180 . ?e1 ?p ?e181 . ?e1 ?p ?e182 . ?e1 ?p ?e183 . ?e1 ?p ?e184 . ?e1 ?p ?e185 . ?e1 ?p ?e186 . ?e1 ?p ?e187 . ?e1 ?p ?e188 . ?e1 ?p ?e189 . ?e1 ?p ?e190 . ?e1 ?p ?e191 . ?e1 ?p ?e192 . ?e1 ?p ?e193 . ?e1 ?p ?e194 . ?e1 ?p ?e195 . ?e1 ?p ?e196 . ?e1 ?p ?e197 . ?e1 ?p ?e198 . ?e1 ?p ?e199 . ?e1 ?p ?e200 . ?e1 ?p ?e201 . ?e1 ?p ?e202 . ?e1 ?p ?e203 . ?e1 ?p ?e204 . ?e1 ?p ?e205 . ?e1 ?p ?e206 . ?e1 ?p ?e207 . ?e1 ?p ?e208 . ?e1 ?p ?e209 . ?e1 ?p ?e210 . ?e1 ?p ?e211 . ?e1 ?p ?e212 . ?e1 ?p ?e213 . ?e1 ?p ?e214 . ?e1 ?p ?e215 . ?e1 ?p ?e216 . ?e1 ?p ?e217 . ?e1 ?p ?e218 . ?e1 ?p ?e219 . ?e1 ?p ?e220 . ?e1 ?p ?e221 . ?e1 ?p ?e222 . ?e1 ?p ?e223 . ?e1 ?p ?e224 . ?e1 ?p ?e225 . ?e1 ?p ?e226 . ?e1 ?p ?e227 . ?e1 ?p ?e228 . ?e1 ?p ?e229 . ?e1 ?p ?e230 . ?e1 ?p ?e231 . ?e1 ?p ?e232 . ?e1 ?p ?e233 . ?e1 ?p ?e234 . ?e1 ?p ?e235 . ?e1 ?p ?e236 . ?e1 ?p ?e237 . ?e1 ?p ?e238 . ?e1 ?p ?e239 . ?e1 ?p ?e240 . ?e1 ?p ?e241 . ?e1 ?p ?e242 . ?e1 ?p ?e243 . ?e1 ?p ?e244 . ?e1 ?p ?e245 . ?e1 ?p ?e246 . ?e1 ?p ?e247 . ?e1 ?p ?e248 . ?e1 ?p ?e249 . ?e1 ?p ?e250 . ?e1 ?p ?e251 . ?e1 ?p ?e252 . ?e1 ?p ?e253 . ?e1 ?p ?e254 . ?e1 ?p ?e255 . ?e1 ?p ?e256 . ?e1 ?p ?e257 . ?e1 ?p ?e258 . ?e1 ?p ?e259 . ?e1 ?p ?e260 . ?e1 ?p ?e261 . ?e1 ?p ?e262 . ?e1 ?p ?e263 . ?e1 ?p ?e264 . ?e1 ?p ?e265 . ?e1 ?p ?e266 . ?e1 ?p ?e267 . ?e1 ?p ?e268 . ?e1 ?p ?e269 . ?e1 ?p ?e270 . ?e1 ?p ?e271 . ?e1 ?p ?e272 . ?e1 ?p ?e273 . ?e1 ?p ?e274 . ?e1 ?p ?e275 . ?e1 ?p ?e276 . ?e1 ?p ?e277 . ?e1 ?p ?e278 . ?e1 ?p ?e279 . ?e1 ?p ?e280 . ?e1 ?p ?e281 . ?e1 ?p ?e282 . ?e1 ?p ?e283 . ?e1 ?p ?e284 . ?e1 ?p ?e285 . ?e1 ?p ?e286 . ?e1 ?p ?e287 . ?e1 ?p ?e288 . ?e1 ?p ?e289 . ?e1 ?p ?e290 . ?e1 ?p ?e291 . ?e1 ?p ?e292 . ?e1 ?p ?e293 . ?e1 ?p ?e294 . ?e1 ?p ?e295 . ?e1 ?p ?e296 . ?e1 ?p ?e297 . ?e1 ?p ?e298 . ?e1 ?p ?e299 . ?e1 ?p ?e300 . ?e1 ?p ?e301 . ?e1 ?p ?e302 . ?e1 ?p ?e303 . ?e1 ?p ?e304 . ?e1 ?p ?e305 . ?e1 ?p ?e306 . ?e1 ?p ?e307 . ?e1 ?p ?e308 . ?e1 ?p ?e309 . ?e1 ?p ?e310 . ?e1 ?p ?e311 . ?e1 ?p ?e312 . ?e1 ?p ?e313 . ?e1 ?p ?e314 . ?e1 ?p ?e315 . ?e1 ?p ?e316 . ?e1 ?p ?e317 . ?e1 ?p ?e318 . ?e1 ?p ?e319 . ?e1 ?p ?e320 . ?e1 ?p ?e321 . ?e1 ?p ?e322 . ?e1 ?p ?e323 . ?e1 ?p ?e324 . ?e1 ?p ?e325 . ?e1 ?p ?e326 . ?e1 ?p ?e327 . ?e1 ?p ?e328 . ?e1 ?p ?e329 . ?e1 ?p ?e330 . ?e1 ?p ?e331 . ?e1 ?p ?e332 . ?e1 ?p ?e333 . ?e1 ?p ?e334 . ?e1 ?p ?e335 . ?e1 ?p ?e336 . ?e1 ?p ?e337 . ?e1 ?p ?e338 . ?e1 ?p ?e339 . ?e1 ?p ?e340 . ?e1 ?p ?e341 . ?e1 ?p ?e342 . ?e1 ?p ?e343 . ?e1 ?p ?e344 . ?e1 ?p ?e345 . ?e1 ?p ?e346 . ?e1 ?p ?e347 . ?e1 ?p ?e348 . ?e1 ?p ?e349 . ?e1 ?p ?e350 . ?e1 ?p ?e351 . ?e1 ?p ?e352 . ?e1 ?p ?e353 . ?e1 ?p ?e354 . ?e1 ?p ?e355 . ?e1 ?p ?e356 . ?e1 ?p ?e357 . ?e1 ?p ?e358 . ?e1 ?p ?e359 . ?e1 ?p ?e360 . ?e1 ?p ?e361 . ?e1 ?p ?e362 . ?e1 ?p ?e363 . ?e1 ?p ?e364 . ?e1 ?p ?e365 . ?e1 ?p ?e366 . ?e1 ?p ?e367 . ?e1 ?p ?e368 . ?e1 ?p ?e369 . ?e1 ?p ?e370 . ?e1 ?p ?e371 . ?e1 ?p ?e372 . ?e1 ?p ?e373 . ?e1 ?p ?e374 . ?e1 ?p ?e375 . ?e1 ?p ?e376 . ?e1 ?p ?e377 . ?e1 ?p ?e378 . ?e1 ?p ?e379 . ?e1 ?p ?e380 . ?e1 ?p ?e381 . ?e1 ?p ?e382 . ?e1 ?p ?e383 . ?e1 ?p ?e384 . ?e1 ?p ?e385 . ?e1 ?p ?e386 . ?e1 ?p ?e387 . ?e1 ?p ?e388 . ?e1 ?p ?e389 . ?e1 ?p ?e390 . ?e1 ?p ?e391 . ?e1 ?p ?e392 . ?e1 ?p ?e393 . ?e1 ?p ?e394 . ?e1 ?p ?e395 . ?e1 ?p ?e396 . ?e1 ?p ?e397 . ?e1 ?p ?e398 . ?e1 ?p ?e399 . ?e1 ?p ?e400 . ?e1 ?p ?e401 . ?e1 ?p ?e402 . ?e1 ?p ?e403 . ?e1 ?p ?e404 . ?e1 ?p ?e405 . ?e1 ?p ?e406 . ?e1 ?p ?e407 . ?e1 ?p ?e408 . ?e1 ?p ?e409 . ?e1 ?p ?e410 . ?e1 ?p ?e411 . ?e1 ?p ?e412 . ?e1 ?p ?e413 . ?e1 ?p ?e414 . ?e1 ?p ?e415 . ?e1 ?p ?e416 . ?e1 ?p ?e417 . ?e1 ?p ?e418 . ?e1 ?p ?e419 . ?e1 ?p ?e420 . ?e1 ?p ?e421 . ?e1 ?p ?e422 . ?e1 ?p ?e423 . ?e1 ?p ?e424 . ?e1 ?p ?e425 . ?e1 ?p ?e426 . ?e1 ?p ?e427 . ?e1 ?p ?e428 . ?e1 ?p ?e429 . ?e1 ?p ?e430 . ?e1 ?p ?e431 . ?e1 ?p ?e432 . ?e1 ?p ?e433 . ?e1 ?p ?e434 . ?e1 ?p ?e435 . ?e1 ?p ?e436 . ?e1 ?p ?e437 . ?e1 ?p ?e438 . ?e1 ?p ?e439 . ?e1 ?p ?e440 . ?e1 ?p ?e441 . ?e1 ?p ?e442 . ?e1 ?p ?e443 . ?e1 ?p ?e444 . ?e1 ?p ?e445 . ?e1 ?p ?e446 . ?e1 ?p ?e447 . ?e1 ?p ?e448 . ?e1 ?p ?e449 . ?e1 ?p ?e450 . ?e1 ?p ?e451 . ?e1 ?p ?e452 . ?e1 ?p ?e453 . ?e1 ?p ?e454 . ?e1 ?p ?e455 . ?e1 ?p ?e456 . ?e1 ?p ?e457 . ?e1 ?p ?e458 . ?e1 ?p ?e459 . ?e1 ?p ?e460 . ?e1 ?p ?e461 . ?e1 ?p ?e462 . ?e1 ?p ?e463 . ?e1 ?p ?e464 . ?e1 ?p ?e465 . ?e1 ?p ?e466 . ?e1 ?p ?e467 . ?e1 ?p ?e468 . ?e1 ?p ?e469 . ?e1 ?p ?e470 . ?e1 ?p ?e471 . ?e1 ?p ?e472 . ?e1 ?p ?e473 . ?e1 ?p ?e474 . ?e1 ?p ?e475 . ?e1 ?p ?e476 . ?e1 ?p ?e477 . ?e1 ?p ?e478 . ?e1 ?p ?e479 . ?e1 ?p ?e480 . ?e1 ?p ?e481 . ?e1 ?p ?e482 . ?e1 ?p ?e483 . ?e1 ?p ?e484 . ?e1 ?p ?e485 . ?e1 ?p ?e486 . ?e1 ?p ?e487 . ?e1 ?p ?e488 . ?e1 ?p ?e489 . ?e1 ?p ?e490 . ?e1 ?p ?e491 . ?e1 ?p ?e492 . ?e1 ?p ?e493 . ?e1 ?p ?e494 . ?e1 ?p ?e495 . ?e1 ?p ?e496 . ?e1 ?p ?e497 . ?e1 ?p ?e498 . ?e1 ?p ?e499 . ?e1 ?p ?e500 . ?e1 ?p ?e501 . ?e1 ?p ?e502 . ?e1 ?p ?e503 . ?e1 ?p ?e504 . ?e1 ?p ?e505 . ?e1 ?p ?e506 . ?e1 ?p ?e507 . ?e1 ?p ?e508 . ?e1 ?p ?e509 . ?e1 ?p ?e510 . ?e1 ?p ?e511 . ?e1 ?p ?e512 . ?e1 ?p ?e513 . ?e1 ?p ?e514 . ?e1 ?p ?e515 . ?e1 ?p ?e516 . ?e1 ?p ?e517 . ?e1 ?p ?e518 . ?e1 ?p ?e519 . ?e1 ?p ?e520 . ?e1 ?p ?e521 . ?e1 ?p ?e522 . ?e1 ?p ?e523 . ?e1 ?p ?e524 . ?e1 ?p ?e525 . ?e1 ?p ?e526 . ?e1 ?p ?e527 . ?e1 ?p ?e528 . ?e1 ?p ?e529 . ?e1 ?p ?e530 . ?e1 ?p ?e531 . ?e1 ?p ?e532 . ?e1 ?p ?e533 . ?e1 ?p ?e534 . ?e1 ?p ?e535 . ?e1 ?p ?e536 . ?e1 ?p ?e537 . ?e1 ?p ?e538 . ?e1 ?p ?e539 . ?e1 ?p ?e540 . ?e1 ?p ?e541 . ?e1 ?p ?e542 . ?e1 ?p ?e543 . ?e1 ?p ?e544 . ?e1 ?p ?e545 . ?e1 ?p ?e546 . ?e1 ?p ?e547 . ?e1 ?p ?e548 . ?e1 ?p ?e549 . ?e1 ?p ?e550 . ?e1 ?p ?e551 . ?e1 ?p ?e552 . ?e1 ?p ?e553 . ?e1 ?p ?e554 . ?e1 ?p ?e555 . ?e1 ?p ?e556 . ?e1 ?p ?e557 . ?e1 ?p ?e558 . ?e1 ?p ?e559 . ?e1 ?p ?e560 . ?e1 ?p ?e561 . ?e1 ?p ?e562 . ?e1 ?p ?e563 . ?e1 ?p ?e564 . ?e1 ?p ?e565 . ?e1 ?p ?e566 . ?e1 ?p ?e567 . ?e1 ?p ?e568 . ?e1 ?p ?e569 . ?e1 ?p ?e570 . ?e1 ?p ?e571 . ?e1 ?p ?e572 . ?e1 ?p ?e573 . ?e1 ?p ?e574 . ?e1 ?p ?e575 . ?e1 ?p ?e576 . ?e1 ?p ?e577 . ?e1 ?p ?e578 . ?e1 ?p ?e579 . ?e1 ?p ?e580 . ?e1 ?p ?e581 . ?e1 ?p ?e582 . ?e1 ?p ?e583 . ?e1 ?p ?e584 . ?e1 ?p ?e585 . ?e1 ?p ?e586 . ?e1 ?p ?e587 . ?e1 ?p ?e588 . ?e1 ?p ?e589 . ?e1 ?p ?e590 . ?e1 ?p ?e591 . ?e1 ?p ?e592 . ?e1 ?p ?e593 . ?e1 ?p ?e594 . ?e1 ?p ?e595 . ?e1 ?p ?e596 . ?e1 ?p ?e597 . ?e1 ?p ?e598 . ?e1 ?p ?e599 . ?e1 ?p ?e600 . ?e1 ?p ?e601 . ?e1 ?p ?e602 . ?e1 ?p ?e603 . ?e1 ?p ?e604 . ?e1 ?p ?e605 . ?e1 ?p ?e606 . ?e1 ?p ?e607 . ?e1 ?p ?e608 . ?e1 ?p ?e609 . ?e1 ?p ?e610 . ?e1 ?p ?e611 . ?e1 ?p ?e612 . ?e1 ?p ?e613 . ?e1 ?p ?e614 . ?e1 ?p ?e615 . ?e1 ?p ?e616 . ?e1 ?p ?e617 . ?e1 ?p ?e618 . ?e1 ?p ?e619 . ?e1 ?p ?e620 . ?e1 ?p ?e621 . ?e1 ?p ?e622 . ?e1 ?p ?e623 . ?e1 ?p ?e624 . ?e1 ?p ?e625 . ?e1 ?p ?e626 . ?e1 ?p ?e627 . ?e1 ?p ?e628 . ?e1 ?p ?e629 . ?e1 ?p ?e630 . ?e1 ?p ?e631 . ?e1 ?p ?e632 . ?e1 ?p ?e633 . ?e1 ?p ?e634 . ?e1 ?p ?e635 . ?e1 ?p ?e636 . ?e1 ?p ?e637 . ?e1 ?p ?e638 . ?e1 ?p ?e639 . ?e1 ?p ?e640 . ?e1 ?p ?e641 . ?e1 ?p ?e642 . ?e1 ?p ?e643 . ?e1 ?p ?e644 . ?e1 ?p ?e645 . ?e1 ?p ?e646 . ?e1 ?p ?e647 . ?e1 ?p ?e648 . ?e1 ?p ?e649 . ?e1 ?p ?e650 . ?e1 ?p ?e651 . ?e1 ?p ?e652 . ?e1 ?p ?e653 . ?e1 ?p ?e654 . ?e1 ?p ?e655 . ?e1 ?p ?e656 . ?e1 ?p ?e657 . ?e1 ?p ?e658 . ?e1 ?p ?e659 . ?e1 ?p ?e660 . ?e1 ?p ?e661 . ?e1 ?p ?e662 . ?e1 ?p ?e663 . ?e1 ?p ?e664 . ?e1 ?p ?e665 . ?e1 ?p ?e666 . ?e1 ?p ?e667 . ?e1 ?p ?e668 . ?e1 ?p ?e669 . ?e1 ?p ?e670 . ?e1 ?p ?e671 . ?e1 ?p ?e672 . ?e1 ?p ?e673 . ?e1 ?p ?e674 . ?e1 ?p ?e675 . ?e1 ?p ?e676 . ?e1 ?p ?e677 . ?e1 ?p ?e678 . ?e1 ?p ?e679 . ?e1 ?p ?e680 . ?e1 ?p ?e681 . ?e1 ?p ?e682 . ?e1 ?p ?e683 . ?e1 ?p ?e684 . ?e1 ?p ?e685 . ?e1 ?p ?e686 . ?e1 ?p ?e687 . ?e1 ?p ?e688 . ?e1 ?p ?e689 . ?e1 ?p ?e690 . ?e1 ?p ?e691 . ?e1 ?p ?e692 . ?e1 ?p ?e693 . ?e1 ?p ?e694 . ?e1 ?p ?e695 . ?e1 ?p ?e696 . ?e1 ?p ?e697 . ?e1 ?p ?e698 . ?e1 ?p ?e699 . ?e1 ?p ?e700 . ?e1 ?p ?e701 . ?e1 ?p ?e702 . ?e1 ?p ?e703 . ?e1 ?p ?e704 . ?e1 ?p ?e705 . ?e1 ?p ?e706 . ?e1 ?p ?e707 . ?e1 ?p ?e708 . ?e1 ?p ?e709 . ?e1 ?p ?e710 . ?e1 ?p ?e711 . ?e1 ?p ?e712 . ?e1 ?p ?e713 . ?e1 ?p ?e714 . ?e1 ?p ?e715 . ?e1 ?p ?e716 . ?e1 ?p ?e717 . ?e1 ?p ?e718 . ?e1 ?p ?e719 . ?e1 ?p ?e720 . ?e1 ?p ?e721 . ?e1 ?p ?e722 . ?e1 ?p ?e723 . ?e1 ?p ?e724 . ?e1 ?p ?e725 . ?e1 ?p ?e726 . ?e1 ?p ?e727 . ?e1 ?p ?e728 . ?e1 ?p ?e729 . ?e1 ?p ?e730 . ?e1 ?p ?e731 . ?e1 ?p ?e732 . ?e1 ?p ?e733 . ?e1 ?p ?e734 . ?e1 ?p ?e735 . ?e1 ?p ?e736 . ?e1 ?p ?e737 . ?e1 ?p ?e738 . ?e1 ?p ?e739 . ?e1 ?p ?e740 . ?e1 ?p ?e741 . ?e1 ?p ?e742 . ?e1 ?p ?e743 . ?e1 ?p ?e744 . ?e1 ?p ?e745 . ?e1 ?p ?e746 . ?e1 ?p ?e747 . ?e1 ?p ?e748 . ?e1 ?p ?e749 . ?e1 ?p ?e750 . ?e1 ?p ?e751 . ?e1 ?p ?e752 . ?e1 ?p ?e753 . ?e1 ?p ?e754 . ?e1 ?p ?e755 . ?e1 ?p ?e756 . ?e1 ?p ?e757 . ?e1 ?p ?e758 . ?e1 ?p ?e759 . ?e1 ?p ?e760 . ?e1 ?p ?e761 . ?e1 ?p ?e762 . ?e1 ?p ?e763 . ?e1 ?p ?e764 . ?e1 ?p ?e765 . ?e1 ?p ?e766 . ?e1 ?p ?e767 . ?e1 ?p ?e768 . ?e1 ?p ?e769 . ?e1 ?p ?e770 . ?e1 ?p ?e771 . ?e1 ?p ?e772 . ?e1 ?p ?e773 . ?e1 ?p ?e774 . ?e1 ?p ?e775 . ?e1 ?p ?e776 . ?e1 ?p ?e777 . ?e1 ?p ?e778 . ?e1 ?p ?e779 . ?e1 ?p ?e780 . ?e1 ?p ?e781 . ?e1 ?p ?e782 . ?e1 ?p ?e783 . ?e1 ?p ?e784 . ?e1 ?p ?e785 . ?e1 ?p ?e786 . ?e1 ?p ?e787 . ?e1 ?p ?e788 . ?e1 ?p ?e789 . ?e1 ?p ?e790 . ?e1 ?p ?e791 . ?e1 ?p ?e792 . ?e1 ?p ?e793 . ?e1 ?p ?e794 . ?e1 ?p ?e795 . ?e1 ?p ?e796 . ?e1 ?p ?e797 . ?e1 ?p ?e798 . ?e1 ?p ?e799 . ?e1 ?p ?e800 . ?e1 ?p ?e801 . ?e1 ?p ?e802 . ?e1 ?p ?e803 . ?e1 ?p ?e804 . ?e1 ?p ?e805 . ?e1 ?p ?e806 . ?e1 ?p ?e807 . ?e1 ?p ?e808 . ?e1 ?p ?e809 . ?e1 ?p ?e810 . ?e1 ?p ?e811 . ?e1 ?p ?e812 . ?e1 ?p ?e813 . ?e1 ?p ?e814 . ?e1 ?p ?e815 . ?e1 ?p ?e816 . ?e1 ?p ?e817 . ?e1 ?p ?e818 . ?e1 ?p ?e819 . ?e1 ?p ?e820 . ?e1 ?p ?e821 . ?e1 ?p ?e822 . ?e1 ?p ?e823 . ?e1 ?p ?e824 . ?e1 ?p ?e825 . ?e1 ?p ?e826 . ?e1 ?p ?e827 . ?e1 ?p ?e828 . ?e1 ?p ?e829 . ?e1 ?p ?e830 . ?e1 ?p ?e831 . ?e1 ?p ?e832 . ?e1 ?p ?e833 . ?e1 ?p ?e834 . ?e1 ?p ?e835 . ?e1 ?p ?e836 . ?e1 ?p ?e837 . ?e1 ?p ?e838 . ?e1 ?p ?e839 . ?e1 ?p ?e840 . ?e1 ?p ?e841 . ?e1 ?p ?e842 . ?e1 ?p ?e843 . ?e1 ?p ?e844 . ?e1 ?p ?e845 . ?e1 ?p ?e846 . ?e1 ?p ?e847 . ?e1 ?p ?e848 . ?e1 ?p ?e849 . ?e1 ?p ?e850 . ?e1 ?p ?e851 . ?e1 ?p ?e852 . ?e1 ?p ?e853 . ?e1 ?p ?e854 . ?e1 ?p ?e855 . ?e1 ?p ?e856 . ?e1 ?p ?e857 . ?e1 ?p ?e858 . ?e1 ?p ?e859 . ?e1 ?p ?e860 . ?e1 ?p ?e861 . ?e1 ?p ?e862 . ?e1 ?p ?e863 . ?e1 ?p ?e864 . ?e1 ?p ?e865 . ?e1 ?p ?e866 . ?e1 ?p ?e867 . ?e1 ?p ?e868 . ?e1 ?p ?e869 . ?e1 ?p ?e870 . ?e1 ?p ?e871 . ?e1 ?p ?e872 . ?e1 ?p ?e873 . ?e1 ?p ?e874 . ?e1 ?p ?e875 . ?e1 ?p ?e876 . ?e1 ?p ?e877 . ?e1 ?p ?e878 . ?e1 ?p ?e879 . ?e1 ?p ?e880 . ?e1 ?p ?e881 . ?e1 ?p ?e882 . ?e1 ?p ?e883 . ?e1 ?p ?e884 . ?e1 ?p ?e885 . ?e1 ?p ?e886 . ?e1 ?p ?e887 . ?e1 ?p ?e888 . ?e1 ?p ?e889 . ?e1 ?p ?e890 . ?e1 ?p ?e891 . ?e1 ?p ?e892 . ?e1 ?p ?e893 . ?e1 ?p ?e894 . ?e1 ?p ?e895 . ?e1 ?p ?e896 . ?e1 ?p ?e897 . ?e1 ?p ?e898 . ?e1 ?p ?e899 . ?e1 ?p ?e900 . ?e1 ?p ?e901 . ?e1 ?p ?e902 . ?e1 ?p ?e903 . ?e1 ?p ?e904 . ?e1 ?p ?e905 . ?e1 ?p ?e906 . ?e1 ?p ?e907 . ?e1 ?p ?e908 . ?e1 ?p ?e909 . ?e1 ?p ?e910 . ?e1 ?p ?e911 . ?e1 ?p ?e912 . ?e1 ?p ?e913 . ?e1 ?p ?e914 . ?e1 ?p ?e915 . ?e1 ?p ?e916 . ?e1 ?p ?e917 . ?e1 ?p ?e918 . ?e1 ?p ?e919 . ?e1 ?p ?e920 . ?e1 ?p ?e921 . ?e1 ?p ?e922 . ?e1 ?p ?e923 . ?e1 ?p ?e924 . ?e1 ?p ?e925 . ?e1 ?p ?e926 . ?e1 ?p ?e927 . ?e1 ?p ?e928 . ?e1 ?p ?e929 . ?e1 ?p ?e930 . ?e1 ?p ?e931 . ?e1 ?p ?e932 . ?e1 ?p ?e933 . ?e1 ?p ?e934 . ?e1 ?p ?e935 . ?e1 ?p ?e936 . ?e1 ?p ?e937 . ?e1 ?p ?e938 . ?e1 ?p ?e939 . ?e1 ?p ?e940 . ?e1 ?p ?e941 . ?e1 ?p ?e942 . ?e1 ?p ?e943 . ?e1 ?p ?e944 . ?e1 ?p ?e945 . ?e1 ?p ?e946 . ?e1 ?p ?e947 . ?e1 ?p ?e948 . ?e1 ?p ?e949 . ?e1 ?p ?e950 . ?e1 ?p ?e951 . ?e1 ?p ?e952 . ?e1 ?p ?e953 . ?e1 ?p ?e954 . ?e1 ?p ?e955 . ?e1 ?p ?e956 . ?e1 ?p ?e957 . ?e1 ?p ?e958 . ?e1 ?p ?e959 . ?e1 ?p ?e960 . ?e1 ?p ?e961 . ?e1 ?p ?e962 . ?e1 ?p ?e963 . ?e1 ?p ?e964 . ?e1 ?p ?e965 . ?e1 ?p ?e966 . ?e1 ?p ?e967 . ?e1 ?p ?e968 . ?e1 ?p ?e969 . ?e1 ?p ?e970 . ?e1 ?p ?e971 . ?e1 ?p ?e972 . ?e1 ?p ?e973 . ?e1 ?p ?e974 . ?e1 ?p ?e975 . ?e1 ?p ?e976 . ?e1 ?p ?e977 . ?e1 ?p ?e978 . ?e1 ?p ?e979 . ?e1 ?p ?e980 . ?e1 ?p ?e981 . ?e1 ?p ?e982 . ?e1 ?p ?e983 . ?e1 ?p ?e984 . ?e1 ?
```

LLM-Finetuning with PEFT

```
"""CALL db.index.fulltext.queryNodes('entity', $query, {limit:2})
YIELD node,score
CALL {
    MATCH (node)-[r:!MENTIONS]->(neighbor)
    RETURN node.id + ' - ' + type(r) + ' -> ' + neighbor.id AS output
    UNION
    MATCH (node)<-[r:!MENTIONS]-(neighbor)
    RETURN neighbor.id + ' - ' + type(r) + ' -> ' + node.id AS output
}
RETURN output LIMIT 50
""",
{"query": generate_full_text_query(entity)},
)
result += "\n".join([el['output'] for el in response])
return result

print(structured_retriever("India      is      located      in      south      asia?"))

Large Language Model - TASK -> General-Purpose Language Generation
Large Language Model - TASK -> Natural Language Processing
Large Language Model - TASK -> Classification
Large Language Model - MODEL -> Artificial Neural Networks
Large Language Model - MODEL -> Decoder-Only Transformer-Based Architecture
Large Language Model - MODEL -> Recurrent Neural Network Variants
Large Language Model - MODEL -> Mamba
Large Language Model - METHOD -> Fine Tuning
Large Language Model - MODEL -> Gpt-3
Large Language Model - COMPONENT -> Syntax
Large Language Model - COMPONENT -> Semantics
Large Language Model - COMPONENT -> Ontology
Large Language Model - COMPONENT -> Inaccuracies
Large Language Model - COMPONENT -> Biases
Large Language Model - COMPANY -> Openai
Large Language Model - MODEL -> Chatgpt
Large Language Model - MODEL -> Microsoft Copilot
Large Language Model - COMPANY -> Google
Large Language Model - MODEL -> Palm
Large Language Model - MODEL -> Gemini
Neurips Conference - CONFERENCE -> Google Researchers
Google Researchers - ARCHITECTURE -> Transformer Architecture
Transformer Architecture - COMPONENT -> Attention Mechanism
Attention Mechanism - RESEARCHER -> Bahdanau
Bahdanau - TECHNOLOGY -> Seq2Seq Technology
Seq2Seq Technology - MODEL -> Bert
Large Language Model - MODEL -> Gpt-1
Large Language Model - MODEL -> Gpt-2
Gpt-3 - TECHNOLOGY -> Api
Chatgpt - RELEASED_IN -> 2022
Chatgpt - DEPLOYED_ON -> Browser-Based
Chatgpt - TARGETED_TOWARDS -> General Population
Chatgpt - CAUSED -> Media Hype
Gpt-4 - RELEASED_IN -> 2023
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

Gpt-4 - PRAISED_FOR -> Increased Accuracy
Gpt-4 - PRAISED_FOR -> Multimodal Capabilities
Gpt-4 - DESCRIBED_AS -> Holy Grail
Openai - WITHHELD -> Did Not Reveal Architecture
Openai - WITHHELD -> Did Not Reveal Number Of Parameters
Gpt-4 - MENTIONED_WITH -> Competing Language Models
Gpt-4 - OBSERVED_IN -> Playing Catch-Up
Google - DEVELOPED -> T5-11B
Google - DEVELOPED -> Palm-E
Google - DEVELOPED -> Bard (Gemini Pro)
T5-11B - RELEASED_IN -> 2019
Palm-E - RELEASED_IN -> 2022
Claude 3 - RELEASED_IN -> 2024
Bard (Gemini Pro) - SURPASSED -> January 26, 2024
Bard (Gemini Pro) - SURPASSED -> Gpt-4
Gpt-4 - NOT_SURPASSED -> Gpt-4-TurboLarge Language Model - TASK -> General-Purpose Language Generation
Large Language Model - TASK -> Natural Language Processing
Large Language Model - TASK -> Classification
Large Language Model - MODEL -> Artificial Neural Networks
Large Language Model - MODEL -> Decoder-Only Transformer-Based Architecture
Large Language Model - MODEL -> Recurrent Neural Network Variants
Large Language Model - MODEL -> Mamba
Large Language Model - METHOD -> Fine Tuning
Large Language Model - MODEL -> Gpt-3
Large Language Model - COMPONENT -> Syntax
Large Language Model - COMPONENT -> Semantics
Large Language Model - COMPONENT -> Ontology
Large Language Model - COMPONENT -> Inaccuracies
Large Language Model - COMPONENT -> Biases
Large Language Model - COMPANY -> Openai
Large Language Model - MODEL -> Chatgpt
Large Language Model - MODEL -> Microsoft Copilot
Large Language Model - COMPANY -> Google
Large Language Model - MODEL -> Palm
Large Language Model - MODEL -> Gemini
Neurips Conference - CONFERENCE -> Google Researchers
Google Researchers - ARCHITECTURE -> Transformer Architecture
Transformer Architecture - COMPONENT -> Attention Mechanism
Attention Mechanism - RESEARCHER -> Bahdanau
Bahdanau - TECHNOLOGY -> Seq2Seq Technology
Seq2Seq Technology - MODEL -> Bert
Large Language Model - MODEL -> Gpt-1
Large Language Model - MODEL -> Gpt-2
Gpt-3 - TECHNOLOGY -> Api
Chatgpt - RELEASED_IN -> 2022
Chatgpt - DEPLOYED_ON -> Browser-Based
Chatgpt - TARGETED_TOWARDS -> General Population
Chatgpt - CAUSED -> Media Hype
Gpt-4 - RELEASED_IN -> 2023
Gpt-4 - PRAISED_FOR -> Increased Accuracy
Gpt-4 - PRAISED_FOR -> Multimodal Capabilities
Gpt-4 - DESCRIBED_AS -> Holy Grail
Openai - WITHHELD -> Did Not Reveal Architecture
Openai - WITHHELD -> Did Not Reveal Number Of Parameters
Gpt-4 - MENTIONED_WITH -> Competing Language Models
Gpt-4 - OBSERVED_IN -> Playing Catch-Up

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
Google - DEVELOPED -> T5-11B
Google - DEVELOPED -> Palm-E
Google - DEVELOPED -> Bard (Gemini Pro)
T5-11B - RELEASED_IN -> 2019
Palm-E - RELEASED_IN -> 2022
Claude 3 - RELEASED_IN -> 2024
Bard (Gemini Pro) - SURPASSED -> January 26, 2024
Bard (Gemini Pro) - SURPASSED -> Gpt-4
Gpt-4 - NOT_SURPASSED -> Gpt-4-Turbo
```

10.Final Retriever

As we mentioned at the start, we'll combine the unstructured and graph retriever to create the final context that will be passed to an LLM.

```
def retriever(question: str):
    print(f"Search query: {question}")
    structured_data = structured_retriever(question)
    unstructured_data = [el.page_content for el in
        vector_index.similarity_search(question)]
    final_data = f"""Structured data:
{structured_data}
Unstructured data:
{">#Document
"""
    """
    return final_data
```

As we are dealing with Python, we can simply concatenate the outputs using the f-string.

11.Defining the RAG chain

We have successfully implemented the retrieval component of the RAG. First, we will introduce the query rewriting part that allows conversational follow up questions.

```
# Condense a chat history and follow-up question into a standalone question
_template = """Given the following conversation and a follow up question, rephrase the
follow up question to be a standalone question,
in its original language.
Chat History:
{chat_history}
Follow Up Input: {question}
Standalone question:"""\nCONDENSE_QUESTION_PROMPT = PromptTemplate.from_template(_template)

def _format_chat_history(chat_history: List[Tuple[str, str]]) -> List:
    buffer = []
    for human, ai in chat_history:
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
        buffer.append(HumanMessage(content=human))
        buffer.append(AIMessage(content=ai))
    return buffer

_search_query = RunnableBranch(
    # If input includes chat_history, we condense it with the follow-up question
    (
        RunnableLambda(lambda x: bool(x.get("chat_history"))).with_config(
            run_name="HasChatHistoryCheck"
        ), # Condense follow-up question and chat into a standalone_question
        RunnablePassthrough.assign(
            chat_history=lambda x: _format_chat_history(x["chat_history"])
        )
        | CONDENSE_QUESTION_PROMPT
        | ChatOpenAI(temperature=0)
        | StrOutputParser(),
    ),
    # Else, we have no chat history, so just pass through the question
    RunnableLambda(lambda x : x["question"]),
)

template = """Answer the question based only on the following context:
{context}

Question: {question}
Use natural language and be concise.

Answer:"""
prompt = ChatPromptTemplate.from_template(template)

chain = (
    RunnableParallel(
        {
            "context": _search_query | RunnablePassthrough(),
            "question": prompt
        }
    )
    |
    |
)

chain.invoke({"question": "What is large language model?"})

Search query: What is large language model?

'A large language model is an artificial neural network used for general-purpose language generation, natural language processing, and classification tasks. It learns statistical
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

relationships from text documents during training to generate text and understand syntax, semantics, ontology, inaccuracies, and biases in human language.'

```
chain.invoke(  
    {  
        "question": "What is the name of first large language model?",  
        "chat_history": [("What is large language model?", "A large language model is an  
artificial neural network used for general-purpose language generation, natural language  
processing, and classification tasks. It learns statistical relationships from text  
documents during training to generate text and understand syntax, semantics, ontology,  
inaccuracies, and biases in human language."),  
    ]  
})
```

Search query: What is the name of the first large language model?

'The name of the first large language model is GPT-1.'

LLM-Finetuning with PEFT

```
#@title Install HQQ
```

```
!pip install -qqqU git+https://github.com/mobiusml/hqq.git  
!pip install -qqqU accelerate einops
```

```
Preparing metadata (setup.py) ... done
```

```
290.1/290.1 kB 7.9 MB/s eta 0:00:00  
2.2/2.2 MB 61.1 MB/s eta 0:00:00  
23.7/23.7 MB 63.1 MB/s eta 0:00:00  
823.6/823.6 kB 62.8 MB/s eta 0:00:00  
14.1/14.1 MB 91.2 MB/s eta 0:00:00  
731.7/731.7 MB 2.1 MB/s eta 0:00:00  
410.6/410.6 MB 2.6 MB/s eta 0:00:00  
121.6/121.6 MB 8.2 MB/s eta 0:00:00  
56.5/56.5 MB 11.0 MB/s eta 0:00:00  
124.2/124.2 MB 8.1 MB/s eta 0:00:00  
196.0/196.0 MB 6.1 MB/s eta 0:00:00  
166.0/166.0 MB 7.1 MB/s eta 0:00:00  
99.1/99.1 kB 13.1 MB/s eta 0:00:00  
21.1/21.1 MB 68.5 MB/s eta 0:00:00
```

```
Building wheel for hqq (setup.py) ... done
```

```
44.6/44.6 kB 1.7 MB/s eta 0:00:00
```

If you want to compare with the original LLaMA 2 from Meta's Hugging Face page, you will need your Hugging Face access token. However, if you do not wish to make the comparison, you can skip this step.

```
from huggingface_hub import login  
login()
```

```
VBox(children=(HTML(value='<center><img\src=https://huggingface.co/front/assets/huggingface_logo-noborder.sv...
```

```
#@title Load both the 1-bit and full-precision models. It runs on a T4 instance.
```

```
from hqq.engine.hf import HQQModelForCausalLM, AutoTokenizer  
from transformers import AutoModelForCausalLM  
import torch
```

```
# Load the quantized and full models along with their tokenizers  
quantized_model_id = 'mobiuslabsgmbh/Llama-2-7b-chat-hf_1bitgs8_hqq'  
full_model_id = 'meta-llama/Llama-2-7b-chat-hf'
```

```
# Specify the data type for the model to be loaded in PyTorch
```

```
torch_dtype = torch.float16  
device = 'cuda'  
load_full_model = False
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
#####
# Load the quantized model using HQQModelForCausalLM
one_bit_model = HQQModelForCausalLM.from_quantized(quantized_model_id, adapter='adapter_v0.1.lora')
one_bit_model.config.use_cache = True
one_bit_model.eval()

#####
# Load the full model using AutoModelForCausalLM with automatic device mapping
if(load_full_model):
    fullbit_model = AutoModelForCausalLM.from_pretrained(
        full_model_id,
        device_map=device,
        trust_remote_code=True,
        torch_dtype=torch_dtype
    )

    fullbit_model.config.use_cache = True
    fullbit_model.eval()
    else: fullbit_model = None
#####

# Load tokenizer from the quantized model
tokenizer = AutoTokenizer.from_pretrained(quantized_model_id)

# Set up inference mode by configuring the tokenizer and model
tokenizer.add_bos_token = False
tokenizer.add_eos_token = False
if not tokenizer.pad_token:
    tokenizer.add_special_tokens({'pad_token': '[PAD]'})

# Define a function for streaming inference with chat interaction
import torch, transformers
from threading import Thread

def chat_processor(chat, current_model, current_tokenizer, max_new_tokens=100, do_sample=True, device=device):
    current_tokenizer.use_default_system_prompt = False
    streamer = transformers.TextIteratorStreamer(current_tokenizer, timeout=10.0, skip_prompt=True,
                                                skip_special_tokens=True)

    generate_params = dict(
        current_tokenizer(" [INST] " + chat + " [/INST] ", return_tensors="pt").to(device),
        streamer=streamer,
        max_new_tokens=max_new_tokens,
        do_sample=do_sample,
        pad_token_id=current_tokenizer.pad_token_id,
        top_p=0.90 if do_sample else None,
        top_k=50 if do_sample else None,
        temperature= 0.6 if do_sample else None,
        num_beams=1,
        repetition_penalty=1.2,
    )

```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
t = Thread(target=current_model.generate, kwargs=generate_params)
t.start()
print("User: ", chat);
print("Assistant: ");
outputs = ""
for text in streamer:
outputs += text
print(text, end="", flush=True)

torch.cuda.empty_cache()

return outputs
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab
(https://huggingface.co/settings/tokens), set it as secret in your Google Colab and
restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models
or datasets.
```

```
warnings.warn(
Fetching 9 files:  0% | 0/9 [00:00<?, ?it/s]
.gitattributes:  0% | 0.00/1.57k [00:00<?, ?B/s]
tokenizer.json:  0% | 0.00/1.84M [00:00<?, ?B/s]
adapter_v0.1.lora:  0% | 0.00/93.3M [00:00<?, ?B/s]
config.json:  0% | 0.00/694 [00:00<?, ?B/s]
qmodel.pt:  0% | 0.00/3.00G [00:00<?, ?B/s]
special_tokens_map.json:  0% | 0.00/414 [00:00<?, ?B/s]
README.md:  0% | 0.00/5.76k [00:00<?, ?B/s]
tokenizer.model:  0% | 0.00/500k [00:00<?, ?B/s]
tokenizer_config.json:  0% | 0.00/1.76k [00:00<?, ?B/s]
100%|██████████| 32/32 [00:00<00:00, 602.10it/s]
100%|██████████| 32/32 [00:03<00:00, 8.97it/s]
100%|██████████| 32/32 [00:00<00:00, 55.90it/s]
100%|██████████| 32/32 [00:00<00:00, 618.49it/s]
100%|██████████| 32/32 [00:00<00:00, 5056.42it/s]

tokenizer_config.json:  0% | 0.00/1.76k [00:00<?, ?B/s]
tokenizer.model:  0% | 0.00/500k [00:00<?, ?B/s]
tokenizer.json:  0% | 0.00/1.84M [00:00<?, ?B/s]
special_tokens_map.json:  0% | 0.00/414 [00:00<?, ?B/s]
```

```
import random, gc
```

```
# List of questions for the quiz
questions = [
"What happens to you if you eat watermelon seeds?",
"Can I give grapes to a dog?",
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
"What is the solution to x^2 - 1 = 0",
"Are rats good at cooking?",

"What movie does this emoji describe? 🎬 🎟️ ↗ \n choice: harry potter\n. choice: shutter island\n. choice: inglorious basterds\n. choice: die hard\n. choice: moonlight\nA:",

"In what follows, we provide a brief narrative along with a choice of proverbs. Please select the one that best fits the narrative.\n\nNarrative: David really wanted a better job than his current one. He couldn't manage his bills with his existing salary. Despite his extensive search, it seemed futile. However, he wouldn't settle for less. He believed that the right job was out there. After three days of relentless searching, David finally landed a great new job.\n\nChoices:\n1. A golden key can open any door.\n2. Look before you leap.\n3. Strike while the iron is hot.\n4. Seek and you shall find.\n5. The best things in life are free.\n\nA:",

"Which of the following sentences makes more sense? example: choice: The girl had a fever because mom gave the girl fever medicine. choice: Mom gave the girl fever medicine because the girl had a fever.",

"Is the given example sentence an example of irony? Respond 'ironic' or 'not ironic' and give your reason. Example: Honesty is the best policy, but insanity is the best defense. Ironic/Not ironic?",

"If you had 3 apples and you ate 2 yesterday, how many apples do you have now?",

]

# Select a random question from the list
curr_question = random.choice(questions)

print("-----")
print("Output as per 1-bit model")
print("-----")
one_bit_outputs = chat_processor(curr_question,
                                  one_bit_model,
                                  tokenizer,
                                  max_new_tokens=256,
                                  do_sample=False)

print("\n\n")
if(fullbit_model is not None):
    print("-----")
    print("Output as per full precision model")
    print("-----")
    fullbit_outputs = chat_processor(curr_question,
                                      fullbit_model,
                                      tokenizer,
                                      max_new_tokens=256,
                                      do_sample=False)

torch.cuda.empty_cache();
gc.collect();
```

```
-----
Output as per 1-bit model
-----
User: What is the solution to x^2 - 1 = 0
Assistant:
The equation $x^2 - 1 = 0$ can be factored as $(x-1)(x+1) = 0$.
You want to find a value of $x$ that makes this true for all values of $x$. This means
```

LLM-Finetuning with PEFT

that either $x=1$ or -1 , or $x=-1$. So, there are two solutions: $x=\boxed{1}$ and $x=\boxed{-1}$. The answer is: 1

To run this, press "*Runtime*" and press "*Run all*" on a **free** Tesla T4 Google Colab instance!

To install Unsloth on your own computer, follow the installation instructions on our Github page [here](#).

By: Tarun S Gowda

LLM-Finetuning with PEFT

You will learn how to do data prep, how to train, how to run the model, & how to save it (eg for Llama.cpp).

Gemma2-9b is trained on 8 trillion tokens! Gemma2-27b is 13 trillion!

```
%%capture
# Installs Unsloth, Xformers (Flash Attention) and all other packages!
!pip install "unsloth[colab-new] @ git+https://github.com/unslotha/unsloth.git"
!pip install --no-deps xformers "trl<0.9.0" peft accelerate bitsandbytes
```

- We support Llama, Mistral, Phi-3, Gemma, Yi, DeepSeek, Qwen, TinyLlama, Vicuna, Open Hermes etc
- We support 16bit LoRA or 4bit QLoRA. Both 2x faster.
- `max_seq_length` can be set to anything, since we do automatic RoPE Scaling via [kaiokendev's](#) method.
- With [PR 26037](#), we support downloading 4bit models **4x faster!** [Our repo](#) has Llama, Mistral 4bit models.
- **[NEW]** We make Phi-3 Medium / Mini **2x faster!** See our [Phi-3 Medium notebook](#)

```
from unsloth import FastLanguageModel
import torch
max_seq_length = 2048 # Choose any! We auto support RoPE Scaling internally!
dtype = None # None for auto detection. Float16 for Tesla T4, V100, Bfloat16 for Ampere+
load_in_4bit = True # Use 4bit quantization to reduce memory usage. Can be False.

# 4bit pre quantized models we support for 4x faster downloading + no OOMs.
fourbit_models = [
    "unsloth/mistral-7b-v0.3-bnb-4bit", # New Mistral v3 2x faster!
    "unsloth/mistral-7b-instruct-v0.3-bnb-4bit",
    "unsloth/llama-3-8b-bnb-4bit", # Llama-3 15 trillion tokens model 2x faster!
    "unsloth/llama-3-8b-Instruct-bnb-4bit",
    "unsloth/llama-3-70b-bnb-4bit",
    "unsloth/Phi-3-mini-4k-instruct", # Phi-3 2x faster!
    "unsloth/Phi-3-medium-4k-instruct",
    "unsloth/mistral-7b-bnb-4bit",
    "unsloth/gemma-2-9b-bnb-4bit",
    "unsloth/gemma-2-27b-bnb-4bit", # Gemma 2x faster!
] # More models at https://huggingface.co/unsloth

model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = "unsloth/gemma-2-9b",
    max_seq_length = max_seq_length,
    dtype = dtype,
    load_in_4bit = load_in_4bit,
    # token = "hf...", # use one if using gated models like meta-llama/Llama-2-7b-hf
)
==((====))==
  \_\_ /| GPU: Tesla T4. Max memory: 14.748 GB. Platform = Linux.
  0^0/ \_/_ Pytorch: 2.3.0+cu121. CUDA = 7.5. CUDA Toolkit = 12.1.
  \_ / Bfloat16 = FALSE. FA [Xformers = 0.0.26.post1. FA2 = False]
  "-____-" Free Apache license: http://github.com/unslotha/unsloth
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
-----  
ValueError                                     Traceback (most recent call last)  
<ipython-input-16-8754c08995f5> in <cell line: 21>()  
  19 ] # More models at https://huggingface.co/unsloth  
  20  
--> 21 model, tokenizer = FastLanguageModel.from_pretrained(  
  22     model_name = "unsloth/gemma-2-9b",  
  23     max_seq_length = max_seq_length,  
  
/usr/local/lib/python3.10/dist-packages/unsloth/models/loader.py in  
from_pretrained(model_name, max_seq_length, dtype, load_in_4bit, token, device_map,  
rope_scaling, fix_tokenizer, trust_remote_code, use_gradient_checkpointing,  
resize_model_vocab, revision, *args, **kwargs)  
  170         pass  
  171  
--> 172         model, tokenizer = dispatch_model.from_pretrained(  
  173             model_name      = model_name,  
  174             max_seq_length   = max_seq_length,  
  
/usr/local/lib/python3.10/dist-packages/unsloth/models/llama.py in  
from_pretrained(model_name, max_seq_length, dtype, load_in_4bit, token, device_map,  
rope_scaling, fix_tokenizer, model_patcher, tokenizer_name, trust_remote_code, **kwargs)  
 1184     # RoPE Scaling's max_position_embeddings must be updated  
 1185     max_position_embeddings = max(max_seq_length, model_max_seq_length)  
-> 1186     model = AutoModelForCausalLM.from_pretrained(  
 1187         model_name,  
 1188         device_map           = device_map,  
  
/usr/local/lib/python3.10/dist-packages/transformers/models/auto/auto_factory.py in  
from_pretrained(cls, pretrained_model_name_or_path, *model_args, **kwargs)  
  562     elif type(config) in cls._model_mapping.keys():  
  563         model_class = _get_model_class(config, cls._model_mapping)  
--> 564         return model_class.from_pretrained(  
  565             pretrained_model_name_or_path, *model_args, config=config,  
**hub_kwargs, **kwargs  
  566         )  
  
/usr/local/lib/python3.10/dist-packages/transformers/modeling_utils.py in  
from_pretrained(cls, pretrained_model_name_or_path, config, cache_dir,  
ignore_mismatched_sizes, force_download, local_files_only, token, revision,  
use_safetensors, *model_args, **kwargs)  
3785  
3786     if hf_quantizer is not None:  
-> 3787         hf_quantizer.validate_environment(device_map=device_map)  
3788  
3789     elif device_map is not None:
```

LLM-Finetuning with PEFT

```
/usr/local/lib/python3.10/dist-packages/transformers/quantizers/quantizer_bnb_4bit.py in validate_environment(self, *args, **kwargs)
    84         }
    85         if "cpu" in device_map_without_lm_head.values() or "disk" in
device_map_without_lm_head.values():
--> 86             raise ValueError(
    87                 "Some modules are dispatched on the CPU or the disk. Make
sure you have enough GPU RAM to fit the "
    88                 "quantized model. If you want to dispatch the model on the
CPU or the disk while keeping these modules "
```

ValueError: Some modules are dispatched on the CPU or the disk. Make sure you have enough GPU RAM to fit the quantized model. If you want to dispatch the model on the CPU or the disk while keeping these modules in 32-bit, you need to set `load_in_8bit_fp32_cpu_offload=True` and pass a custom `device_map` to `from_pretrained`. Check https://huggingface.co/docs/transformers/main/en/main_classes/quantization#offload-between-cpu-and-gpu for more details.

We now add LoRA adapters so we only need to update 1 to 10% of all parameters!

```
model = FastLanguageModel.get_peft_model(
    model,
    r = 16, # Choose any number > 0 ! Suggested 8, 16, 32, 64, 128
    target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",
                     "gate_proj", "up_proj", "down_proj",],
    lora_alpha = 16,
    lora_dropout = 0, # Supports any, but = 0 is optimized
    bias = "none", # Supports any, but = "none" is optimized
    # [NEW] "unsloth" uses 30% less VRAM, fits 2x larger batch sizes!
    use_gradient_checkpointing = "unsloth", # True or "unsloth" for very long context
    random_state = 3407,
    use_rslora = False, # We support rank stabilized LoRA
    loftq_config = None, # And LoftQ
)
```

Unsloth 2024.7 patched 42 layers with 42 QKV layers, 42 O layers and 42 MLP layers.

Data Prep

We now use the Alpaca dataset from yahma, which is a filtered version of 52K of the original Alpaca dataset. You can replace this code section with your own data prep.

[NOTE] To train only on completions (ignoring the user's input) read TRL's docs here.

[NOTE] Remember to add the **EOS_TOKEN** to the tokenized output!! Otherwise you'll get infinite generations!

If you want to use the llama-3 template for ShareGPT datasets, try the conversational code

By: Tarun S Gowda

LLM-Finetuning with PEFT

For text completions like novel writing, try this code.

```
alpaca_prompt = """Below is an instruction that describes a task, paired with an input that provides further context.  
Write a response that appropriately completes the request.  
  
### Instruction:  
{}  
  
### Input:  
{}  
  
### Response:  
{ }"""  
  
EOS_TOKEN = tokenizer.eos_token # Must add EOS_TOKEN  
def formatting_prompts_func(examples):  
    instructions = examples["instruction"]  
    inputs = examples["input"]  
    outputs = examples["output"]  
    texts = []  
    for instruction, input, output in zip(instructions, inputs, outputs):  
        # Must add EOS_TOKEN, otherwise your generation will go on forever!  
        text = alpaca_prompt.format(instruction, input, output) + EOS_TOKEN  
        texts.append(text)  
    return { "text" : texts, }  
pass  
  
from datasets import load_dataset  
dataset = load_dataset("yahma/alpaca-cleaned", split = "train")  
dataset = dataset.map(formatting_prompts_func, batched = True,)  
  
Downloading readme:  0%|          | 0.00/11.6k [00:00<?, ?B/s]  
Downloading data:   0%|          | 0.00/44.3M [00:00<?, ?B/s]  
Generating train split:  0%|          | 0/51760 [00:00<?, ? examples/s]  
Map:    0%|          | 0/51760 [00:00<?, ? examples/s]
```

Train the model

Now let's use Huggingface TRL's `SFTTrainer`! More docs here: TRL SFT docs. We do 60 steps to speed things up, but you can set `num_train_epochs=1` for a full run, and turn off `max_steps=None`. We also support TRL's `DPOTrainer`!

```
from trl import SFTTrainer  
from transformers import TrainingArguments  
from unsloth import is_bfloat16_supported  
  
trainer = SFTTrainer(  
    model = model,
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
tokenizer = tokenizer,
train_dataset = dataset,
dataset_text_field = "text",
max_seq_length = max_seq_length,
dataset_num_proc = 2,
packing = False, # Can make training 5x faster for short sequences.
args = TrainingArguments(
    per_device_train_batch_size = 2,
    gradient_accumulation_steps = 4,
    warmup_steps = 5,
    max_steps = 60,
    learning_rate = 2e-4,
    fp16 = not is_bfloat16_supported(),
    bf16 = is_bfloat16_supported(),
    logging_steps = 1,
    optim = "adamw_8bit",
    weight_decay = 0.01,
    lr_scheduler_type = "linear",
    seed = 3407,
    output_dir = "outputs",
),
)

/usr/local/lib/python3.10/dist-packages/multiprocess/popen_fork.py:66: RuntimeWarning:
os.fork() was called. os.fork() is incompatible with multithreaded code, and JAX is
multithreaded, so this will likely lead to a deadlock.
    self.pid = os.fork()

Map (num_proc=2):  0%|          | 0/51760 [00:00<?, ? examples/s]
max_steps is given, it will override any value given in num_train_epochs

#@title Show current memory stats
gpu_stats = torch.cuda.get_device_properties(0)
start_gpu_memory = round(torch.cuda.max_memory_reserved() / 1024 / 1024 / 1024, 3)
max_memory = round(gpu_stats.total_memory / 1024 / 1024 / 1024, 3)
print(f"GPU = {gpu_stats.name}. Max memory = {max_memory} GB.")
print(f"{start_gpu_memory} GB of memory reserved.")

GPU = Tesla T4. Max memory = 14.748 GB.
9.383 GB of memory reserved.

trainer_stats = trainer.train()

==((=====))= Unslloth - 2x faster free finetuning | Num GPUs = 1
  \_\_ /| Num examples = 51,760 | Num Epochs = 1
0^0/ \_/\_ Batch size per device = 2 | Gradient Accumulation steps = 4
\_\_ / Total batch size = 8 | Total steps = 60
"-____-" Number of trainable parameters = 54,018,048

[60/60 11:30, Epoch 0/1]
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
StepTraining
Loss11.70060022.21150031.56670041.71880051.35810061.30090070.91860081.10410090.926800101.018500110.84500
0120.816000130.787000140.981800150.773800160.793500170.925000181.171500190.890300200.785600210.79430022
0.827900230.797400240.885000250.981200260.956000270.951200280.819800290.769500300.806300310.770300320.81
3000330.905700340.770500350.842000360.790200370.816000380.686200391.011700401.016600410.828500420.87210
0430.865800440.819100450.853700460.889000470.815800481.091100490.807300500.966600510.922600520.84490053
0.908100541.086400550.732400560.965900570.816200580.750000590.781500600.836500
```

```
#@title Show final memory and time stats
used_memory = round(torch.cuda.max_memory_reserved() / 1024 / 1024 / 1024, 3)
used_memory_for_lora = round(used_memory - start_gpu_memory, 3)
used_percentage = round(used_memory / max_memory * 100, 3)
lora_percentage = round(used_memory_for_lora / max_memory * 100, 3)
print(f'{trainer_stats.metrics["train_runtime"]} seconds used for training.')
print(f'{round(trainer_stats.metrics["train_runtime"]/60, 2)} minutes used for training.')
print(f'Peak reserved memory = {used_memory} GB.')
print(f'Peak reserved memory for training = {used_memory_for_lora} GB.')
print(f'Peak reserved memory % of max memory = {used_percentage} %.')
print(f'Peak reserved memory for training % of max memory = {lora_percentage} %.)
```

```
776.294 seconds used for training.
12.94 minutes used for training.
Peak reserved memory = 9.383 GB.
Peak reserved memory for training = 0.0 GB.
Peak reserved memory % of max memory = 63.622 %.
Peak reserved memory for training % of max memory = 0.0 %.
```

Inference

Let's run the model! You can change the instruction and input - leave the output blank!

```
# alpaca_prompt = Copied from above
FastLanguageModel.for_inference(model) # Enable native 2x faster inference
inputs = tokenizer([
    alpaca_prompt.format(
        "Continue the fibonacci sequence.", # instruction
        "1, 1, 2, 3, 5, 8", # input
        "", # output - leave this blank for generation!
    ),
], return_tensors = "pt").to("cuda")

outputs = model.generate(**inputs, max_new_tokens = 64, use_cache = True)
tokenizer.batch_decode(outputs)
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
[ '<bos>Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.\n\n###\nInstruction:\nContinue the fibonnaci sequence.\n\nInput:\n1, 2, 3, 5, 8\n\n###\nResponse:\n13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6']
```

You can also use a `TextStreamer` for continuous inference - so you can see the generation token by token, instead of waiting the whole time!

```
# alpaca_prompt = Copied from above
FastLanguageModel.for_inference(model) # Enable native 2x faster inference
inputs = tokenizer([
    alpaca_prompt.format(
        "Continue the fibonnaci sequence.", # instruction
        "1, 1, 2, 3, 5, 8", # input
        "", # output - leave this blank for generation!
    )
], return_tensors = "pt").to("cuda")

from transformers import TextStreamer
text_streamer = TextStreamer(tokenizer)
_ = model.generate(**inputs, streamer = text_streamer, max_new_tokens = 128)

<bos>Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

### Instruction:
Continue the fibonnaci sequence.

### Input:
1, 1, 2, 3, 5, 8

### Response:
13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657,
46368, 75025, 121393, 196418, 317811,
```

Saving, loading finetuned models

To save the final model as LoRA adapters, either use Huggingface's `push_to_hub` for an online save or `save_pretrained` for a local save.

[NOTE] This ONLY saves the LoRA adapters, and not the full model. To save to 16bit or GGUf, scroll down!

```
model.save_pretrained("gemma2_9b") # Local saving
tokenizer.save_pretrained("gemma2_9b_tokenizer")
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
# model.push_to_hub("ashishpatel26/gemma2_9b", token = "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") #  
# Online saving  
# tokenizer.push_to_hub("ashishpatel26/gemma2_9b_tokenizer", token =  
# "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") # Online saving  
  
('gemma2_9b_tokenizer/tokenizer_config.json',  
 'gemma2_9b_tokenizer/special_tokens_map.json',  
 'gemma2_9b_tokenizer/tokenizer.model',  
 'gemma2_9b_tokenizer/added_tokens.json',  
 'gemma2_9b_tokenizer/tokenizer.json')
```

Now if you want to load the LoRA adapters we just saved for inference, set `False` to `True`

```
if False:  
    from unsloth import FastLanguageModel  
    model, tokenizer = FastLanguageModel.from_pretrained(  
        model_name = "lora_model", # YOUR MODEL YOU USED FOR TRAINING  
        max_seq_length = max_seq_length,  
        dtype = dtype,  
        load_in_4bit = load_in_4bit,  
    )  
    FastLanguageModel.for_inference(model) # Enable native 2x faster inference  
  
    # alpaca_prompt = You MUST copy from above!  
  
    inputs = tokenizer(  
        [  
            alpaca_prompt.format(  
                "What is a famous tall tower in Paris?", # instruction  
                "", # input  
                "", # output - leave this blank for generation!  
        ]  
    ), return_tensors = "pt").to("cuda")  
  
    outputs = model.generate(**inputs, max_new_tokens = 64, use_cache = True) tokenizer.batch_decode(outputs)  
  
    ['<bos>Below is an instruction that describes a task, paired with an input that provides  
    further context. Write a response that appropriately completes the request.\n\n###  
    Instruction:\nWhat is a famous tall tower in Paris?\n\n### Input:\n\n### Response:\nThe  
    Eiffel Tower is a famous tall tower in Paris, France. It is located on the Champ de Mars  
    in the 7th arrondissement of Paris, on the left bank of the River Seine. The tower was  
    designed by engineer Gustave Eiffel and his company, and was constructed from 1887 to  
    18']  
You can also use Hugging Face's AutoModelForPeftCausalLM. Only use this if you do not have unsloth installed. It  
can be hopelessly slow, since 4bit model downloading is not supported, and Unsloth's inference is 2x faster.
```

```
if False:  
    # I highly do NOT suggest - use Unsloth if possible  
    from peft import AutoPeftModelForCausalLM  
    from transformers import AutoTokenizer  
    model = AutoPeftModelForCausalLM.from_pretrained(
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
"lora_model", # YOUR MODEL YOU USED FOR TRAINING  
load_in_4bit = load_in_4bit,  
) tokenizer = AutoTokenizer.from_pretrained("lora_model")
```

Saving to float16 for VLLM

We also support saving to `float16` directly. Select `merged_16bit` for float16 or `merged_4bit` for int4. We also allow `lora` adapters as a fallback. Use `push_to_hub_merged` to upload to your Hugging Face account! You can go to <https://huggingface.co/settings/tokens> for your personal tokens.

```
# Merge to 16bit  
if False: model.save_pretrained_merged("model", tokenizer, save_method = "merged_16bit",)  
if False: model.push_to_hub_merged("hf/model", tokenizer, save_method = "merged_16bit", token = "")  
  
# Merge to 4bit  
if False: model.save_pretrained_merged("model", tokenizer, save_method = "merged_4bit",)  
if False: model.push_to_hub_merged("hf/model", tokenizer, save_method = "merged_4bit", token = "")  
  
# Just LoRA adapters  
if False: model.save_pretrained_merged("model", tokenizer, save_method = "lora",)  
if False: model.push_to_hub_merged("hf/model", tokenizer, save_method = "lora", token = "")
```

GGUF/llama.cpp Conversion

To save to GGUF / llama.cpp, we support it natively now! We clone llama.cpp and we default save it to `q8_0`. We allow all methods like `q4_k_m`. Use `save_pretrained_gguf` for local saving and `push_to_hub_gguf` for uploading to HF.

Some supported quant methods

- `q8_0` - Fast conversion. High resource use, but generally acceptable.
- `q4_k_m` - Recommended. Uses `Q6_K` for half of the `attention.wv` and `feed_forward.w2` tensors, else `Q4_K`.
- `q5_k_m` - Recommended. Uses `Q6_K` for half of the `attention.wv` and `feed_forward.w2` tensors, else `Q5_K`.

```
# Save to 8bit Q8_0  
if False: model.save_pretrained_gguf("model", tokenizer,)  
if False: model.push_to_hub_gguf("hf/model", tokenizer, token = "")  
  
# Save to 16bit GGUF  
if False: model.save_pretrained_gguf("model", tokenizer, quantization_method = "f16")  
if False: model.push_to_hub_gguf("hf/model", tokenizer, quantization_method = "f16", token = "")  
  
# Save to q4_k_m GGUF  
if False: model.save_pretrained_gguf("model", tokenizer, quantization_method = "q4_k_m")
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
if False: model.push_to_hub_gguf("hf/model", tokenizer, quantization_method = "q4_k_m", token = "")
```

Now, use the `model-unsloth.gguf` file or `model-unsloth-Q4_K_M.gguf` file in `llama.cpp` or a UI based system like GPT4All. You can install GPT4All on the internet

Some other links:

LLM RAG Evaluation with MLflow Example

In this notebook, we will demonstrate how to evaluate various a RAG system with MLflow.

We need to set our OpenAI API key.

In order to set your private key safely, please be sure to either export your key through a command-line terminal for your current instance, or, for a permanent addition to all user-based sessions, configure your favored environment management configuration file (i.e., `.bashrc`, `.zshrc`) to have the following entry:

```
OPENAI_API_KEY=<your openai API key>
```

If using Azure OpenAI, you will instead need to set

```
OPENAI_API_TYPE="azure"
```

```
OPENAI_API_VERSION=<YYYY-MM-DD>
```

```
OPENAI_API_KEY=<https://<>.<>.<>.com>
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

OPENAI_API_DEPLOYMENT_NAME=<deployment name>

Notebook compatibility

With rapidly changing libraries such as `langchain`, examples can become outdated rather quickly and will no longer work. For the purposes of demonstration, here are the critical dependencies that are recommended to use to effectively run this notebook:

Package	Version
langchain	0.1.16
lanchain-community	0.0.33
langchain-openai	0.0.8
openai	1.12.0
mlflow	2.12.1
chromadb	0.4.24

If you attempt to execute this notebook with different versions, it may function correctly, but it is recommended to use the precise versions above to ensure that your code executes properly.

Create a RAG system

Use Langchain and Chroma to create a RAG system that answers questions based on the MLflow documentation.

```
!pip install -U langchain langchain-community langchain-openai openai mlflow chromadb  
evaluate
```

```
Requirement already satisfied: langchain in /usr/local/lib/python3.10/dist-packages  
(0.1.16) Collecting langchain Downloading langchain-0.2.6-py3-none-any.whl (975 kB)  
_____ 975.5/975.5 kB 20.5 MB/s eta 0:00:00  
Requirement already satisfied: langchain-community in /usr/local/lib/python3.10/dist-packages  
(0.0.33) Collecting langchain-community Downloading langchain_community-0.2.6-  
py3-none-any.whl (2.2 MB) _____ 2.2/2.2 MB 69.8 MB/s  
eta 0:00:00 Requirement already satisfied: langchain-openai in  
/usr/local/lib/python3.10/dist-packages (0.0.8) Collecting langchain-openai Downloading  
langchain_openai-0.1.14-py3-none-any.whl (45 kB)  
_____ 45.9/45.9 kB 7.7 MB/s eta 0:00:00 Requirement  
already satisfied: openai in /usr/local/lib/python3.10/dist-packages (1.12.0) Collecting  
openai Downloading openai-1.35.10-py3-none-any.whl (328 kB)  
_____ 328.3/328.3 kB 40.2 MB/s eta 0:00:00
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
Requirement already satisfied: mlflow in /usr/local/lib/python3.10/dist-packages  
(2.12.1) Collecting mlflow Downloading mlflow-2.14.2-py3-none-any.whl (25.8 MB)  
      25.8/25.8 MB 53.7 MB/s eta 0:00:00 Requirement  
already satisfied: chromadb in /usr/local/lib/python3.10/dist-packages (0.4.24)  
Collecting chromadb Downloading chromadb-0.5.3-py3-none-any.whl (559 kB)  
      559.5/559.5 kB 46.7 MB/s eta 0:00:00 Collecting  
evaluate Downloading evaluate-0.4.2-py3-none-any.whl (84 kB)  
      84.1/84.1 kB 13.6 MB/s eta 0:00:00 Requirement  
already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.10/dist-packages (from  
langchain) (6.0.1) Requirement already satisfied: SQLAlchemy<3,>=1.4 in  
/usr/local/lib/python3.10/dist-packages (from langchain) (2.0.31) Requirement already  
satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/python3.10/dist-packages (from  
langchain) (3.9.5) Requirement already satisfied: async-timeout<5.0.0,>=4.0.0 in  
/usr/local/lib/python3.10/dist-packages (from langchain) (4.0.3) Collecting langchain-  
core<0.3.0,>=0.2.10 (from langchain) Downloading langchain_core-0.2.11-py3-none-any.whl  
(337 kB) 337.4/337.4 kB 40.8 MB/s eta 0:00:00  
Collecting langchain-text-splitters<0.3.0,>=0.2.0 (from langchain) Downloading  
langchain_text_splitters-0.2.2-py3-none-any.whl (25 kB) Requirement already satisfied:  
langsmith<0.2.0,>=0.1.17 in /usr/local/lib/python3.10/dist-packages (from langchain)  
(0.1.84) Requirement already satisfied: numpy<2,>=1 in /usr/local/lib/python3.10/dist-  
packages (from langchain) (1.25.2) Requirement already satisfied: pydantic<3,>=1 in  
/usr/local/lib/python3.10/dist-packages (from langchain) (2.8.0) Requirement already  
satisfied: requests<3,>=2 in /usr/local/lib/python3.10/dist-packages (from langchain)  
(2.31.0) Requirement already satisfied: tenacity!=8.4.0,<9.0.0,>=8.1.0 in  
/usr/local/lib/python3.10/dist-packages (from langchain) (8.4.2) Requirement already  
satisfied: dataclasses-json<0.7,>=0.5.7 in /usr/local/lib/python3.10/dist-packages (from  
langchain-community) (0.6.7) Requirement already satisfied: tiktoken<1,>=0.7 in  
/usr/local/lib/python3.10/dist-packages (from langchain-openai) (0.7.0) Requirement  
already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.10/dist-packages (from  
openai) (3.7.1) Requirement already satisfied: distro<2,>=1.7.0 in  
/usr/lib/python3/dist-packages (from openai) (1.7.0) Requirement already satisfied:  
httpx<1,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from openai) (0.27.0)  
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages (from  
openai) (1.3.1) Requirement already satisfied: tqdm>4 in /usr/local/lib/python3.10/dist-  
packages (from openai) (4.66.4) Requirement already satisfied: typing-extensions<5,>=4.7  
in /usr/local/lib/python3.10/dist-packages (from openai) (4.12.2) Requirement already  
satisfied: Flask<4 in /usr/local/lib/python3.10/dist-packages (from mlflow) (2.2.5)  
Requirement already satisfied: alembic!=1.10.0,<2 in /usr/local/lib/python3.10/dist-  
packages (from mlflow) (1.13.2) Requirement already satisfied: cachetools<6,>=5.0.0 in  
/usr/local/lib/python3.10/dist-packages (from mlflow) (5.3.3) Requirement already  
satisfied: click<9,>=7.0 in /usr/local/lib/python3.10/dist-packages (from mlflow)  
(8.1.7) Requirement already satisfied:云cloudpickle<4 in /usr/local/lib/python3.10/dist-  
packages (from mlflow) (2.2.1) Requirement already satisfied: docker<8,>=4.0.0 in  
/usr/local/lib/python3.10/dist-packages (from mlflow) (7.1.0) Requirement already  
satisfied: entrypoints<1 in /usr/local/lib/python3.10/dist-packages (from mlflow) (0.4)  
Requirement already satisfied: gitpython<4,>=3.1.9 in /usr/local/lib/python3.10/dist-  
packages (from mlflow) (3.1.43) Requirement already satisfied: graphene<4 in  
/usr/local/lib/python3.10/dist-packages (from mlflow) (3.3) Requirement already  
satisfied: importlib-metadata!=4.7.0,<8,>=3.7.0 in /usr/local/lib/python3.10/dist-  
packages (from mlflow) (7.1.0) Requirement already satisfied: markdown<4,>=3.3 in
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
/usr/local/lib/python3.10/dist-packages (from mlflow) (3.6) Requirement already
satisfied: matplotlib<4 in /usr/local/lib/python3.10/dist-packages (from mlflow) (3.7.1)
Requirement already satisfied: opentelemetry-api<3,>=1.9.0 in
/usr/local/lib/python3.10/dist-packages (from mlflow) (1.25.0) Requirement already
satisfied: opentelemetry-sdk<3,>=1.9.0 in /usr/local/lib/python3.10/dist-packages (from
mlflow) (1.25.0) Requirement already satisfied: packaging<25 in
/usr/local/lib/python3.10/dist-packages (from mlflow) (23.2) Requirement already
satisfied: pandas<3 in /usr/local/lib/python3.10/dist-packages (from mlflow) (2.0.3)
Requirement already satisfied: protobuf<5,>=3.12.0 in /usr/local/lib/python3.10/dist-
packages (from mlflow) (3.20.3) Requirement already satisfied: pyarrow<16,>=4.0.0 in
/usr/local/lib/python3.10/dist-packages (from mlflow) (14.0.2) Requirement already
satisfied: pytz<2025 in /usr/local/lib/python3.10/dist-packages (from mlflow) (2023.4)
Requirement already satisfied: querystring-parser<2 in /usr/local/lib/python3.10/dist-
packages (from mlflow) (1.2.4) Requirement already satisfied: scikit-learn<2 in
/usr/local/lib/python3.10/dist-packages (from mlflow) (1.2.2) Requirement already
satisfied: scipy<2 in /usr/local/lib/python3.10/dist-packages (from mlflow) (1.11.4)
Requirement already satisfied: sqlparse<1,>=0.4.0 in /usr/local/lib/python3.10/dist-
packages (from mlflow) (0.5.0) Requirement already satisfied: Jinja2<4,>=2.11 in
/usr/local/lib/python3.10/dist-packages (from mlflow) (3.1.4) Requirement already
satisfied: gunicorn<23 in /usr/local/lib/python3.10/dist-packages (from mlflow) (21.2.0)
Requirement already satisfied: build>=1.0.3 in /usr/local/lib/python3.10/dist-packages
(from chromadb) (1.2.1) Requirement already satisfied: chroma-hnswlib==0.7.3 in
/usr/local/lib/python3.10/dist-packages (from chromadb) (0.7.3) Requirement already
satisfied: fastapi>=0.95.2 in /usr/local/lib/python3.10/dist-packages (from chromadb)
(0.111.0) Requirement already satisfied: uvicorn[standard]>=0.18.3 in
/usr/local/lib/python3.10/dist-packages (from chromadb) (0.30.1) Requirement already
satisfied: posthog>=2.4.0 in /usr/local/lib/python3.10/dist-packages (from chromadb)
(3.5.0) Requirement already satisfied: onnxruntime>=1.14.1 in
/usr/local/lib/python3.10/dist-packages (from chromadb) (1.18.1) Requirement already
satisfied: opentelemetry-exporter-otlp-proto-grpc>=1.2.0 in
/usr/local/lib/python3.10/dist-packages (from chromadb) (1.25.0) Requirement already
satisfied: opentelemetry-instrumentation-fastapi>=0.41b0 in
/usr/local/lib/python3.10/dist-packages (from chromadb) (0.46b0) Requirement already
satisfied: tokenizers>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from chromadb)
(0.19.1) Requirement already satisfied: pypika>=0.48.9 in
/usr/local/lib/python3.10/dist-packages (from chromadb) (0.48.9) Requirement already
satisfied: overrides>=7.3.1 in /usr/local/lib/python3.10/dist-packages (from chromadb)
(7.7.0) Requirement already satisfied: importlib-resources in
/usr/local/lib/python3.10/dist-packages (from chromadb) (6.4.0) Requirement already
satisfied: grpcio>=1.58.0 in /usr/local/lib/python3.10/dist-packages (from chromadb)
(1.64.1) Requirement already satisfied: bcrypt>=4.0.1 in /usr/local/lib/python3.10/dist-
packages (from chromadb) (4.1.3) Requirement already satisfied: typer>=0.9.0 in
/usr/local/lib/python3.10/dist-packages (from chromadb) (0.12.3) Requirement already
satisfied: kubernetes>=28.1.0 in /usr/local/lib/python3.10/dist-packages (from chromadb)
(30.1.0) Requirement already satisfied: mmh3>=4.0.1 in /usr/local/lib/python3.10/dist-
packages (from chromadb) (4.1.0) Requirement already satisfied: orjson>=3.9.12 in
/usr/local/lib/python3.10/dist-packages (from chromadb) (3.10.6) Collecting
datasets>=2.0.0 (from evaluate) Downloading datasets-2.20.0-py3-none-any.whl (547 kB)
----- 547.8/547.8 kB 48.2 MB/s eta 0:00:00 Collecting
dill (from evaluate) Downloading dill-0.3.8-py3-none-any.whl (116 kB)
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
116.3/116.3 kB 18.9 MB/s eta 0:00:00 Collecting
xxhash (from evaluate) Downloading xxhash-3.4.1-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
194.1/194.1 kB 29.2 MB/s eta 0:00:00 Collecting
multiprocess (from evaluate) Downloading multiprocess-0.70.16-py310-none-any.whl (134
kB) 13
4.8/134.8 kB 22.6 MB/s eta 0:00:00
Requirement already satisfied: fsspec[http]>=2021.05.0 in /usr/local/lib/python3.10/dist-
packages (from evaluate) (2023.6.0)
Requirement already satisfied: huggingface-hub>=0.7.0 in /usr/local/lib/python3.10/dist-
packages (from evaluate) (0.23.4)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-
packages (from aiohttp<4.0.0,>=3.8.3->langchain) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages
(from aiohttp<4.0.0,>=3.8.3->langchain) (23.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from aiohttp<4.0.0,>=3.8.3->langchain) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-
packages (from aiohttp<4.0.0,>=3.8.3->langchain) (6.0.5)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages
(from aiohttp<4.0.0,>=3.8.3->langchain) (1.9.4)
Requirement already satisfied: Mako in /usr/local/lib/python3.10/dist-packages (from
alembic!=1.10.0,<2->mlflow) (1.3.5)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-packages (from
anyio<5,>=3.5.0->openai) (3.7)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages
(from anyio<5,>=3.5.0->openai) (1.2.1)
Requirement already satisfied: pyproject_hooks in /usr/local/lib/python3.10/dist-packages
(from build>=1.0.3->chromadb) (1.1.0)
Requirement already satisfied: toml>=1.1.0 in /usr/local/lib/python3.10/dist-packages
(from build>=1.0.3->chromadb) (2.0.1)
Requirement already satisfied: marshmallow<4.0.0,>=3.18.0 in
/usr/local/lib/python3.10/dist-packages (from dataclasses-json<0.7,>=0.5.7->langchain-
community) (3.21.3)
Requirement already satisfied: typing-inspect<1,>=0.4.0 in
/usr/local/lib/python3.10/dist-packages (from dataclasses-json<0.7,>=0.5.7->langchain-
community) (0.9.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from
datasets>=2.0.0->evaluate) (3.15.4)
Collecting pyarrow<16,>=4.0.0 (from mlflow)
    Downloading pyarrow-15.0.2-cp310-cp310-manylinux_2_28_x86_64.whl (38.3 MB)
38.3/38.3 kB 17.9 MB/s eta 0:00:00
Requirement already satisfied: pyarrow-hotfix in /usr/local/lib/python3.10/dist-packages
(from datasets>=2.0.0->evaluate) (0.6)
Collecting requests<3,>=2 (from langchain)
    Downloading requests-2.32.3-py3-none-any.whl (64 kB)
64.9/64.9 kB 10.8 MB/s eta 0:00:00
Requirement already satisfied: urllib3>=1.26.0 in /usr/local/lib/python3.10/dist-packages
(from docker<8,>=4.0.0->mlflow) (2.0.7)
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
Requirement already satisfied: starlette<0.38.0,>=0.37.2 in
/usr/local/lib/python3.10/dist-packages (from fastapi>=0.95.2->chromadb) (0.37.2)
Requirement already satisfied: fastapi-cli>=0.0.2 in /usr/local/lib/python3.10/dist-
packages (from fastapi>=0.95.2->chromadb) (0.0.4)
Requirement already satisfied: python-multipart>=0.0.7 in /usr/local/lib/python3.10/dist-
packages (from fastapi>=0.95.2->chromadb) (0.0.9)
Requirement already satisfied:
ujson!=4.0.2,!=4.1.0,!=4.2.0,!=4.3.0,!=5.0.0,!=5.1.0,>=4.0.1 in
/usr/local/lib/python3.10/dist-packages (from fastapi>=0.95.2->chromadb) (5.10.0)
Requirement already satisfied: email_validator>=2.0.0 in /usr/local/lib/python3.10/dist-
packages (from fastapi>=0.95.2->chromadb) (2.2.0)
Requirement already satisfied: Werkzeug>=2.2.2 in /usr/local/lib/python3.10/dist-packages
(from Flask<4->mlflow) (3.0.3)
Requirement already satisfied: itsdangerous>=2.0 in /usr/local/lib/python3.10/dist-
packages (from Flask<4->mlflow) (2.2.0)
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.10/dist-packages
(from gitpython<4,>=3.1.9->mlflow) (4.0.11)
Requirement already satisfied: graphql-core<3.3,>=3.1 in /usr/local/lib/python3.10/dist-
packages (from graphene<4->mlflow) (3.2.3)
Requirement already satisfied: graphql-relay<3.3,>=3.1 in /usr/local/lib/python3.10/dist-
packages (from graphene<4->mlflow) (3.2.0)
Requirement already satisfied: aniso8601<10,>=8 in /usr/local/lib/python3.10/dist-
packages (from graphene<4->mlflow) (9.0.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from
httpx<1,>=0.23.0->openai) (2024.6.2)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.10/dist-packages
(from httpx<1,>=0.23.0->openai) (1.0.5)
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.10/dist-packages
(from httpcore==1.*->httpx<1,>=0.23.0->openai) (0.14.0)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.10/dist-packages (from
importlib-metadata!=4.7.0,<8,>=3.7.0->mlflow) (3.19.2)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages
(from Jinja2<4,>=2.11->mlflow) (2.1.5)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.10/dist-packages
(from kubernetes>=28.1.0->chromadb) (1.16.0)
Requirement already satisfied: python-dateutil>=2.5.3 in /usr/local/lib/python3.10/dist-
packages (from kubernetes>=28.1.0->chromadb) (2.8.2)
Requirement already satisfied: google-auth>=1.0.1 in /usr/local/lib/python3.10/dist-
packages (from kubernetes>=28.1.0->chromadb) (2.27.0)
Requirement already satisfied: websocket-client!=0.40.0,!=0.41.*,!=0.42.*,>=0.32.0 in
/usr/local/lib/python3.10/dist-packages (from kubernetes>=28.1.0->chromadb) (1.8.0)
Requirement already satisfied: requests-oauthlib in /usr/local/lib/python3.10/dist-
packages (from kubernetes>=28.1.0->chromadb) (1.3.1)
Requirement already satisfied: oauthlib>=3.2.2 in /usr/local/lib/python3.10/dist-packages
(from kubernetes>=28.1.0->chromadb) (3.2.2)
Requirement already satisfied: jsonpatch<2.0,>=1.33 in /usr/local/lib/python3.10/dist-
packages (from langchain-core<0.3.0,>=0.2.10->langchain) (1.33)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-
packages (from matplotlib<4->mlflow) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
(from matplotlib<4->mlflow) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib<4->mlflow) (4.53.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-
packages (from matplotlib<4->mlflow) (1.4.5)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages
(from matplotlib<4->mlflow) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-
packages (from matplotlib<4->mlflow) (3.1.2)
Requirement already satisfied: coloredlogs in /usr/local/lib/python3.10/dist-packages
(from onnxruntime>=1.14.1->chromadb) (15.0.1)
Requirement already satisfied: flatbuffers in /usr/local/lib/python3.10/dist-packages
(from onnxruntime>=1.14.1->chromadb) (24.3.25)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from
onnxruntime>=1.14.1->chromadb) (1.12.1)
Requirement already satisfied: deprecated>=1.2.6 in /usr/local/lib/python3.10/dist-
packages (from opentelemetry-api<3,>=1.9.0->mlflow) (1.2.14)
Requirement already satisfied: googleapis-common-protos~=1.52 in
/usr/local/lib/python3.10/dist-packages (from opentelemetry-exporter-otlp-proto-
grpc>=1.2.0->chromadb) (1.63.2)
Requirement already satisfied: opentelemetry-exporter-otlp-proto-common==1.25.0 in
/usr/local/lib/python3.10/dist-packages (from opentelemetry-exporter-otlp-proto-
grpc>=1.2.0->chromadb) (1.25.0)
Requirement already satisfied: opentelemetry-proto==1.25.0 in
/usr/local/lib/python3.10/dist-packages (from opentelemetry-exporter-otlp-proto-
grpc>=1.2.0->chromadb) (1.25.0)
Requirement already satisfied: opentelemetry-instrumentation-asgi==0.46b0 in
/usr/local/lib/python3.10/dist-packages (from opentelemetry-instrumentation-
fastapi>=0.41b0->chromadb) (0.46b0)
Requirement already satisfied: opentelemetry-instrumentation==0.46b0 in
/usr/local/lib/python3.10/dist-packages (from opentelemetry-instrumentation-
fastapi>=0.41b0->chromadb) (0.46b0)
Requirement already satisfied: opentelemetry-semantic-conventions==0.46b0 in
/usr/local/lib/python3.10/dist-packages (from opentelemetry-instrumentation-
fastapi>=0.41b0->chromadb) (0.46b0)
Requirement already satisfied: opentelemetry-util-http==0.46b0 in
/usr/local/lib/python3.10/dist-packages (from opentelemetry-instrumentation-
fastapi>=0.41b0->chromadb) (0.46b0)
Requirement already satisfied: setuptools>=16.0 in /usr/local/lib/python3.10/dist-
packages (from opentelemetry-instrumentation==0.46b0->opentelemetry-instrumentation-
fastapi>=0.41b0->chromadb) (67.7.2)
Requirement already satisfied: wrapt<2.0.0,>=1.0.0 in /usr/local/lib/python3.10/dist-
packages (from opentelemetry-instrumentation==0.46b0->opentelemetry-instrumentation-
fastapi>=0.41b0->chromadb) (1.14.1)
Requirement already satisfied: asgiref~=3.0 in /usr/local/lib/python3.10/dist-packages
(from opentelemetry-instrumentation-asgi==0.46b0->opentelemetry-instrumentation-
fastapi>=0.41b0->chromadb) (3.8.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages
(from pandas<3->mlflow) (2024.1)
Requirement already satisfied: monotonic>=1.5 in /usr/local/lib/python3.10/dist-packages
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
(from posthog>=2.4.0->chromadb) (1.6)
Requirement already satisfied: backoff>=1.10.0 in /usr/local/lib/python3.10/dist-packages
(from posthog>=2.4.0->chromadb) (2.2.1)
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python3.10/dist-
packages (from pydantic<3,>=1->langchain) (0.7.0)
Requirement already satisfied: pydantic-core==2.20.0 in /usr/local/lib/python3.10/dist-
packages (from pydantic<3,>=1->langchain) (2.20.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langchain) (3.3.2)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages
(from scikit-learn<2->mlflow) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn<2->mlflow) (3.5.0)
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.10/dist-
packages (from SQLAlchemy<3,>=1.4->langchain) (3.0.3)
Requirement already satisfied: regex>=2022.1.18 in /usr/local/lib/python3.10/dist-
packages (from tiktoken<1,>=0.7->langchain-openai) (2024.5.15)
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.10/dist-
packages (from typer>=0.9.0->chromadb) (1.5.4)
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.10/dist-packages
(from typer>=0.9.0->chromadb) (13.7.1)
Requirement already satisfied: httptools>=0.5.0 in /usr/local/lib/python3.10/dist-
packages (from uvicorn[standard]>=0.18.3->chromadb) (0.6.1)
Requirement already satisfied: python-dotenv>=0.13 in /usr/local/lib/python3.10/dist-
packages (from uvicorn[standard]>=0.18.3->chromadb) (1.0.1)
Requirement already satisfied: uvloop!=0.15.0,!>=0.15.1,>=0.14.0 in
/usr/local/lib/python3.10/dist-packages (from uvicorn[standard]>=0.18.3->chromadb)
(0.19.0)
Requirement already satisfied: watchfiles>=0.13 in /usr/local/lib/python3.10/dist-
packages (from uvicorn[standard]>=0.18.3->chromadb) (0.22.0)
Requirement already satisfied: websockets>=10.4 in /usr/local/lib/python3.10/dist-
packages (from uvicorn[standard]>=0.18.3->chromadb) (12.0)
Requirement already satisfied: dnspython>=2.0.0 in /usr/local/lib/python3.10/dist-
packages (from email_validator>=2.0.0->fastapi>=0.95.2->chromadb) (2.6.1)
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.10/dist-packages
(from gitdb<5,>=4.0.1->gitpython<4,>=3.1.9->mlflow) (5.0.1)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-
packages (from google-auth>=1.0.1->kubernetes>=28.1.0->chromadb) (0.4.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages
(from google-auth>=1.0.1->kubernetes>=28.1.0->chromadb) (4.9)
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.10/dist-
packages (from jsonpatch<2.0,>=1.33->langchain-core<0.3.0,>=0.2.10->langchain) (3.0.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-
packages (from rich>=10.11.0->typer>=0.9.0->chromadb) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-
packages (from rich>=10.11.0->typer>=0.9.0->chromadb) (2.16.1)
Requirement already satisfied: mypy-extensions>=0.3.0 in /usr/local/lib/python3.10/dist-
packages (from typing-inspect<1,>=0.4.0->dataclasses-json<0.7,>=0.5.7->langchain-
community) (1.0.0)
Requirement already satisfied: humanfriendly>=9.1 in /usr/local/lib/python3.10/dist-
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
packages (from coloredlogs->onnxruntime>=1.14.1->chromadb) (10.0)
Requirement already satisfied: mpmath<1.4.0,>=1.1.0 in /usr/local/lib/python3.10/dist-
packages (from sympy->onnxruntime>=1.14.1->chromadb) (1.3.0)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages
(from markdown-it-py>=2.2.0->rich>=10.11.0->typer>=0.9.0->chromadb) (0.1.2)
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in /usr/local/lib/python3.10/dist-
packages (from pyasn1-modules>=0.2.1->google-auth>=1.0.1->kubernetes>=28.1.0->chromadb)
(0.6.0)
Installing collected packages: xxhash, requests, pyarrow, dill, multiprocessing, openai,
langchain-core, datasets, mlflow, langchain-text-splitters, langchain-openai, evaluate,
langchain, chromadb, langchain-community
Attempting uninstall: requests
  Found existing installation: requests 2.31.0
  Uninstalling requests-2.31.0:
    Successfully uninstalled requests-2.31.0
Attempting uninstall: pyarrow
  Found existing installation: pyarrow 14.0.2
  Uninstalling pyarrow-14.0.2:
    Successfully uninstalled pyarrow-14.0.2
Attempting uninstall: openai
  Found existing installation: openai 1.12.0
  Uninstalling openai-1.12.0:
    Successfully uninstalled openai-1.12.0
Attempting uninstall: langchain-core
  Found existing installation: langchain-core 0.1.52
  Uninstalling langchain-core-0.1.52:
    Successfully uninstalled langchain-core-0.1.52
Attempting uninstall: mlflow
  Found existing installation: mlflow 2.12.1
  Uninstalling mlflow-2.12.1:
    Successfully uninstalled mlflow-2.12.1
Attempting uninstall: langchain-text-splitters
  Found existing installation: langchain-text-splitters 0.0.2
  Uninstalling langchain-text-splitters-0.0.2:
    Successfully uninstalled langchain-text-splitters-0.0.2
Attempting uninstall: langchain-openai
  Found existing installation: langchain-openai 0.0.8
  Uninstalling langchain-openai-0.0.8:
    Successfully uninstalled langchain-openai-0.0.8
Attempting uninstall: langchain
  Found existing installation: langchain 0.1.16
  Uninstalling langchain-0.1.16:
    Successfully uninstalled langchain-0.1.16
Attempting uninstall: chromadb
  Found existing installation: chromadb 0.4.24
  Uninstalling chromadb-0.4.24:
    Successfully uninstalled chromadb-0.4.24
Attempting uninstall: langchain-community
  Found existing installation: langchain-community 0.0.33
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
Uninstalling langchain-community-0.0.33:  
  Successfully uninstalled langchain-community-0.0.33  
ERROR: pip's dependency resolver does not currently take into account all the packages  
that are installed. This behaviour is the source of the following dependency conflicts.  
cudf-cu12 24.4.1 requires pyarrow<15.0.0a0,>=14.0.1, but you have pyarrow 15.0.2 which is  
incompatible.  
google-colab 1.0.0 requires requests==2.31.0, but you have requests 2.32.3 which is  
incompatible.  
Successfully installed chromadb-0.5.3 datasets-2.20.0 dill-0.3.8 evaluate-0.4.2  
langchain-0.2.6 langchain-community-0.2.6 langchain-core-0.2.11 langchain-openai-0.1.14  
langchain-text-splitters-0.2.2 mlflow-2.14.2 multiprocess-0.70.16 openai-1.35.10 pyarrow-  
15.0.2 requests-2.32.3 xxhash-3.4.1
```

```
import pandas as pd  
from langchain.chains import RetrievalQA  
from langchain.document_loaders import WebBaseLoader  
from langchain.text_splitter import CharacterTextSplitter  
from langchain.vectorstores import Chroma  
from langchain_openai import OpenAI, OpenAIEMBEDDINGS  
  
import mlflow  
import os  
os.environ["OPENAI_API_KEY"] = "sk-"  
  
WARNING:langchain_community.utils.user_agent:USER_AGENT environment variable not set,  
consider setting it to identify your requests.
```

```
loader = WebBaseLoader("https://mlflow.org/docs/latest/index.html")  
  
documents = loader.load()  
text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=0)  
texts = text_splitter.split_documents(documents)  
  
embeddings = OpenAIEMBEDDINGS()  
docsearch = Chroma.from_documents(texts, embeddings)  
  
qa = RetrievalQA.from_chain_type(  
    llm=OpenAI(temperature=0),  
    chain_type="stuff",  
    retriever=docsearch.as_retriever(),  
    return_source_documents=True,  
)
```

Evaluate the RAG system using mlflow.evaluate()

Create a simple function that runs each input through the RAG chain

```
def model(input_df):  
    answer = []
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
for index, row in input_df.iterrows():
    answer.append(qa(row["questions"]))

return answer
```

Create an eval dataset

```
eval_df = pd.DataFrame(
{
    "questions": [
        "What is MLflow?",
        "How to run mlflow.evaluate()?",
        "How to log_table()?",
        "How to load_table()?",
    ],
})
```

Create a faithfulness metric

```
from mlflow.metrics.genai import EvaluationExample, faithfulness

# Create a good and bad example for faithfulness in the context of this problem
faithfulness_examples = [
    EvaluationExample(
        input="How do I disable MLflow autologging?",
        output="mlflow.autolog(disable=True) will disable autologging for all functions.
In Databricks, autologging is enabled by default. ",
        score=2,
        justification="The output provides a working solution, using the
mlflow.autolog() function that is provided in the context.",
        grading_context={
            "context": "mlflow.autolog(log_input_examples: bool = False,
log_model_signatures: bool = True, log_models: bool = True, log_datasets: bool = True,
disable: bool = False, exclusive: bool = False, disable_for_unsupported_versions: bool =
False, silent: bool = False, extra_tags: Optional[Dict[str, str]] = None) → None[source]
Enables (or disables) and configures autologging for all supported integrations. The
parameters are passed to any autologging integrations that support them. See the
tracking docs for a list of supported autologging integrations. Note that framework-
specific configurations set at any point will take precedence over any configurations
set by this function."
        },
    ),
    EvaluationExample(
        input="How do I disable MLflow autologging?",
        output="mlflow.autolog(disable=True) will disable autologging for all
functions.",
        score=5,
        justification="The output provides a solution that is using the mlflow.autolog()
function that is provided in the context.",
        grading_context={
            "context": "mlflow.autolog(log_input_examples: bool = False,
log_model_signatures: bool = True, log_models: bool = True, log_datasets: bool = True,
disable: bool = False, exclusive: bool = False, disable_for_unsupported_versions: bool =

```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
    False, silent: bool = False, extra_tags: Optional[Dict[str, str]] = None) → None[source]
Enables (or disables) and configures autologging for all supported integrations. The
parameters are passed to any autologging integrations that support them. See the
tracking docs for a list of supported autologging integrations. Note that framework-
specific configurations set at any point will take precedence over any configurations
set by this function."
    },
),
]
```

```
faithfulness_metric = faithfulness(model="openai:/gpt-4",
examples=faithfulness_examples)
print(faithfulness_metric)
```

```
EvaluationMetric(name=faithfulness, greater_is_better=True, long_name=faithfulness,
version=v1, metric_details=
Task:
You must return the following fields in your response in two lines, one below the other:
score: Your numerical score for the model's faithfulness based on the rubric
justification: Your reasoning about the model's faithfulness score
```

You are an impartial judge. You will be given an input that was sent to a machine learning model, and you will be given an output that the model produced. You may also be given additional information that was used by the model to generate the output.

Your task is to determine a numerical score called faithfulness based on the input and output.

A definition of faithfulness and a grading rubric are provided below.
You must use the grading rubric to determine your score. You must also justify your score.

Examples could be included below for reference. Make sure to use them as references and to understand them before completing the task.

Input:
{input}

Output:
{output}

{grading_context_columns}

Metric definition:

Faithfulness is only evaluated with the provided output and provided context, please ignore the provided input entirely when scoring faithfulness. Faithfulness assesses how much of the provided output is factually consistent with the provided context. A higher score indicates that a higher proportion of claims present in the output can be derived from the provided context. Faithfulness does not consider how much extra information from the context is not present in the output.

By: Tarun S Gowda

LLM-Finetuning with PEFT

Grading rubric:

Faithfulness: Below are the details for different scores:

- Score 1: None of the claims in the output can be inferred from the provided context.
- Score 2: Some of the claims in the output can be inferred from the provided context, but the majority of the output is missing from, inconsistent with, or contradictory to the provided context.
- Score 3: Half or more of the claims in the output can be inferred from the provided context.
- Score 4: Most of the claims in the output can be inferred from the provided context, with very little information that is not directly supported by the provided context.
- Score 5: All of the claims in the output are directly supported by the provided context, demonstrating high faithfulness to the provided context.

Examples:

Example Output:

`mlflow.autolog(disable=True)` will disable autologging for all functions. In Databricks, autologging is enabled by default.

Additional information used by the model:

key: context

value:

`mlflow.autolog(log_input_examples: bool = False, log_model_signatures: bool = True, log_models: bool = True, log_datasets: bool = True, disable: bool = False, exclusive: bool = False, disable_for_unsupported_versions: bool = False, silent: bool = False, extra_tags: Optional[Dict[str, str]] = None) → None[source]` Enables (or disables) and configures autologging for all supported integrations. The parameters are passed to any autologging integrations that support them. See the tracking docs for a list of supported autologging integrations. Note that framework-specific configurations set at any point will take precedence over any configurations set by this function.

Example score: 2

Example justification: The output provides a working solution, using the `mlflow.autolog()` function that is provided in the context.

Example Output:

`mlflow.autolog(disable=True)` will disable autologging for all functions.

Additional information used by the model:

key: context

value:

`mlflow.autolog(log_input_examples: bool = False, log_model_signatures: bool = True, log_models: bool = True, log_datasets: bool = True, disable: bool = False, exclusive: bool = False, disable_for_unsupported_versions: bool = False, silent: bool = False, extra_tags: Optional[Dict[str, str]] = None) → None[source]` Enables (or disables) and configures

By: Tarun S Gowda

LLM-Finetuning with PEFT

autologging for all supported integrations. The parameters are passed to any autologging integrations that support them. See the tracking docs for a list of supported autologging integrations. Note that framework-specific configurations set at any point will take precedence over any configurations set by this function.

Example score: 5

Example justification: The output provides a solution that is using the mlflow.autolog() function that is provided in the context.

You must return the following fields in your response in two lines, one below the other:

score: Your numerical score for the model's faithfulness based on the rubric

justification: Your reasoning about the model's faithfulness score

Do not add additional new lines. Do not add any other fields.

)

Create a relevance metric. You can see the full grading prompt by printing the metric or by accessing the metric_details attribute of the metric.

:

```
from mlflow.metrics.genai import EvaluationExample, relevance
```

```
relevance_metric = relevance(model="openai:/gpt-4")
print(relevance_metric)
```

EvaluationMetric(name=relevance, greater_is_better=True, long_name=relevance, version=v1,
metric_details=

Task:

You must return the following fields in your response in two lines, one below the other:

score: Your numerical score for the model's relevance based on the rubric

justification: Your reasoning about the model's relevance score

You are an impartial judge. You will be given an input that was sent to a machine learning model, and you will be given an output that the model produced. You may also be given additional information that was used by the model to generate the output.

Your task is to determine a numerical score called relevance based on the input and output. A definition of relevance and a grading rubric are provided below.

You must use the grading rubric to determine your score. You must also justify your score.

Examples could be included below for reference. Make sure to use them as references and to understand them before completing the task.

Input:

{input}

By: Tarun S Gowda

LLM-Finetuning with PEFT

Output:
{output}

{grading_context_columns}

Metric definition:

Relevance encompasses the appropriateness, significance, and applicability of the output with respect to both the input and context. Scores should reflect the extent to which the output directly addresses the question provided in the input, given the provided context.

Grading rubric:

Relevance: Below are the details for different scores:- Score 1: The output doesn't mention anything about the question or is completely irrelevant to the provided context.

- Score 2: The output provides some relevance to the question and is somehow related to the provided context.

- Score 3: The output mostly answers the question and is largely consistent with the provided context.

- Score 4: The output answers the question and is consistent with the provided context.

- Score 5: The output answers the question comprehensively using the provided context.

Examples:

Example Input:

How is MLflow related to Databricks?

Example Output:

Databricks is a data engineering and analytics platform designed to help organizations process and analyze large amounts of data. Databricks is a company specializing in big data and machine learning solutions.

Additional information used by the model:

key: context

value:

MLflow is an open-source platform for managing the end-to-end machine learning (ML) lifecycle. It was developed by Databricks, a company that specializes in big data and machine learning solutions. MLflow is designed to address the challenges that data scientists and machine learning engineers face when developing, training, and deploying machine learning models.

Example score: 2

Example justification: The output provides relevant information about Databricks, mentioning it as a company specializing in big data and machine learning solutions. However, it doesn't directly address how MLflow is related to Databricks, which is the specific question asked in the input. Therefore, the output is only somewhat related to the provided context.

By: Tarun S Gowda

LLM-Finetuning with PEFT

Example Input:

How is MLflow related to Databricks?

Example Output:

MLflow is a product created by Databricks to enhance the efficiency of machine learning processes.

Additional information used by the model:

key: context

value:

MLflow is an open-source platform for managing the end-to-end machine learning (ML) lifecycle. It was developed by Databricks, a company that specializes in big data and machine learning solutions. MLflow is designed to address the challenges that data scientists and machine learning engineers face when developing, training, and deploying machine learning models.

Example score: 4

Example justification: The output provides a relevant and accurate statement about the relationship between MLflow and Databricks. While it doesn't provide extensive detail, it still offers a substantial and meaningful response. To achieve a score of 5, the response could be further improved by providing additional context or details about how MLflow specifically functions within the Databricks ecosystem.

You must return the following fields in your response in two lines, one below the other:

score: Your numerical score for the model's relevance based on the rubric

justification: Your reasoning about the model's relevance score

Do not add additional new lines. Do not add any other fields.

)

```
results = mlflow.evaluate(
    model,
    eval_df,
    model_type="question-answering",
    evaluators="default",
    predictions="result",
    extra_metrics=[faithfulness_metric, relevance_metric, mlflow.metrics.latency()],
    evaluator_config={
        "col_mapping": {
            "inputs": "questions",
            "context": "source_documents",
        }
    },
)
print(results.metrics)
```

2024/07/08 05:44:13 INFO mlflow.models.evaluation.default_evaluator: Computing model predictions.

/usr/local/lib/python3.10/dist-packages/langchain_core/_api/deprecation.py:139:

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
LangChainDeprecationWarning: The method `Chain.__call__` was deprecated in langchain
0.1.0 and will be removed in 0.3.0. Use invoke instead.
    warn_deprecated(
2024/07/08 05:44:14 WARNING mlflow.tracing.processor.mlflow: Creating a trace within the
default experiment with id '0'. It is strongly recommended to not use the default
experiment to log traces due to ambiguous search results and probable performance issues
over time due to directory table listing performance degradation with high volumes of
directories within a specific path. To avoid performance and disambiguation issues, set
the experiment for your environment using `mlflow.set_experiment()` API.
2024/07/08 05:44:19 INFO mlflow.models.evaluation.default_evaluator: Testing metrics on
first row...

Downloading builder script:  0%|          | 0.00/6.08k [00:00<?, ?B/s]
WARNING:evaluate_modules.metrics.evaluate-measurement--
toxicity.2390290fa0bf6d78480143547c6b08f3d4f8805b249df8c7a8e80d0ce8e3778b.toxicity:Using
default facebook roberta-hate-speech-dynabench-r4-target checkpoint
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab
(https://huggingface.co/settings/tokens), set it as secret in your Google Colab and
restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models
or datasets.
    warnings.warn(
config.json:  0%|          | 0.00/816 [00:00<?, ?B/s]
model.safetensors:  0%|          | 0.00/499M [00:00<?, ?B/s]
tokenizer_config.json:  0%|          | 0.00/1.11k [00:00<?, ?B/s]
vocab.json:  0%|          | 0.00/899k [00:00<?, ?B/s]
merges.txt:  0%|          | 0.00/456k [00:00<?, ?B/s]
special_tokens_map.json:  0%|          | 0.00/239 [00:00<?, ?B/s]
2024/07/08 05:44:53 WARNING mlflow.metrics.metric_definitions: Failed to import textstat
for flesch kincaid metric, skipping metric logging. Please install textstat using 'pip
install textstat'.
2024/07/08 05:44:53 WARNING mlflow.models.evaluation.default_evaluator: Did not log
builtin metric 'flesch_kincaid_grade_level' because it returned None.
2024/07/08 05:44:53 WARNING mlflow.metrics.metric_definitions: Failed to import textstat
for automated readability index metric, skipping metric logging. Please install textstat
using 'pip install textstat'.
2024/07/08 05:44:53 WARNING mlflow.models.evaluation.default_evaluator: Did not log
builtin metric 'ari_grade_level' because it returned None.
2024/07/08 05:44:53 WARNING mlflow.models.evaluation.default_evaluator: Did not log
builtin metric 'exact_match' because it returned None.

 0%|          | 0/1 [00:00<?, ?it/s]
 0%|          | 0/1 [00:00<?, ?it/s]
2024/07/08 05:45:00 WARNING mlflow.metrics.metric_definitions: Failed to import textstat
for flesch kincaid metric, skipping metric logging. Please install textstat using 'pip
install textstat'.
2024/07/08 05:45:00 WARNING mlflow.models.evaluation.default_evaluator: Did not log
builtin metric 'flesch_kincaid_grade_level' because it returned None.
2024/07/08 05:45:00 WARNING mlflow.metrics.metric_definitions: Failed to import textstat
for automated readability index metric, skipping metric logging. Please install textstat
using 'pip install textstat'.
2024/07/08 05:45:00 WARNING mlflow.models.evaluation.default_evaluator: Did not log
builtin metric 'ari_grade_level' because it returned None.
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
2024/07/08 05:45:00 WARNING mlflow.models.evaluation.default_evaluator: Did not log
builtin metric 'exact_match' because it returned None.

0% | 0/4 [00:00<?, ?it/s]
WARNING:root:Request #0 failed with error {'message': 'Rate limit reached for gpt-4 in
organization org-tFcecN0lfDae98cjuXdWbfcY on tokens per min (TPM): Limit 10000, Used
7846, Requested 2254. Please try again in 600ms. Visit
https://platform.openai.com/account/rate-limits to learn more.', 'type': 'tokens',
'param': None, 'code': 'rate_limit_exceeded'}
WARNING:root:Request #0 failed with error {'message': 'Rate limit reached for gpt-4 in
organization org-tFcecN0lfDae98cjuXdWbfcY on tokens per min (TPM): Limit 10000, Used
7848, Requested 2665. Please try again in 3.078s. Visit
https://platform.openai.com/account/rate-limits to learn more.', 'type': 'tokens',
'param': None, 'code': 'rate_limit_exceeded'}
WARNING:root:Request #0 failed with error {'message': 'Rate limit reached for gpt-4 in
organization org-tFcecN0lfDae98cjuXdWbfcY on tokens per min (TPM): Limit 10000, Used
7833, Requested 2254. Please try again in 522ms. Visit
https://platform.openai.com/account/rate-limits to learn more.', 'type': 'tokens',
'param': None, 'code': 'rate_limit_exceeded'}
WARNING:root:Request #0 failed with error {'message': 'Rate limit reached for gpt-4 in
organization org-tFcecN0lfDae98cjuXdWbfcY on tokens per min (TPM): Limit 10000, Used
7831, Requested 2665. Please try again in 2.976s. Visit
https://platform.openai.com/account/rate-limits to learn more.', 'type': 'tokens',
'param': None, 'code': 'rate_limit_exceeded'}

0% | 0/4 [00:00<?, ?it/s]
WARNING:root:Request #0 failed with error {'message': 'Rate limit reached for gpt-4 in
organization org-tFcecN0lfDae98cjuXdWbfcY on tokens per min (TPM): Limit 10000, Used
7871, Requested 2554. Please try again in 2.55s. Visit
https://platform.openai.com/account/rate-limits to learn more.', 'type': 'tokens',
'param': None, 'code': 'rate_limit_exceeded'}
WARNING:root:Request #0 failed with error {'message': 'Rate limit reached for gpt-4 in
organization org-tFcecN0lfDae98cjuXdWbfcY on tokens per min (TPM): Limit 10000, Used
7868, Requested 2183. Please try again in 306ms. Visit
https://platform.openai.com/account/rate-limits to learn more.', 'type': 'tokens',
'param': None, 'code': 'rate_limit_exceeded'}
WARNING:root:Request #0 failed with error {'message': 'Rate limit reached for gpt-4 in
organization org-tFcecN0lfDae98cjuXdWbfcY on tokens per min (TPM): Limit 10000, Used
7869, Requested 2301. Please try again in 1.02s. Visit
https://platform.openai.com/account/rate-limits to learn more.', 'type': 'tokens',
'param': None, 'code': 'rate_limit_exceeded'}
WARNING:root:Request #0 failed with error {'message': 'Rate limit reached for gpt-4 in
organization org-tFcecN0lfDae98cjuXdWbfcY on tokens per min (TPM): Limit 10000, Used
7869, Requested 2596. Please try again in 2.79s. Visit
https://platform.openai.com/account/rate-limits to learn more.', 'type': 'tokens',
'param': None, 'code': 'rate_limit_exceeded'}
WARNING:root:Request #0 failed with error {'message': 'Rate limit reached for gpt-4 in
organization org-tFcecN0lfDae98cjuXdWbfcY on tokens per min (TPM): Limit 10000, Used
7860, Requested 2554. Please try again in 2.484s. Visit
https://platform.openai.com/account/rate-limits to learn more.', 'type': 'tokens',
'param': None, 'code': 'rate_limit_exceeded'}
WARNING:root:Request #0 failed with error {'message': 'Rate limit reached for gpt-4 in
organization org-tFcecN0lfDae98cjuXdWbfcY on tokens per min (TPM): Limit 10000, Used
7856, Requested 2596. Please try again in 2.712s. Visit
https://platform.openai.com/account/rate-limits to learn more.', 'type': 'tokens',
'param': None, 'code': 'rate_limit_exceeded'}
WARNING:root:Request #0 failed with error {'message': 'Rate limit reached for gpt-4 in
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
organization org-tFcecN0lfDae98cjuXdWbfcY on tokens per min (TPM): Limit 10000, Used  
7855, Requested 2183. Please try again in 228ms. Visit  
https://platform.openai.com/account/rate-limits to learn more.', 'type': 'tokens',  
'param': None, 'code': 'rate_limit_exceeded'}  
WARNING:root:Request #0 failed with error {'message': 'Rate limit reached for gpt-4 in  
organization org-tFcecN0lfDae98cjuXdWbfcY on tokens per min (TPM): Limit 10000, Used  
7852, Requested 2301. Please try again in 918ms. Visit  
https://platform.openai.com/account/rate-limits to learn more.', 'type': 'tokens',  
'param': None, 'code': 'rate_limit_exceeded'}  
WARNING:root:Request #0 failed with error {'message': 'Rate limit reached for gpt-4 in  
organization org-tFcecN0lfDae98cjuXdWbfcY on tokens per min (TPM): Limit 10000, Used  
7907, Requested 2301. Please try again in 1.248s. Visit  
https://platform.openai.com/account/rate-limits to learn more.', 'type': 'tokens',  
'param': None, 'code': 'rate_limit_exceeded'}  
WARNING:root:Request #0 failed with error {'message': 'Rate limit reached for gpt-4 in  
organization org-tFcecN0lfDae98cjuXdWbfcY on tokens per min (TPM): Limit 10000, Used  
7905, Requested 2596. Please try again in 3.006s. Visit  
https://platform.openai.com/account/rate-limits to learn more.', 'type': 'tokens',  
'param': None, 'code': 'rate_limit_exceeded'}  
WARNING:root:Request #0 failed with error {'message': 'Rate limit reached for gpt-4 in  
organization org-tFcecN0lfDae98cjuXdWbfcY on tokens per min (TPM): Limit 10000, Used  
7907, Requested 2183. Please try again in 540ms. Visit  
https://platform.openai.com/account/rate-limits to learn more.', 'type': 'tokens',  
'param': None, 'code': 'rate_limit_exceeded'}  
WARNING:root:Request #0 failed with error {'message': 'Rate limit reached for gpt-4 in  
organization org-tFcecN0lfDae98cjuXdWbfcY on tokens per min (TPM): Limit 10000, Used  
7893, Requested 2596. Please try again in 2.934s. Visit  
https://platform.openai.com/account/rate-limits to learn more.', 'type': 'tokens',  
'param': None, 'code': 'rate_limit_exceeded'}  
WARNING:root:Request #0 failed with error {'message': 'Rate limit reached for gpt-4 in  
organization org-tFcecN0lfDae98cjuXdWbfcY on tokens per min (TPM): Limit 10000, Used  
7890, Requested 2183. Please try again in 438ms. Visit  
https://platform.openai.com/account/rate-limits to learn more.', 'type': 'tokens',  
'param': None, 'code': 'rate_limit_exceeded'}  
WARNING:root:Request #0 failed with error {'message': 'Rate limit reached for gpt-4 in  
organization org-tFcecN0lfDae98cjuXdWbfcY on tokens per min (TPM): Limit 10000, Used  
7892, Requested 2301. Please try again in 1.158s. Visit  
https://platform.openai.com/account/rate-limits to learn more.', 'type': 'tokens',  
'param': None, 'code': 'rate_limit_exceeded'}  
WARNING:root:Request #0 failed with error {'message': 'Rate limit reached for gpt-4 in  
organization org-tFcecN0lfDae98cjuXdWbfcY on tokens per min (TPM): Limit 10000, Used  
9356, Requested 2596. Please try again in 11.712s. Visit  
https://platform.openai.com/account/rate-limits to learn more.', 'type': 'tokens',  
'param': None, 'code': 'rate_limit_exceeded'}  
WARNING:root:Request #0 failed with error {'message': 'Rate limit reached for gpt-4 in  
organization org-tFcecN0lfDae98cjuXdWbfcY on tokens per min (TPM): Limit 10000, Used  
9343, Requested 2596. Please try again in 11.634s. Visit  
https://platform.openai.com/account/rate-limits to learn more.', 'type': 'tokens',  
'param': None, 'code': 'rate_limit_exceeded'}  
  
{'latency/mean': 1.5472033023834229, 'latency/variance': 0.2672591728462237,  
'latency/p90': 2.0967818975448607, 'toxicity/v1/mean': 0.0001993731602851767,  
'toxicity/v1/variance': 9.251570116961669e-09, 'toxicity/v1/p90': 0.0003003335921675899,  
'toxicity/v1/ratio': 0.0, 'faithfulness/v1/mean': 5.0, 'faithfulness/v1/variance': 0.0,  
'faithfulness/v1/p90': 5.0, 'relevance/v1/mean': 3.75, 'relevance/v1/variance': 2.6875,  
'relevance/v1/p90': 5.0}
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
results.tables["eval_results_table"]

Downloading artifacts:  0% | 0/1 [00:00<?, ?it/s]

questionsoutputssource_documentslatencytoken_counttoxicity/v1/scorefai
thfulness/v1/scorefaithfulness/v1/justificationrelevance/v1/scorerelev
ance/v1/justification0What is MLflow?MLflow is an open-source platform
designed to...[{'id': None, 'lc_attributes': {}},
'lc_secrets...2.151866480.0001405The output accurately describes
MLflow as an o...5The output provides a comprehensive answer
to ...1How to run mlflow.evaluate()?To run mlflow.evaluate(), you
would first nee...[{'id': None, 'lc_attributes': {}},
'lc_secrets...1.968252900.0003665The output accurately describes the
process of...5The output provides a comprehensive answer to ...2How to
log_table()?I don't know, as there is no mention of a "lo...[{'id':
None, 'lc_attributes': {}}, 'lc_secrets...1.041242220.0001445The output
claim that there is no mention of a...4The output directly addresses
the question in ...3How to load_table()?I don't know, as the context
provided does no...[{'id': None, 'lc_attributes': {}},
'lc_secrets...1.027452170.0001475The output claim that the model
doesn't know a...1The output does not provide any relevant infor...
```

Evaluate a Hugging Face LLM with `mlflow.evaluate()`

This final part will show how to load a pre-trained Hugging Face pipeline, log it to MLflow, and use `mlflow.evaluate()` to evaluate builtin metrics as well as custom LLM-judged metrics for the model.

For detailed information, please read the documentation on using MLflow evaluate.

By: Tarun S Gowda

Start MLflow Server

You can either:

- Start a local tracking server by running `mlflow ui` within the same directory that your notebook is in.
 - Use a tracking server, as described in this overview.

Install necessary dependencies

```
%pip install -q mlflow transformers torch torchvision evaluate datasets openai tiktoken fastapi rouge_score textstat
```

Preparing metadata (setup.py) ... done

25.8/25.8 MB 22.5 MB/s eta 0:00:00
84.1/84.1 kB 3.6 MB/s eta 0:00:00
547.8/547.8 kB 10.0 MB/s eta 0:00:00
328.3/328.3 kB 10.4 MB/s eta 0:00:00
1.1/1.1 MB 26.5 MB/s eta 0:00:00
92.0/92.0 kB 1.5 MB/s eta 0:00:00

105.1/105.1 kB 5.0 MB/s eta 0:00:00
233.0/233.0 kB 7.5 MB/s eta 0:00:00
147.8/147.8 kB 4.2 MB/s eta 0:00:00
207.3/207.3 kB 7.5 MB/s eta 0:00:00
128.2/128.2 kB 10.3 MB/s eta 0:00:00
59.9/59.9 kB 1.9 MB/s eta 0:00:00
107.0/107.0 kB 4.1 MB/s eta 0:00:00
84.4/84.4 kB 4.3 MB/s eta 0:00:00
21.3/21.3 MB 56.5 MB/s eta 0:00:00
116.3/116.3 kB 11.3 MB/s eta 0:00:00
194.1/194.1 kB 18.9 MB/s eta 0:00:00
134.8/134.8 kB 14.8 MB/s eta 0:00:00
38.3/38.3 MB 12.9 MB/s eta 0:00:00
64.9/64.9 kB 7.3 MB/s eta 0:00:00
75.6/75.6 kB 7.5 MB/s eta 0:00:00
71.9/71.9 kB 7.5 MB/s eta 0:00:00
53.6/53.6 kB 4.3 MB/s eta 0:00:00
141.1/141.1 kB 12.5 MB/s eta 0:00:00
62.4/62.4 kB 5.0 MB/s eta 0:00:00
2.1/2.1 MB 54.1 MB/s eta 0:00:00
78.6/78.6 kB 4.1 MB/s eta 0:00:00
307.7/307.7 kB 19.4 MB/s eta 0:00:00
62.7/62.7 kB 4.6 MB/s eta 0:00:00
202.9/202.9 kB 15.1 MB/s eta 0:00:00
52.8/52.8 kB 4.8 MB/s eta 0:00:00
77.9/77.9 kB 8.5 MB/s eta 0:00:00
58.3/58.3 kB 5.6 MB/s eta 0:00:00
130.5/130.5 kB 12.6 MB/s eta 0:00:00
341.4/341.4 kB 32.8 MB/s eta 0:00:00
3.4/3.4 MB 77.4 MB/s eta 0:00:00
1.2/1.2 MB 56.7 MB/s eta 0:00:00
130.2/130.2 kB 13.2 MB/s eta 0:00:00

LLM-Finetuning with PEFT

```
Building wheel for rouge_score (setup.py) ... done
ERROR: pip's dependency resolver does not currently take into account all the packages
that are installed. This behaviour is the source of the following dependency conflicts.
cudf-cu12 24.4.1 requires pyarrow<15.0.0a0,>=14.0.1, but you have pyarrow 15.0.2 which is
incompatible.
google-colab 1.0.0 requires requests==2.31.0, but you have requests 2.32.3 which is
incompatible.

# Necessary imports
import warnings

import pandas as pd
from datasets import load_dataset
from transformers import pipeline

import mlflow
from mlflow.metrics.genai import EvaluationExample, answer_correctness,
make_genai_metric

# Disable FutureWarnings
warnings.filterwarnings("ignore", category=FutureWarning)

import os
os.environ["HF_TOKEN"] = "hf_xxPuykSGuOIIJdmwMBr0mUjYSfcKRFjlWi"
```

Load a pretrained Hugging Face pipeline

Here we are loading a text generation pipeline, but you can also use either a text summarization or question answering pipeline.

```
mpt_pipeline = pipeline("text-generation", model="mosaicml/mpt-7b-chat")

/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab
(https://huggingface.co/settings/tokens), set it as secret in your Google Colab and
restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models
or datasets.
warnings.warn(
config.json:  0%| 0.00/1.23k [00:00<?, ?B/s]
pytorch_model.bin.index.json:  0%| 0.00/16.0k [00:00<?, ?B/s]
Downloading shards:  0%| 0/2 [00:00<?, ?it/s]
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
pytorch_model-00001-of-00002.bin: 0% | 0.00/9.94G [00:00<?, ?B/s]
pytorch_model-00002-of-00002.bin: 0% | 0.00/3.36G [00:00<?, ?B/s]
Loading checkpoint shards: 0% | 0/2 [00:00<?, ?it/s]
```

Log the model using MLflow

We log our pipeline as an MLflow Model, which follows a standard format that lets you save a model in different "flavors" that can be understood by different downstream tools. In this case, the model is of the transformers "flavor".

```
mlflow.set_experiment("Evaluate Hugging Face Text Pipeline")

# Define the signature
signature = mlflow.models.infer_signature(
    model_input="What are the three primary colors?",
    model_output="The three primary colors are red, yellow, and blue.",
)

# Log the model using mlflow
with mlflow.start_run():
    model_info = mlflow.transformers.log_model(
        transformers_model=mpt_pipeline,
        artifact_path="mpt-7b",
        signature=signature,
        registered_model_name="mpt-7b-chat",
    )
```

Load Evaluation Data

Load in a dataset from Hugging Face Hub to use for evaluation.

The data fields in the dataset below represent:

- **instruction**: Describes the task that the model should perform. Each row within the dataset is a unique instruction (task) to be performed.
- **input**: Optional contextual information that relates to the task defined in the **instruction** field. For example, for the instruction "Identify the odd one out", the **input** contextual guidance is given as the list of items to select an outlier from, "Twitter, Instagram, Telegram".
- **output**: The answer to the instruction (with the optional **input** context provided) as generated by the original evaluation model (**text-davinci-003** from OpenAI)
- **text**: The final total text as a result of applying the **instruction**, **input**, and **output** to the prompt template used, which is sent to the model for fine tuning purposes.

```
dataset = load_dataset("tatsu-lab/alpaca")
eval_df = pd.DataFrame(dataset["train"])
eval_df.head(10)
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

Output:

	Instruction	input	output	text
0	Give three tips for staying healthy.		1.Eat a balanced diet and below is an instruction Make sure to include that describes a task	
1	What are the three primary colors?		An atom is made up of a nucleus, which contain... There are a number of ways to reduce air pollu...	Below is an instruction that describes a task
2	Describe the structure of an atom.		I had to make a difficult decision when I was ...	Below is an instruction that describes a task
3	How can we reduce air pollution?			Below is an instruction that describes a task
4	Describe a time when you had to make a difficult decision.			Below is an instruction that describes a task
5	Identify the odd one out.	Twitter, instagram, Telegram	Telegram	Below is an instruction that describes a task
6	Explain why the following fraction is equivalent...	4/16	The fraction 4/16 is equivalent to 1/4 because... John was at a crossroads in his life. He had j...	Below is an instruction that describes a task
7	Write a short story in third person narration ...		<nooutput> This type of instruction cannot be ...	Below is an instruction that describes a task
8	Render a 3D model of a house			Below is an instruction that describes a task
9	Evaluate this sentence for spelling and grammar...	He finished his meal and left the restaurent	He finished his meal and left the restaurant	Below is an instruction that describes a task

Define Metrics

Since we are evaluating how well our model can provide an answer to a given instruction, we may want to choose some metrics to help measure this on top of any builtin metrics that mlflow.evaluate() gives us.

Let's measure how well our model is doing on the following two metrics:

- **Is the answer correct?** Let's use the predefined metric answer_correctness here.
- **Is the answer fluent, clear, and concise?** We will define a custom metric answer_quality to measure this.

We will need to pass both of these into the extra_metrics argument for mlflow.evaluate() in order to assess the quality of our model.

What is an Evaluation Metric?

An evaluation metric encapsulates any quantitative or qualitative measure you want to calculate for your model. For each model type, `mlflow.evaluate()` will automatically calculate some set of builtin metrics. Refer here for which builtin metrics will be calculated for each model type. You can also pass in any other metrics you want to calculate as extra metrics. MLflow provides a set of predefined metrics that you can find here, or you can define your own custom metrics. In the example here, we will use the combination of predefined metrics `mlflow.metrics.genai.answer_correctness` and a custom metric for the quality evaluation.

Let's load our predefined metrics - in this case we are using `answer_correctness` with GPT-4.

```
answer_correctness_metric = answer_correctness(model="openai:/gpt-4")
```

Now we want to create a custom LLM-judged metric named `answer_quality` using `make_genai_metric()`. We need to define a metric definition and grading rubric, as well as some examples for the LLM judge to use.

```
# The definition explains what "answer quality" entails
answer_quality_definition = """Please evaluate answer quality for the provided output on
the following criteria:
fluency, clarity, and conciseness. Each of the criteria is defined as follows:
- Fluency measures how naturally and smooth the output reads.
- Clarity measures how understandable the output is.
- Conciseness measures the brevity and efficiency of the output without compromising
meaning.
The more fluent, clear, and concise a text, the higher the score it deserves.
"""

# The grading prompt explains what each possible score means
answer_quality_grading_prompt = """Answer quality: Below are the details for different
scores:
- Score 1: The output is entirely incomprehensible and cannot be read.
- Score 2: The output conveys some meaning, but needs lots of improvement in to improve
fluency, clarity, and conciseness.
- Score 3: The output is understandable but still needs improvement.
- Score 4: The output performs well on two of fluency, clarity, and conciseness, but
could be improved on one of these criteria.
- Score 5: The output reads smoothly, is easy to understand, and clear. There is no
clear way to improve the output on these criteria.
"""

# We provide an example of a "bad" output
example1 = EvaluationExample(
    input="What is MLflow?",
    output="MLflow is an open-source platform. For managing machine learning workflows,
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
it "
    "including experiment tracking model packaging versioning and deployment as well as a
platform "
    "simplifying for on the ML lifecycle.",
    score=2,
    justification="The output is difficult to understand and demonstrates extremely low
clarity. "
    "However, it still conveys some meaning so this output deserves a score of 2.",
)
# We also provide an example of a "good" output
example2 = EvaluationExample(
    input="What is MLflow?",
    output="MLflow is an open-source platform for managing machine learning workflows,
including "
    "experiment tracking, model packaging, versioning, and deployment.",
    score=5,
    justification="The output is easily understandable, clear, and concise. It deserves a
score of 5.",
)
answer_quality_metric = make_genai_metric(
    name="answer_quality",
    definition=answer_quality_definition,
    grading_prompt=answer_quality_grading_prompt,
    version="v1",
    examples=[example1, example2],
    model="openai:/gpt-4",
    greater_is_better=True,
)
```

Evaluate

We need to set our OpenAI API key, since we are using GPT-4 for our LLM-judged metrics.

In order to set your private key safely, please be sure to either export your key through a command-line terminal for your current instance, or, for a permanent addition to all user-based sessions, configure your favored environment management configuration file (i.e., .bashrc, .zshrc) to have the following entry:

```
OPENAI_API_KEY=<your openai API key>
```

Now, we can call `mlflow.evaluate()`. Just to test it out, let's use the first 10 rows of the data. Using the "text" model type, toxicity and readability metrics are calculated as builtin metrics. We also pass in the two metrics we defined above into the `extra_metrics` parameter to be evaluated.

```
with mlflow.start_run():
    results = mlflow.evaluate(
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
model_info.model_uri,  
eval_df.head(10),  
evaluators="default",  
model_type="text",  
targets="output",  
extra_metrics=[answer_correctness_metric, answer_quality_metric],  
evaluator_config={"col_mapping": {"inputs": "instruction"}},  
)  
  
Downloading artifacts: 0% | 0/79 [00:00<?, ?it/s]  
2023/12/28 11:57:30 INFO mlflow.store.artifact.artifact_repo: The progress bar can be  
disabled by setting the environment variable MLFLOW_ENABLE_ARTIFACTS_PROGRESS_BAR to  
false  
  
Loading checkpoint shards: 0% | 0/66 [00:00<?, ?it/s]  
2023/12/28 12:00:25 INFO mlflow.models.evaluation.base: Evaluating the model with the  
default evaluator.  
2023/12/28 12:00:25 INFO mlflow.models.evaluation.default_evaluator: Computing model  
predictions.  
2023/12/28 12:02:23 INFO mlflow.models.evaluation.default_evaluator: Testing metrics on  
first row...  
Using default facebook/roberta-hate-speech-dynabench-r4-target checkpoint  
  
0% | 0/1 [00:00<?, ?it/s]  
0% | 0/1 [00:00<?, ?it/s]  
2023/12/28 12:02:43 INFO mlflow.models.evaluation.default_evaluator: Evaluating builtin  
metrics: token_count  
2023/12/28 12:02:43 INFO mlflow.models.evaluation.default_evaluator: Evaluating builtin  
metrics: toxicity  
2023/12/28 12:02:44 INFO mlflow.models.evaluation.default_evaluator: Evaluating builtin  
metrics: flesch_kincaid_grade_level  
2023/12/28 12:02:44 INFO mlflow.models.evaluation.default_evaluator: Evaluating builtin  
metrics: ari_grade_level  
2023/12/28 12:02:44 INFO mlflow.models.evaluation.default_evaluator: Evaluating metrics:  
answer_correctness  
  
0% | 0/10 [00:00<?, ?it/s]  
2023/12/28 12:02:53 INFO mlflow.models.evaluation.default_evaluator: Evaluating metrics:  
answer_quality  
  
0% | 0/10 [00:00<?, ?it/s]
```

View results

`results.metrics` is a dictionary with the aggregate values for all the metrics calculated. Refer [here](#) for details on the builtin metrics for each model type.

```
results.metrics
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
{'toxicity/v1/mean': 0.00809656630299287,  
 'toxicity/v1/variance': 0.0004603014839856817,  
 'toxicity/v1/p90': 0.010559113975614286,  
 'toxicity/v1/ratio': 0.0,  
 'flesch_kincaid_grade_level/v1/mean': 4.9,  
 'flesch_kincaid_grade_level/v1/variance': 6.350000000000005,  
 'flesch_kincaid_grade_level/v1/p90': 6.829999999999998,  
 'ari_grade_level/v1/mean': 4.189999999999995,  
 'ari_grade_level/v1/variance': 16.6329,  
 'ari_grade_level/v1/p90': 7.949999999999998,  
 'answer_correctness/v1/mean': 1.5,  
 'answer_correctness/v1/variance': 1.45,  
 'answer_correctness/v1/p90': 2.299999999999999,  
 'answer_quality/v1/mean': 2.4,  
 'answer_quality/v1/variance': 1.44,  
 'answer_quality/v1/p90': 4.1}
```

We can also view the `eval_results_table`, which shows us the metrics for each row of data.

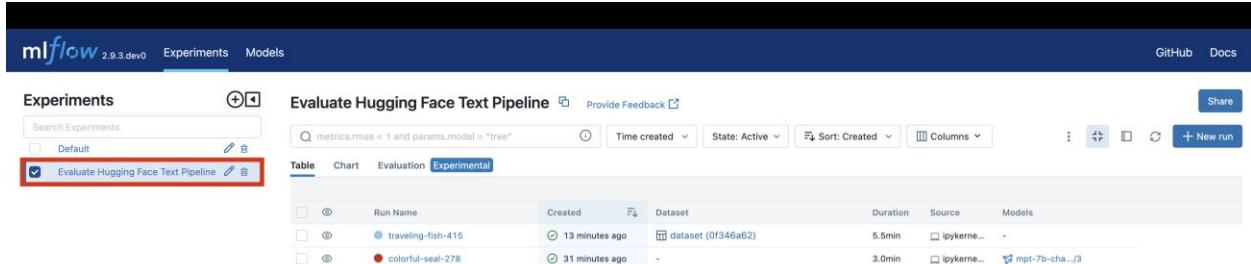
```
results.tables["eval_results_table"]
```

```
Downloading artifacts:  0% | 0/1 [00:00<?, ?it/s]
```

View results in UI

Finally, we can view our evaluation results in the MLflow UI. We can select our experiment on the left sidebar, which will bring us to the following page. We can see that one run logged our model "mpt-7b-chat", and the other run has the dataset we evaluated.

LLM-Finetuning with PEFT



The screenshot shows the mlflow UI interface. At the top, there's a navigation bar with 'mlflow 2.9.3.dev0', 'Experiments', 'Models', 'GitHub', and 'Docs'. Below the navigation bar is a search bar with the query 'metrics.rmse < 1 and params.model = "tree"'. To the right of the search bar are filters for 'Time created', 'State: Active', 'Sort: Created', and 'Columns'. There are also icons for filtering, sorting, and column selection. A 'Share' button is located in the top right corner. The main area is titled 'Evaluate Hugging Face Text Pipeline' with a 'Provide Feedback' link. Below the title is a table header with columns: Run Name, Created, Dataset, Duration, Source, and Models. The table contains three rows:

	Run Name	Created	Dataset	Duration	Source	Models
<input type="checkbox"/>	traveling-fish-415	13 minutes ago	dataset (0f346a62)	5.5min	ipykernel...	-
<input type="checkbox"/>	colorful-seal-278	31 minutes ago	-	3.0min	ipykernel...	mpt-7b-chat-/3

We click on the Evaluation tab and hide any irrelevant runs.

By: Tarun S Gowda

LLM-Finetuning with PEFT

The screenshot shows the mlflow UI interface for an "Evaluate Hugging Face Text Pipeline" experiment. The top navigation bar includes links for Experiments and Models, along with GitHub and Docs buttons. The main page displays experimental runs and their results.

Experiments: A sidebar on the left allows searching for experiments and selecting a default run. The current run selected is "Evaluate Hugging Face Text Pipeline".

Evaluation Results: The main content area shows the results of two runs: "traveling-fish-415" (blue dot) and "colorful-seal-278" (red dot). The results are presented in a table format:

instruction	output	
Give three tips for staying healthy.	1.Eat a balanced diet and make sure to include plenty fruits and vegetables. 2. Exercise regularly to keep your body active and strong. 3. Get enough sleep and maintain a consistent sleep schedule.	(empty)
What are the three primary colors?	The three primary colors are red, blue, and yellow.	(empty)
Describe the structure of an atom.	An atom is made up of a nucleus, which contains protons and neutrons, surrounded by electrons that travel in orbits around the nucleus. The protons and neutrons have a positive charge, while the electrons have a negative charge, resulting in an overall neutral atom. The number of each particle determines the atomic number and the type of atom.	(empty)

Filtering and Grouping: At the top of the results table, there are filters for "Run Name" and "Table: eval_results_table.json". Below the table, there are dropdown menus for "Group by: instruction" and "Compare: output".

We can now choose what columns we want to group by, as well as which column we want to compare. In the following example, we are looking at the score for answer correctness for each input-output pair, but we could choose any other metric to compare.

LLM-Finetuning with PEFT

[mlflow](#) 2.9.3.dev0 Experiments Models GitHub Docs

Evaluate Hugging Face Text Pipeline [Provide Feedback](#)

Experiment ID: 459698453446712789 Artifact Location: mlflow-artifacts:/459698453446712789

Description Edit

metrics.rmse < 1 and params.model = "tree" Time created State: Active Sort: Created

Table Chart Evaluation Experimental

Table: eval_results_table.json

Group by: instruction +3 Compare: answer_correctness/v1/score

instruction	output	answer_correctness/v1/score
Give three tips for staying healthy.	1.Eat a balanced diet make sure to include fruits and vegetables. Exercise regularly to keep your body active. 3. Get enough sleep to maintain a consistent mood.	The output provided by the model only includes one tip for staying healthy, which is to eat a balanced diet. This aligns with the first target provided. However, the output does not mention the other two targets. Therefore, the score is 2.
What are the three primary colors?	The three primary colors are red, blue, and yellow.	The output provided by the model is completely correct. It accurately identifies the three primary colors as red, blue, and yellow, which aligns perfectly with the provided targets. Therefore, the score is 5.
Describe the structure of an atom.	An atom is made up of a nucleus, which contains protons and neutrons, surrounded by electrons that travel in orbits around the nucleus. The protons and neutrons have a positive...	The output provided by the model is incomplete and does not provide any information about the structure of an atom. It does not mention the nucleus, protons, neutrons, or electrons, which are key...

Group by: instruction +3 Compare: answer_correctness/v1/score

instruction

input

text

output

outputs

token_count

toxicity/v1/score

flesch_kincaid_grade_level/v1/score

ari_grade_level/v1/score

answer_correctness/v1/score

answer_correctness/v1/justification

answer_quality/v1/score

answer_quality/v1/justification

traveling-fish-415

dataset (0f346a62)

answer_correctness/v1/jus...

The output provided by the model only includes one tip for staying healthy, which is to eat a balanced diet. This aligns with the first target provided. However, the output does not mention the other two targets. Therefore, the score is 2.

answer_correctness/v1/justification

The output provided by the model is completely correct. It accurately identifies the three primary colors as red, blue, and yellow, which aligns perfectly with the provided targets. Therefore, the score is 5.

answer_correctness/v1/score

The output provided by the model is incomplete and does not provide any information about the structure of an atom. It does not mention the nucleus, protons, neutrons, or electrons, which are key...

answer_correctness/v1/justification

The output provided by the model is incomplete and does not provide any information about the structure of an atom. It does not mention the nucleus, protons, neutrons, or electrons, which are key...

Finally, we get to the following view, where we can see the justification and score for answer correctness for each row.

By: Tarun S Gowda

LLM-Finetuning with PEFT

The screenshot shows the mlflow web interface at version 2.9.3.dev0. The top navigation bar includes links for 'Experiments' and 'Models', and buttons for 'GitHub' and 'Docs'. The main page title is 'Evaluate Hugging Face Text Pipeline' with a 'Provide Feedback' link. Below the title, it displays 'Experiment ID: 459698453446712789' and 'Artifact Location: mlflow-artifacts:/459698453446712789'. A 'Description' link with an edit icon is present. The interface features a search bar and filters for 'Time created', 'State: Active', and 'Sort: Created'. A 'New run' button is located in the top right.

The central content is a table titled 'eval_results_table.json' under the 'Evaluation' tab. The table has columns: 'instruction', 'output', 'outputs', 'answer_correctness/v1/justification', and 'answer_correctness/v1/score'. The table contains three rows of data:

instruction	output	outputs	answer_correctness/v1/justification	answer_correctness/v1/score
Give three tips for staying healthy.	1.Eat a balanced diet and make sure to include plenty of fruits and vegetables. 2. Exercise regularly to keep your body active and strong. 3. Get enough sleep and maintain a consistent sleep schedule.	Give three tips for staying healthy. 1. Eat a balanced diet: A healthy diet is essential	The output provided by the model only includes one tip for staying healthy, which is to eat a balanced diet. This aligns with the first target provided. However, the output does not mention the other two tips provided in the targets, which are to exercise regularly and to get enough sleep. Therefore, the output has...	2
What are the three primary colors?	The three primary colors are red, blue, and yellow.	What are the three primary colors? The three primary colors are red, blue, and yellow.	The output provided by the model is completely correct. It accurately identifies the three primary colors as red, blue, and yellow, which aligns perfectly with the provided targets. Therefore, the model's answer_correctness score is 5 according to the grading rubric.	5
Describe the structure of an atom.	An atom is made up of a nucleus, which contains protons and neutrons, surrounded by electrons that travel in orbits around the nucleus. The protons and neutrons have a positive charge, while the electrons have a negative charge, resulting in an overall neutral atom. The number of each particle determines the atomic number and the type of atom.	Describe the structure of an atom. An atom is the basic unit of matter and consists of	The output provided by the model is incomplete and does not provide any information about the structure of an atom. It does not mention the nucleus, protons, neutrons, or electrons, which are key components of an atom according to the provided targets. Therefore, the model's...	1
How can we reduce air pollution?	There are a number of ways to reduce air pollution.	How can we reduce air pollution? There are many...	The output provided by the model is...	4

By: Tarun S Gowda