Self Hosted Embeddings

Let's load the SelfHostedEmbeddings, SelfHostedHuggingFaceEmbeddings, and SelfHostedHuggingFaceInstructEmbeddings classes.

```
from langchain.embeddings import (
    SelfHostedEmbeddings,
    SelfHostedHuggingFaceEmbeddings,
    SelfHostedHuggingFaceInstructEmbeddings,
)
import runhouse as rh
```

```
# For an on-demand A100 with GCP, Azure, or Lambda
gpu = rh.cluster(name="rh-a10x", instance_type="A100:1", use_spot=False)

# For an on-demand A10G with AWS (no single A100s on AWS)
# gpu = rh.cluster(name='rh-a10x', instance_type='g5.2xlarge', provider='aws')

# For an existing cluster
# gpu = rh.cluster(ips=['<ip of the cluster>'],
#                  ssh_creds={'ssh_user': '...',
'ssh_private_key':'<path_to_key>'},
#                  name='my-cluster')
```

```
embeddings = SelfHostedHuggingFaceEmbeddings(hardware=gpu)
```

```
text = "This is a test document."
```

```
query_result = embeddings.embed_query(text)
```

And similarly for SelfHostedHuggingFaceInstructEmbeddings:

```
embeddings = SelfHostedHuggingFaceInstructEmbeddings(hardware=gpu)
```

Now let's load an embedding model with a custom load function:

Skip to main content

```
def get_pipeline():
    from transformers import (
        AutoModelForCausalLM,
        AutoTokenizer,
        pipeline,
    )  # Must be inside the function in notebooks

    model_id = "facebook/bart-base"
    tokenizer = AutoTokenizer.from_pretrained(model_id)
    model = AutoModelForCausalLM.from_pretrained(model_id)
    return pipeline("feature-extraction", model=model, tokenizer=tokenizer)


def inference_fn(pipeline, prompt):
    # Return last hidden state of the model
    if isinstance(prompt, list):
        return [emb[0][-1] for emb in pipeline(prompt)]
    return pipeline(prompt)[0][-1]
```

```
embeddings = SelfHostedEmbeddings(
    model_load_fn=get_pipeline,
    hardware=gpu,
    model_reqs=["./", "torch", "transformers"],
    inference_fn=inference_fn,
)
```

```
query_result = embeddings.embed_query(text)
```