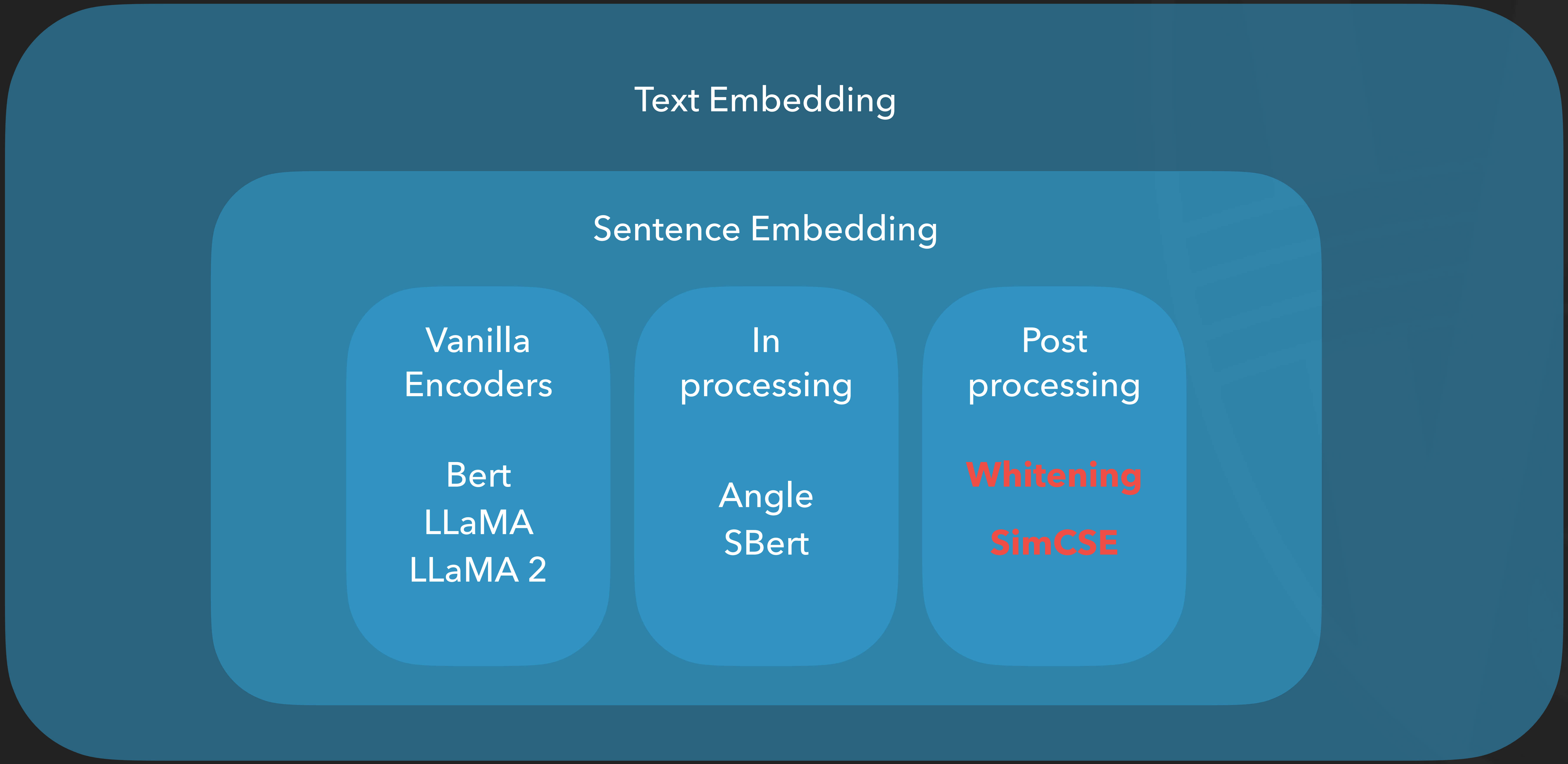
FEB 2, 2024

EFFECT OF WHITENING ON TEXT CLASSIFICATION

GOAL


Text Embedding
Sentence Embedding
Vanilla Encoders
Bert
LLaMA
LLaMA 2
In processing
Angle
SBert
Post processing
Whitening
SimCSE

WHITENING

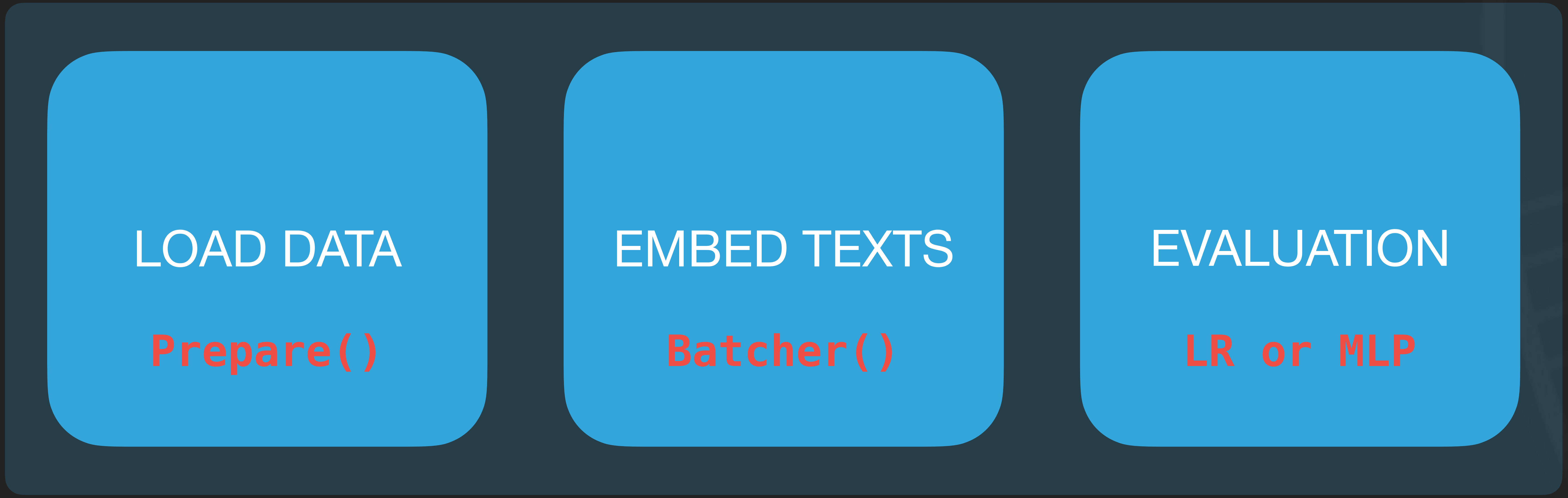
- In which scenarios does the process of whitening demonstrate effectiveness?
    - Which Data?
    - Which Tasks?
    - Which Encoders?

WHITENING

- Which Data?
  - MR, CR, STS, …
- Which Tasks?
  - Sentence Similarity, Text Classification, …
- Which Encoders?
  - LLaMA, Bert, ChatGPT, …

SENTEVAL

Pipeline of SentEval:



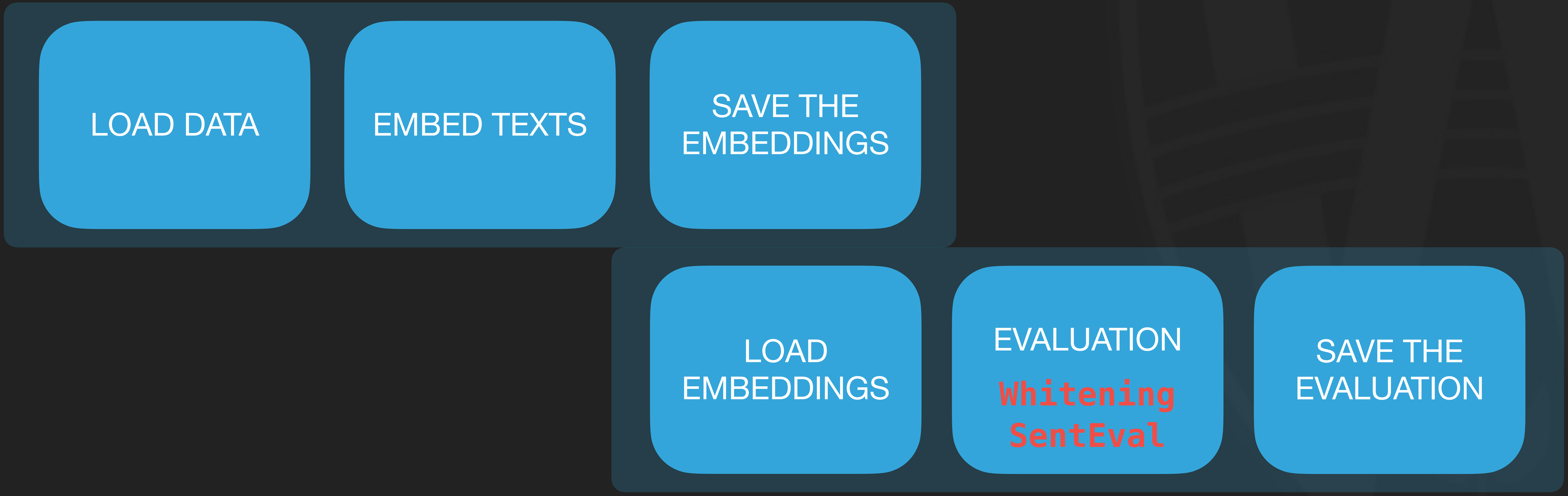
```
PARAMS = {
'TASK_PATH':PATH_TO_DATA,
'USEPYTORCH': TRUE,
'KFOLD': 10
}
```

```
PARAMS['CLASSIFIER'] = {
    'NHID': 0,
    'OPTIM': 'ADAM',
    'BATCH_SIZE': 64,
    'TENACITY': 5,
    'EPOCH_SIZE': 4
}
```

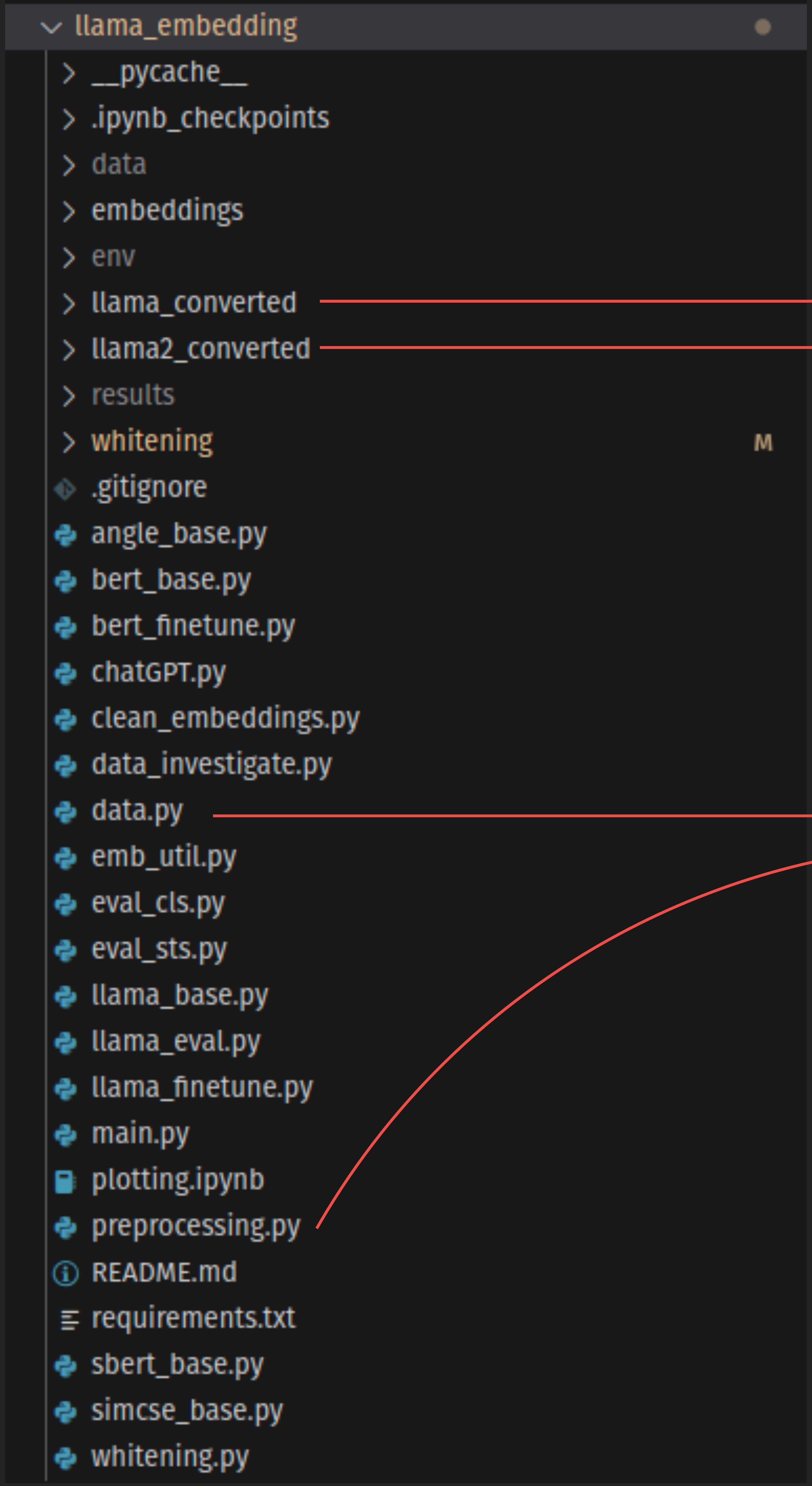
You are going to lose the embeddings in this pipeline. This could be costly for ChatGPT, LLaMA, ...

OUR PIPELINE

1) Embedding Generation



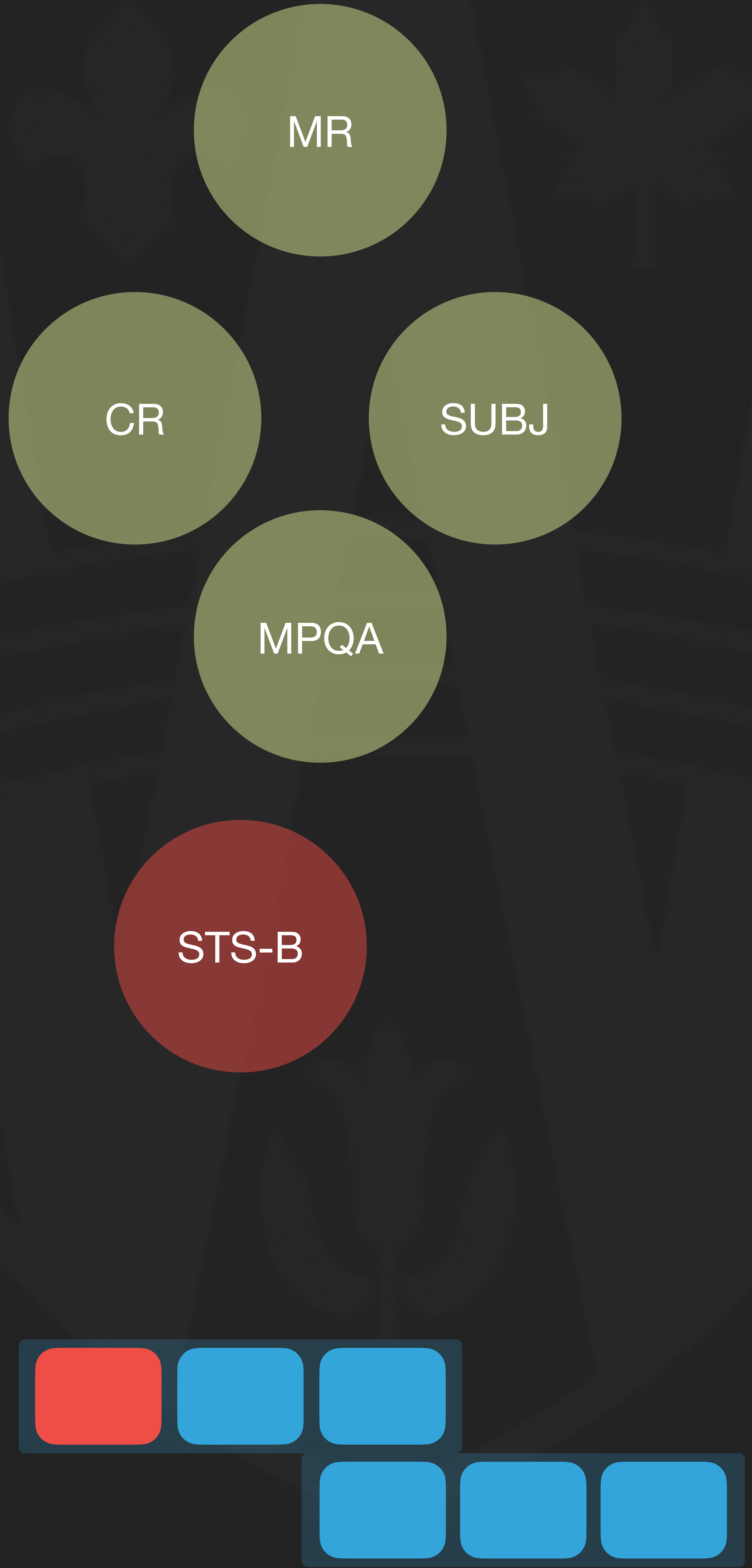
2) Embedding Evaluation

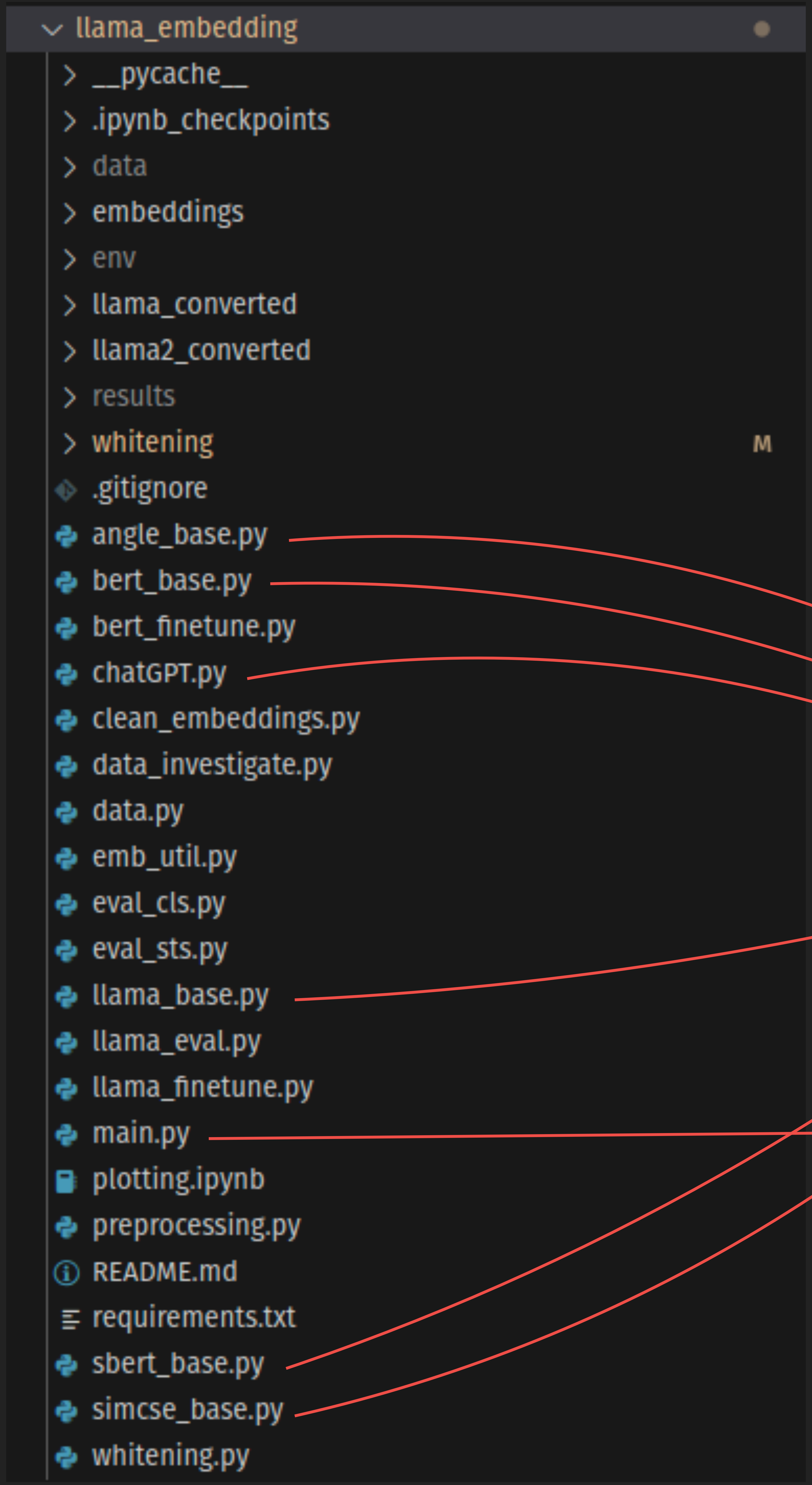


```
llama_embedding
  > __pycache__
  > .ipynb_checkpoints
  > data
  > embeddings
  > env
  > llama_converted
  > llama2_converted
  > results
  > whitening                M
  .gitignore
  angle_base.py
  bert_base.py
  bert_finetune.py
  chatGPT.py
  clean_embeddings.py
  data_investigate.py
  data.py
  emb_util.py
  eval_cls.py
  eval_sts.py
  llama_base.py
  llama_eval.py
  llama_finetune.py
  main.py
  plotting.ipynb
  preprocessing.py
  README.md
  requirements.txt
  sbert_base.py
  simcse_base.py
  whitening.py
```

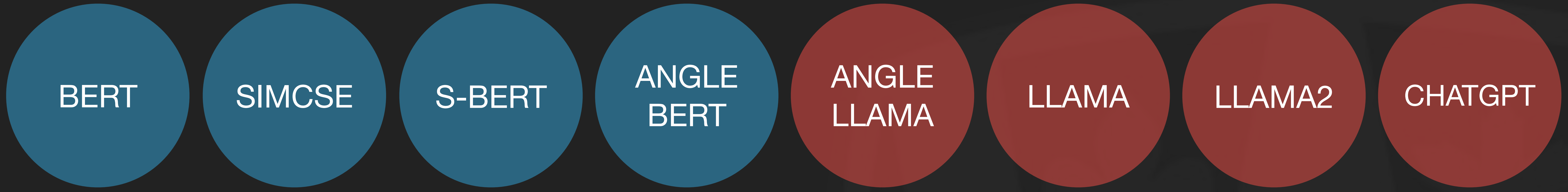
- Storing LLaMA weights

- Pre-Processing data
- Store raw data.
- Functions to clean data.
- Create HuggingFace datasets



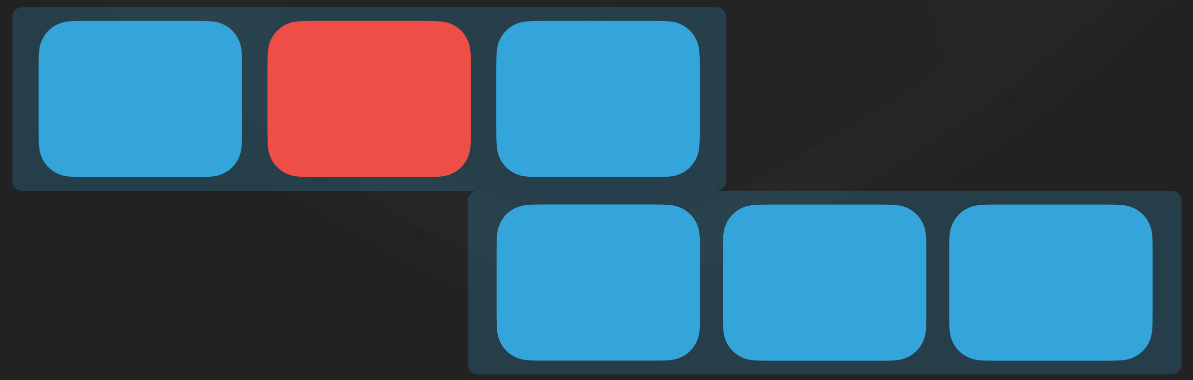


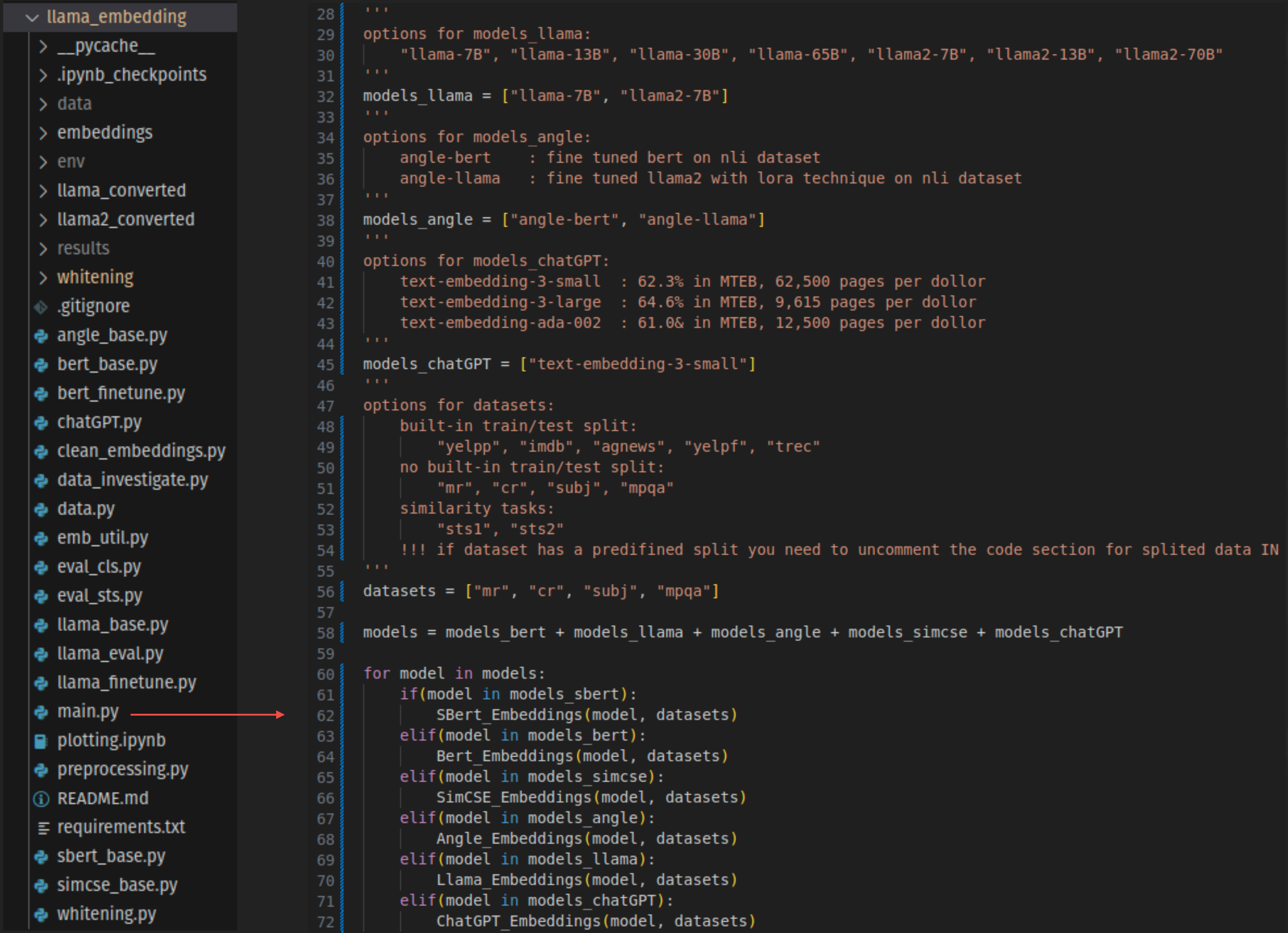
```
llama_embedding
  > __pycache__
  > .ipynb_checkpoints
  > data
  > embeddings
  > env
  > llama_converted
  > llama2_converted
  > results
  > whitening                M
  .gitignore
  angle_base.py
  bert_base.py
  bert_finetune.py
  chatGPT.py
  clean_embeddings.py
  data_investigate.py
  data.py
  emb_util.py
  eval_cls.py
  eval_sts.py
  llama_base.py
  llama_eval.py
  llama_finetune.py
  main.py
  plotting.ipynb
  preprocessing.py
  README.md
  requirements.txt
  sbert_base.py
  simcse_base.py
  whitening.py
```



- Load Models
- Manage loading weights into multiple GPUs
- Generate Embeddings
- Store Embeddings

- Loop for generating Embeddings over datasets and encoders



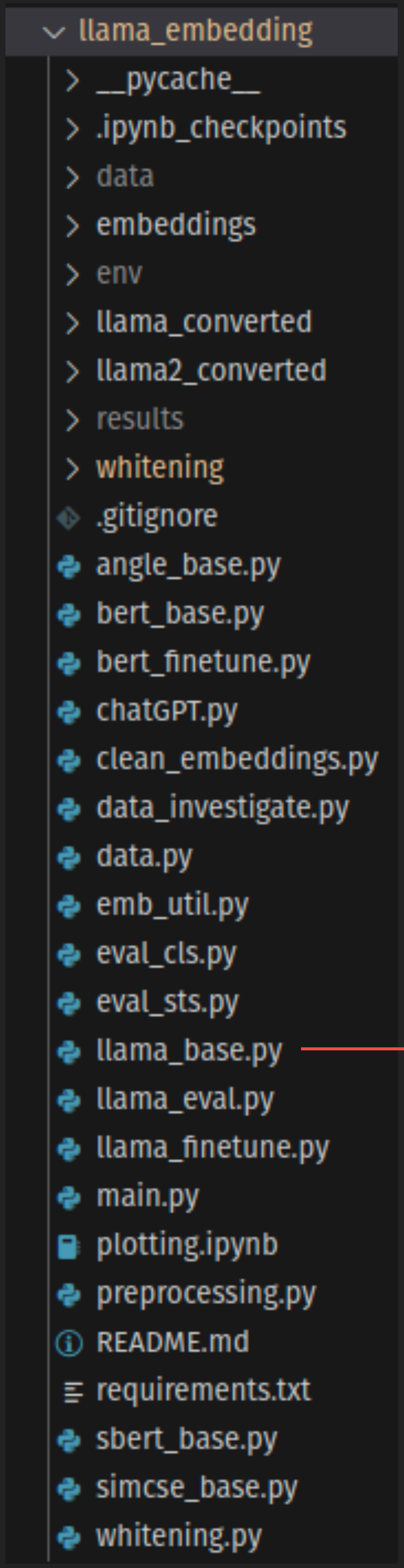


- llama_embedding
  - __pycache__
  - .ipynb_checkpoints
  - data
  - embeddings
  - env
  - llama_converted
  - llama2_converted
  - results
  - whitening
  - .gitignore
  - angle_base.py
  - bert_base.py
  - bert_finetune.py
  - chatGPT.py
  - clean_embeddings.py
  - data_investigate.py
  - data.py
  - emb_util.py
  - eval_cls.py
  - eval_sts.py
  - llama_base.py
  - llama_eval.py
  - llama_finetune.py
  - main.py
  - plotting.ipynb
  - preprocessing.py
  - README.md
  - requirements.txt
  - sbert_base.py
  - simcse_base.py
  - whitening.py

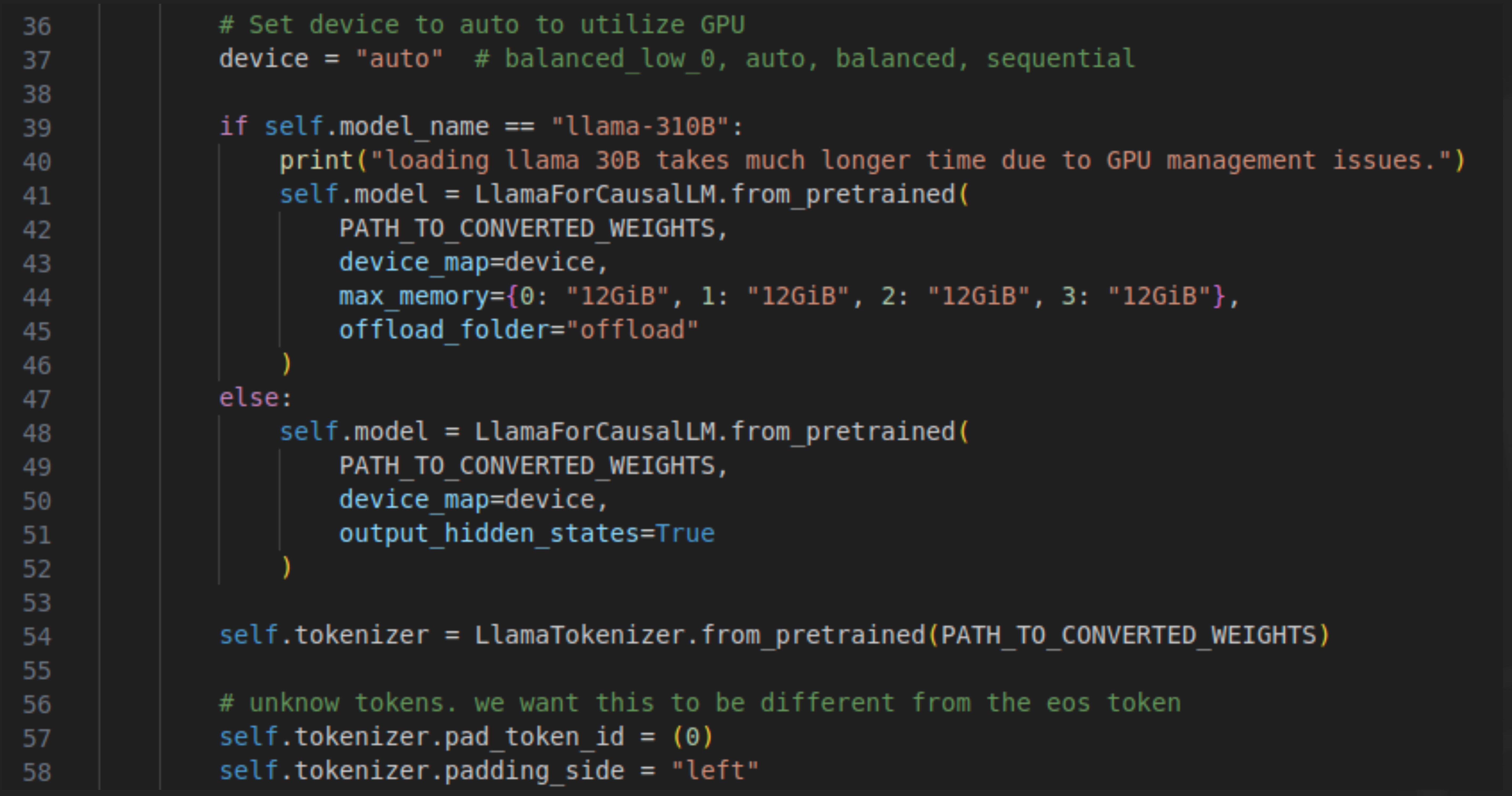
```python
'''
options for models_llama:
    "llama-7B", "llama-13B", "llama-30B", "llama-65B", "llama2-7B", "llama2-13B", "llama2-70B"
'''
models_llama = ["llama-7B", "llama2-7B"]
'''
options for models_angle:
    angle-bert    : fine tuned bert on nli dataset
    angle-llama   : fine tuned llama2 with lora technique on nli dataset
'''
models_angle = ["angle-bert", "angle-llama"]
'''
options for models_chatGPT:
    text-embedding-3-small  : 62.3% in MTEB, 62,500 pages per dollor
    text-embedding-3-large  : 64.6% in MTEB, 9,615 pages per dollor
    text-embedding-ada-002  : 61.0& in MTEB, 12,500 pages per dollor
'''
models_chatGPT = ["text-embedding-3-small"]
'''
options for datasets:
    built-in train/test split:
        "yelpp", "imdb", "agnews", "yelpf", "trec"
    no built-in train/test split:
        "mr", "cr", "subj", "mpqa"
    similarity tasks:
        "sts1", "sts2"
    !!! if dataset has a predifined split you need to uncomment the code section for splited data IN
'''
datasets = ["mr", "cr", "subj", "mpqa"]

models = models_bert + models_llama + models_angle + models_simcse + models_chatGPT

for model in models:
    if(model in models_sbert):
        SBert_Embeddings(model, datasets)
    elif(model in models_bert):
        Bert_Embeddings(model, datasets)
    elif(model in models_simcse):
        SimCSE_Embeddings(model, datasets)
    elif(model in models_angle):
        Angle_Embeddings(model, datasets)
    elif(model in models_llama):
        Llama_Embeddings(model, datasets)
    elif(model in models_chatGPT):
        ChatGPT_Embeddings(model, datasets)
```



```
∨ llama_embedding
  > __pycache__
  > .ipynb_checkpoints
  > data
  > embeddings
  > env
  > llama_converted
  > llama2_converted
  > results
  > whitening
  .gitignore
  angle_base.py
  bert_base.py
  bert_finetune.py
  chatGPT.py
  clean_embeddings.py
  data_investigate.py
  data.py
  emb_util.py
  eval_cls.py
  eval_sts.py
  llama_base.py
  llama_eval.py
  llama_finetune.py
  main.py
  plotting.ipynb
  preprocessing.py
  README.md
  requirements.txt
  sbert_base.py
  simcse_base.py
  whitening.py
```



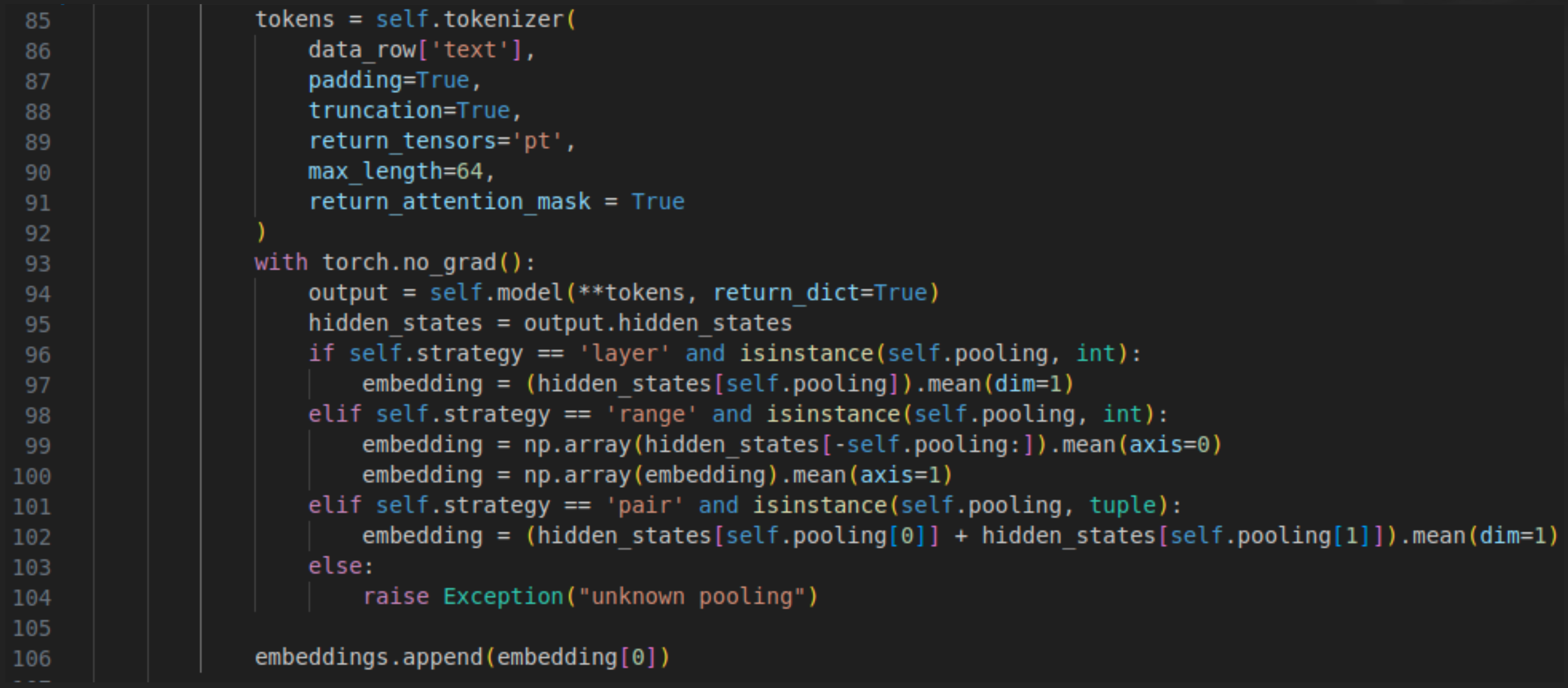
```
        # Set device to auto to utilize GPU
        device = "auto"  # balanced_low_0, auto, balanced, sequential

        if self.model_name == "llama-310B":
            print("loading llama 30B takes much longer time due to GPU management issues.")
            self.model = LlamaForCausalLM.from_pretrained(
                PATH_TO_CONVERTED_WEIGHTS,
                device_map=device,
                max_memory={0: "12GiB", 1: "12GiB", 2: "12GiB", 3: "12GiB"},
                offload_folder="offload"
            )
        else:
            self.model = LlamaForCausalLM.from_pretrained(
                PATH_TO_CONVERTED_WEIGHTS,
                device_map=device,
                output_hidden_states=True
            )

        self.tokenizer = LlamaTokenizer.from_pretrained(PATH_TO_CONVERTED_WEIGHTS)

        # unknow tokens. we want this to be different from the eos token
        self.tokenizer.pad_token_id = (0)
        self.tokenizer.padding_side = "left"
```

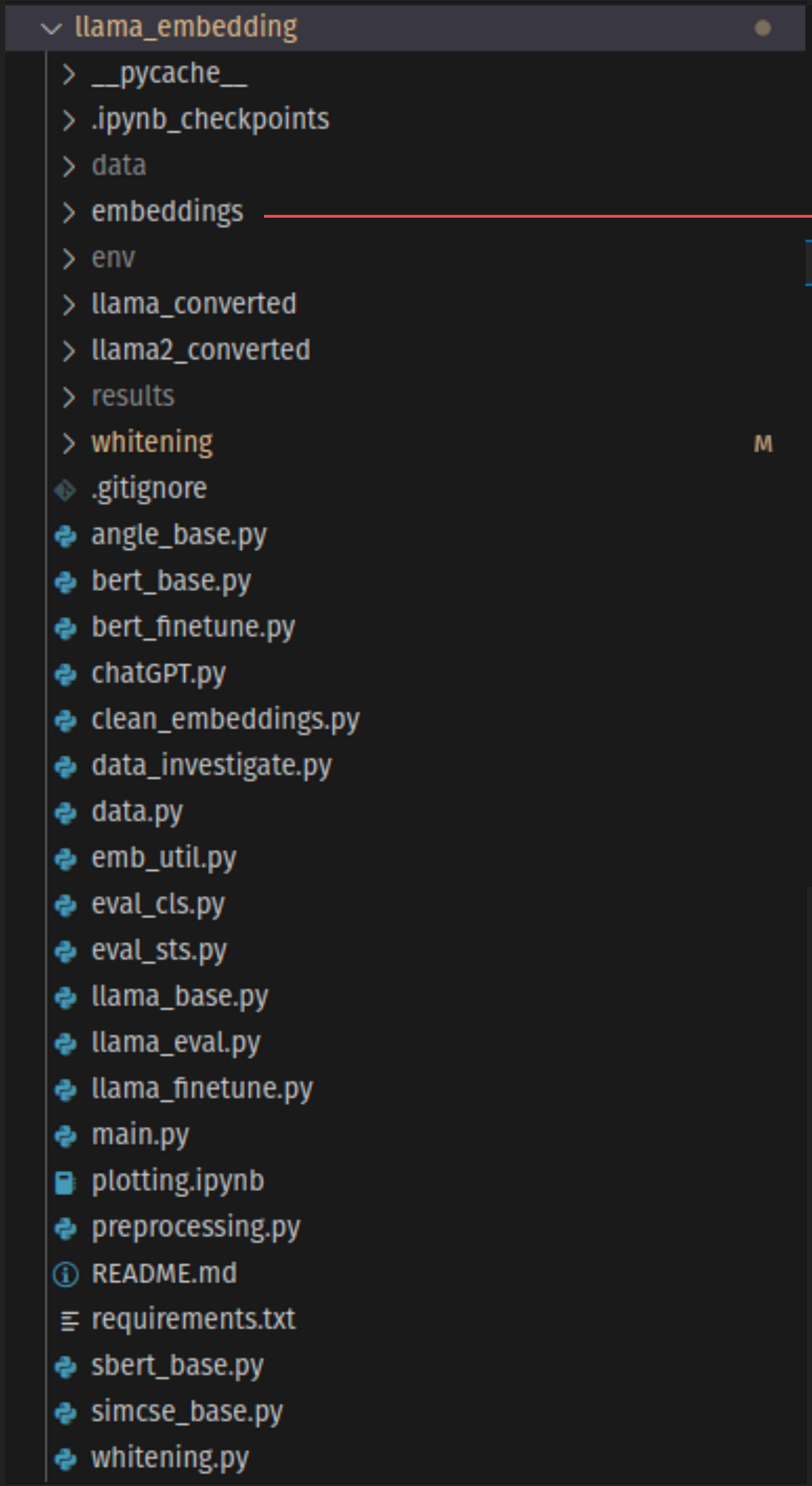
- Load the model and manage loading weights into multiple GPUs



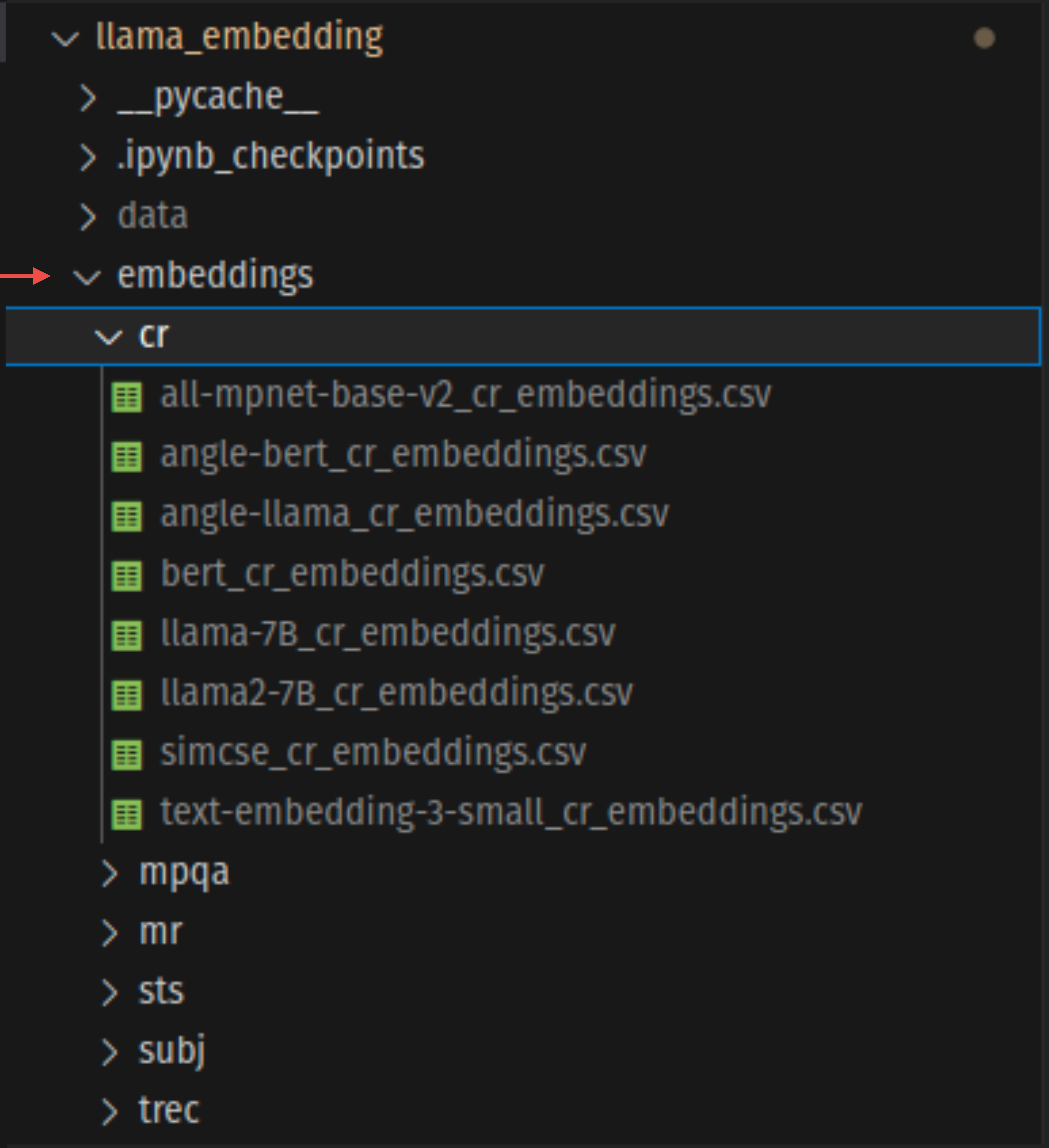
```
            tokens = self.tokenizer(
                data_row['text'],
                padding=True,
                truncation=True,
                return_tensors='pt',
                max_length=64,
                return_attention_mask = True
            )
            with torch.no_grad():
                output = self.model(**tokens, return_dict=True)
                hidden_states = output.hidden_states
                if self.strategy == 'layer' and isinstance(self.pooling, int):
                    embedding = (hidden_states[self.pooling]).mean(dim=1)
                elif self.strategy == 'range' and isinstance(self.pooling, int):
                    embedding = np.array(hidden_states[-self.pooling:]).mean(axis=0)
                    embedding = np.array(embedding).mean(axis=1)
                elif self.strategy == 'pair' and isinstance(self.pooling, tuple):
                    embedding = (hidden_states[self.pooling[0]] + hidden_states[self.pooling[1]]).mean(dim=1)
                else:
                    raise Exception("unknown pooling")

            embeddings.append(embedding[0])
```

- Embedding generation with different pooling strategies

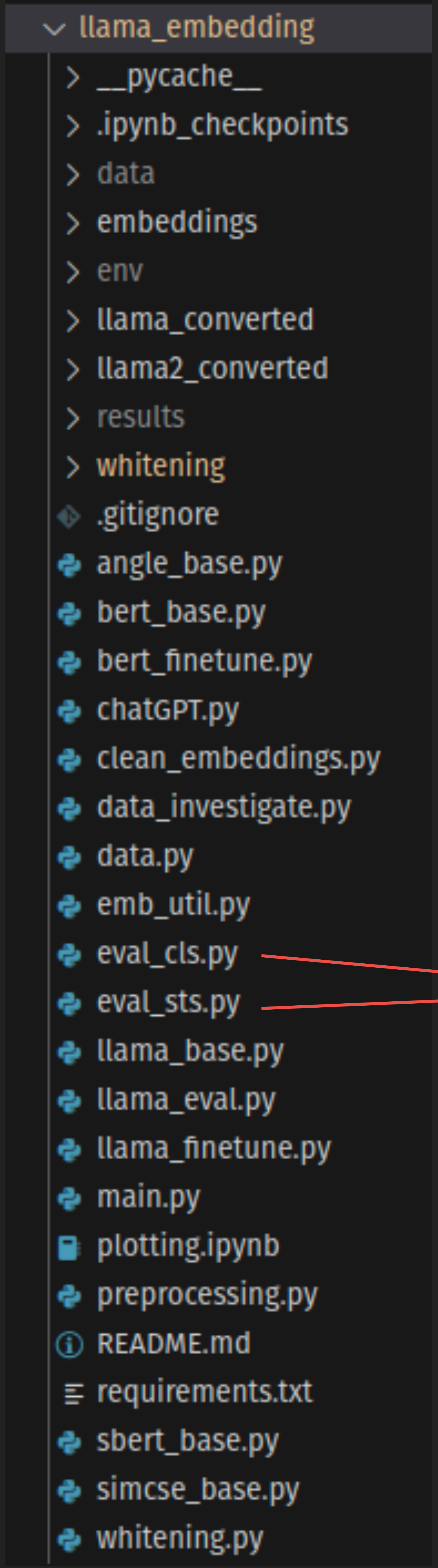


```
llama_embedding
  > __pycache__
  > .ipynb_checkpoints
  > data
  > embeddings
  > env
  > llama_converted
  > llama2_converted
  > results
  > whitening                M
  .gitignore
  angle_base.py
  bert_base.py
  bert_finetune.py
  chatGPT.py
  clean_embeddings.py
  data_investigate.py
  data.py
  emb_util.py
  eval_cls.py
  eval_sts.py
  llama_base.py
  llama_eval.py
  llama_finetune.py
  main.py
  plotting.ipynb
  preprocessing.py
  README.md
  requirements.txt
  sbert_base.py
  simcse_base.py
  whitening.py
```

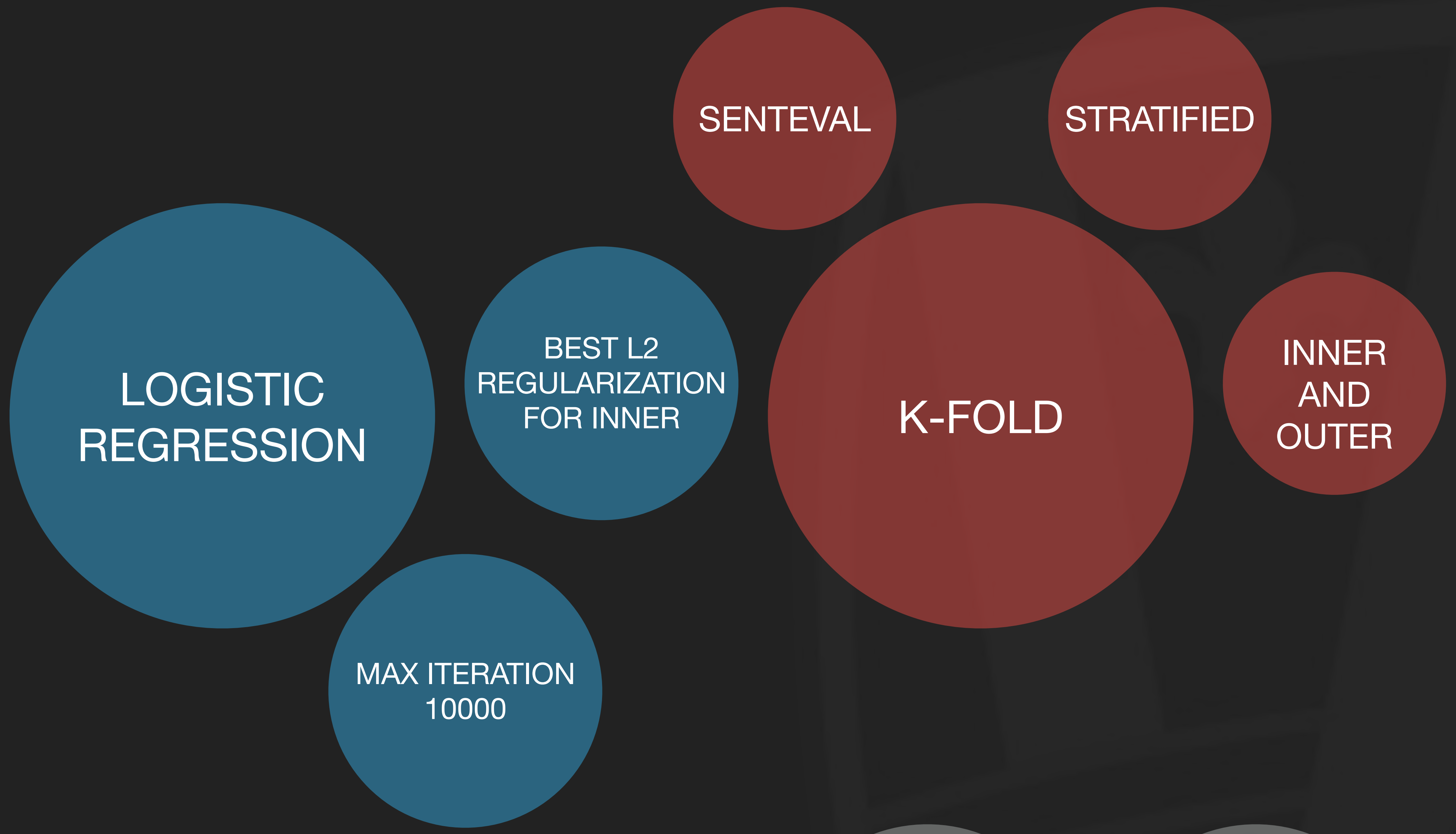


```
llama_embedding
  > __pycache__
  > .ipynb_checkpoints
  > data
  v embeddings
    v cr
      all-mpnet-base-v2_cr_embeddings.csv
      angle-bert_cr_embeddings.csv
      angle-llama_cr_embeddings.csv
      bert_cr_embeddings.csv
      llama-7B_cr_embeddings.csv
      llama2-7B_cr_embeddings.csv
      simcse_cr_embeddings.csv
      text-embedding-3-small_cr_embeddings.csv
    > mpqa
    > mr
    > sts
    > subj
    > trec
```

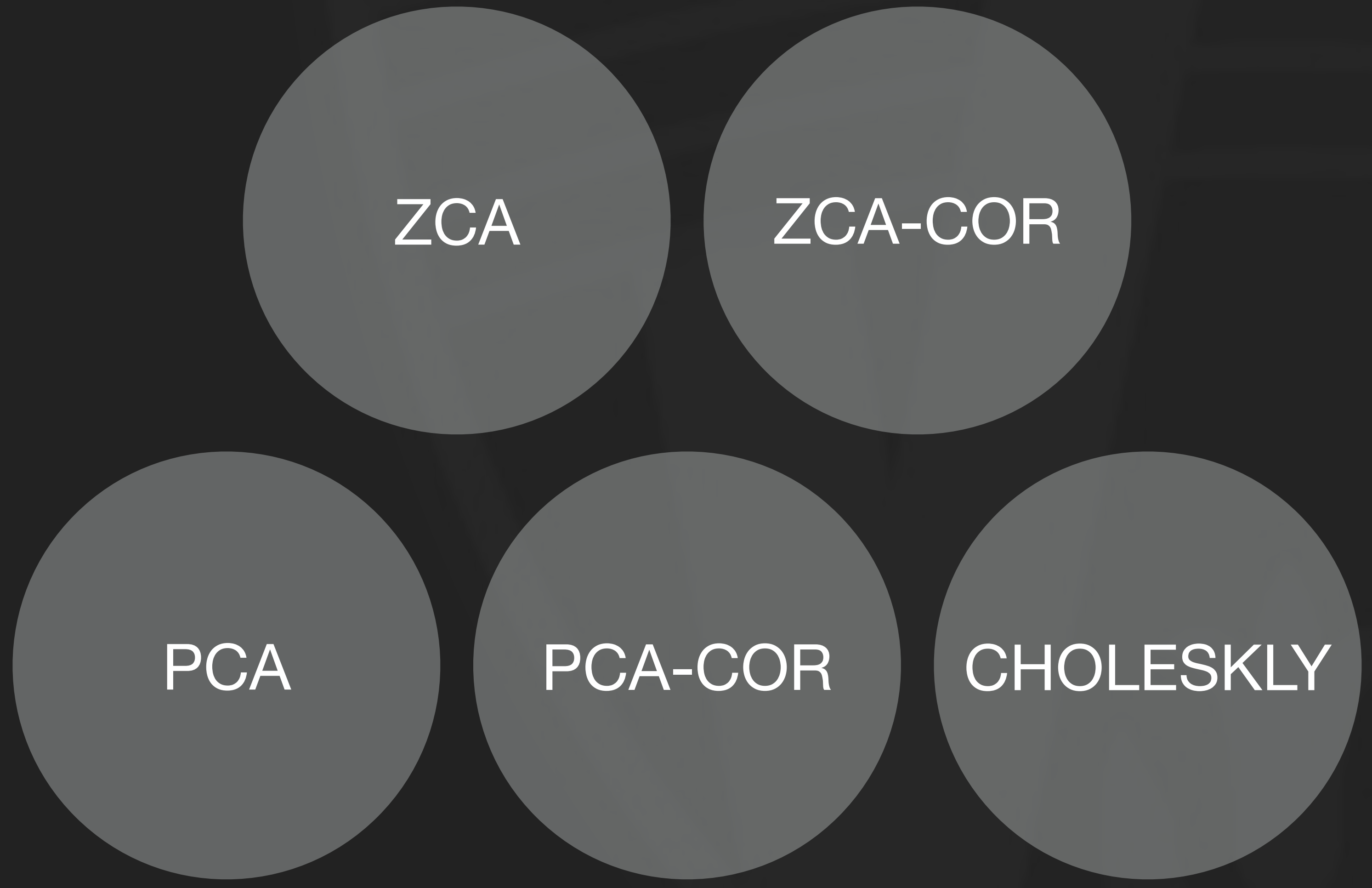
- Storing embeddings of different datasets and models

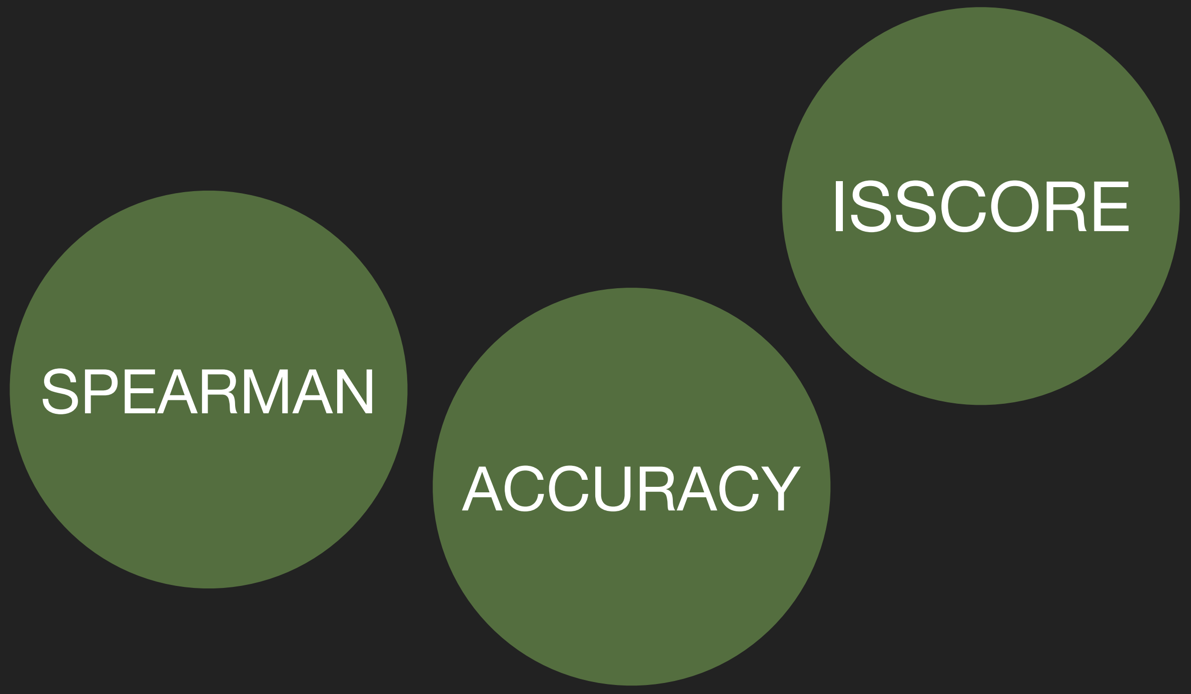


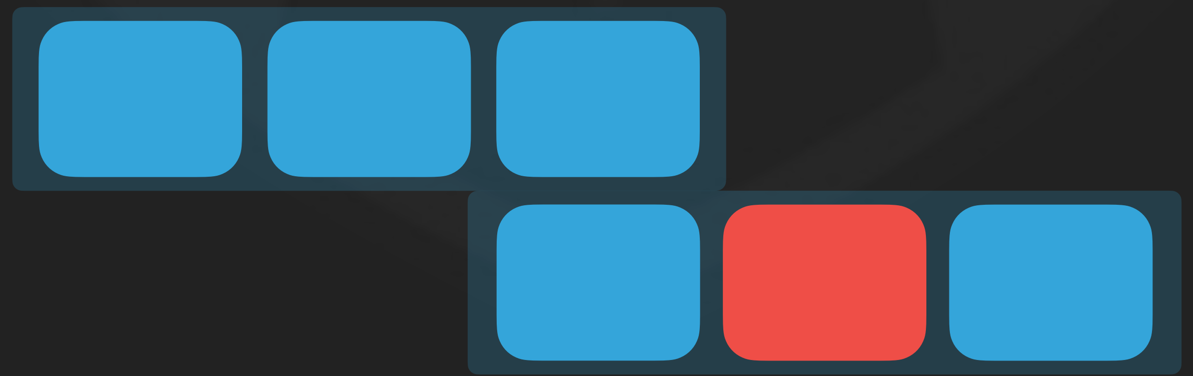
```
llama_embedding
  > __pycache__
  > .ipynb_checkpoints
  > data
  > embeddings
  > env
  > llama_converted
  > llama2_converted
  > results
  > whitening
  .gitignore
  angle_base.py
  bert_base.py
  bert_finetune.py
  chatGPT.py
  clean_embeddings.py
  data_investigate.py
  data.py
  emb_util.py
  eval_cls.py
  eval_sts.py
  llama_base.py
  llama_eval.py
  llama_finetune.py
  main.py
  plotting.ipynb
  preprocessing.py
  README.md
  requirements.txt
  sbert_base.py
  simcse_base.py
  whitening.py
```



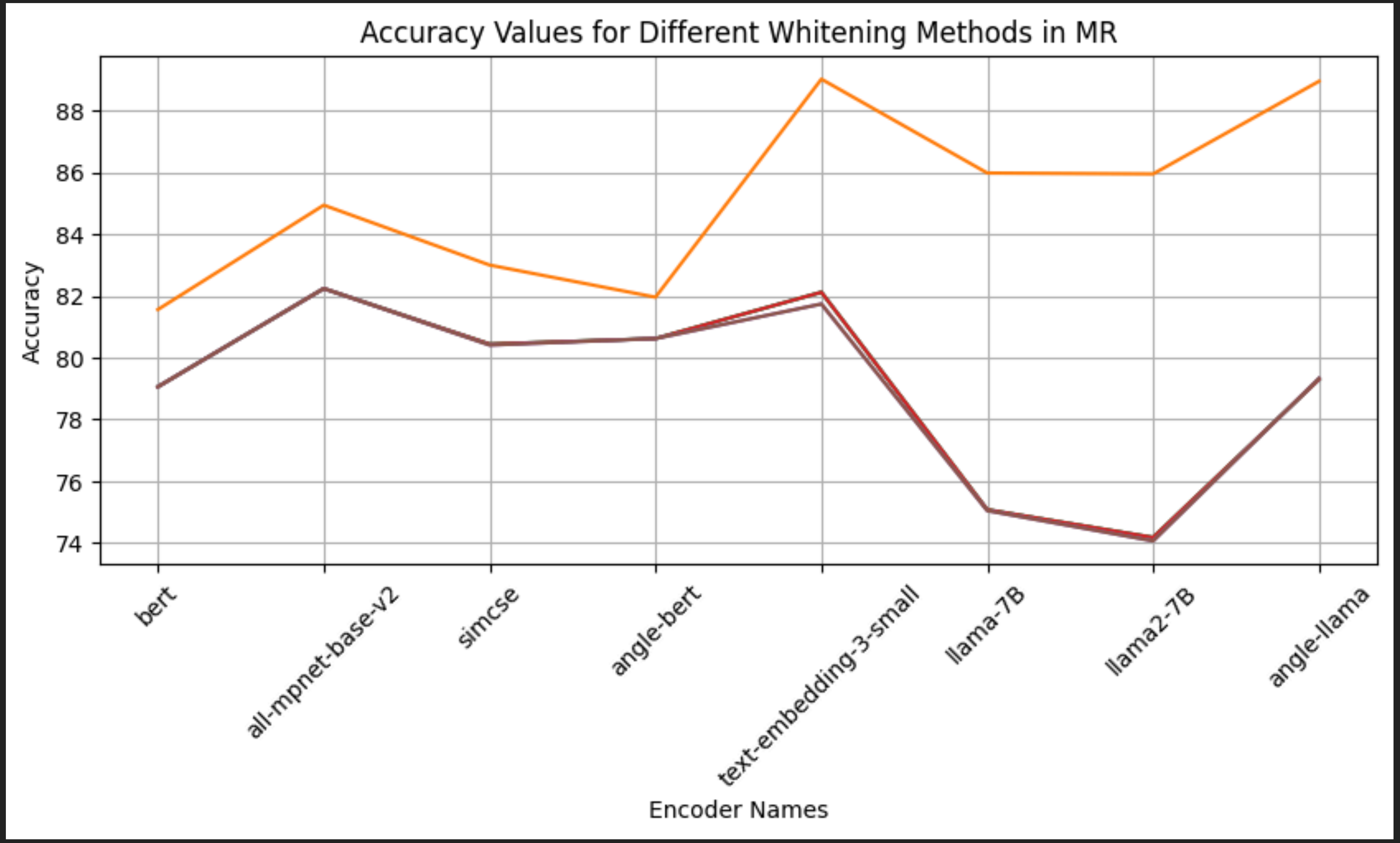
- **Evaluation** Code from SentEval
- **Whitening** Code from Remlok Github
- **Save results**

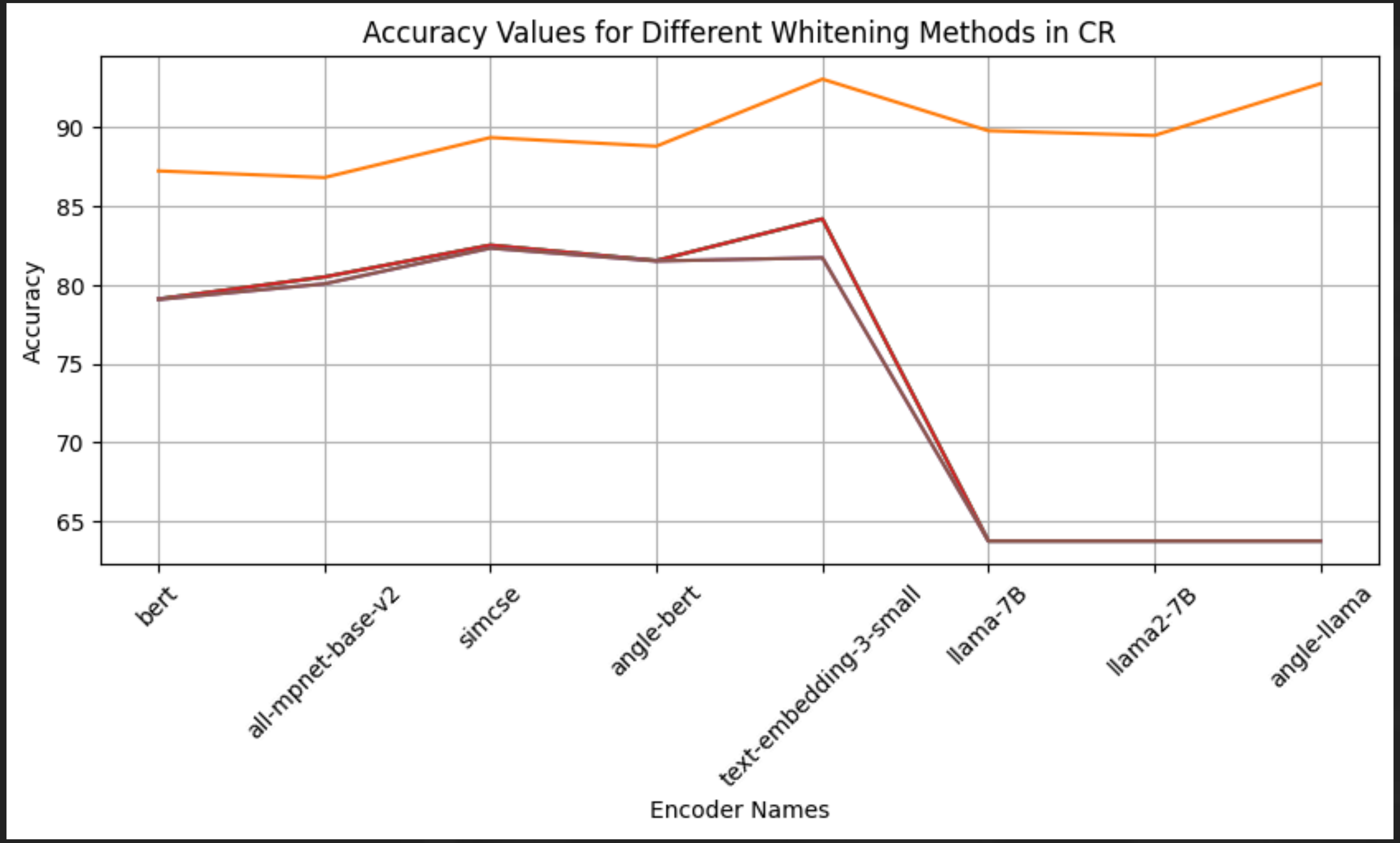


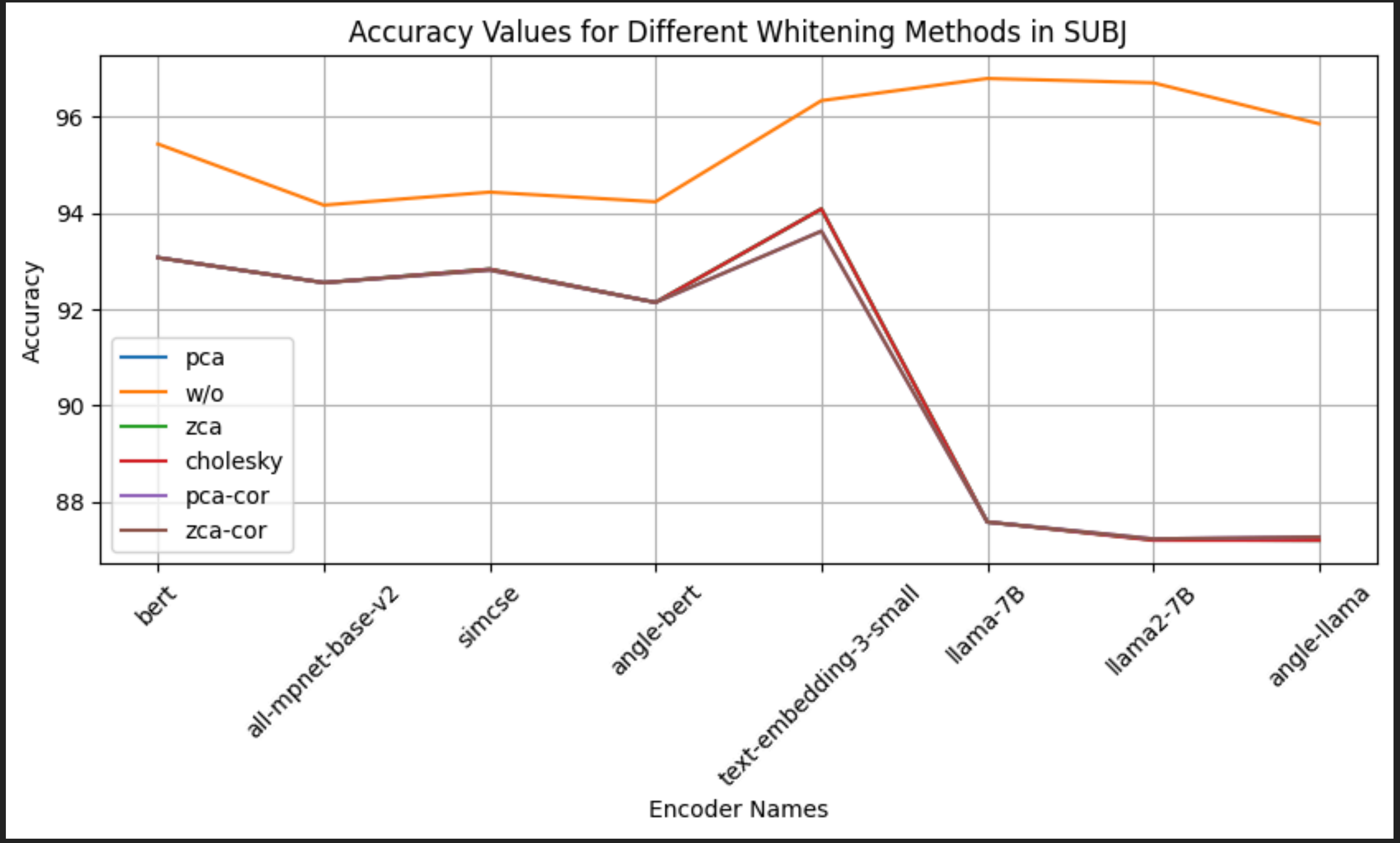


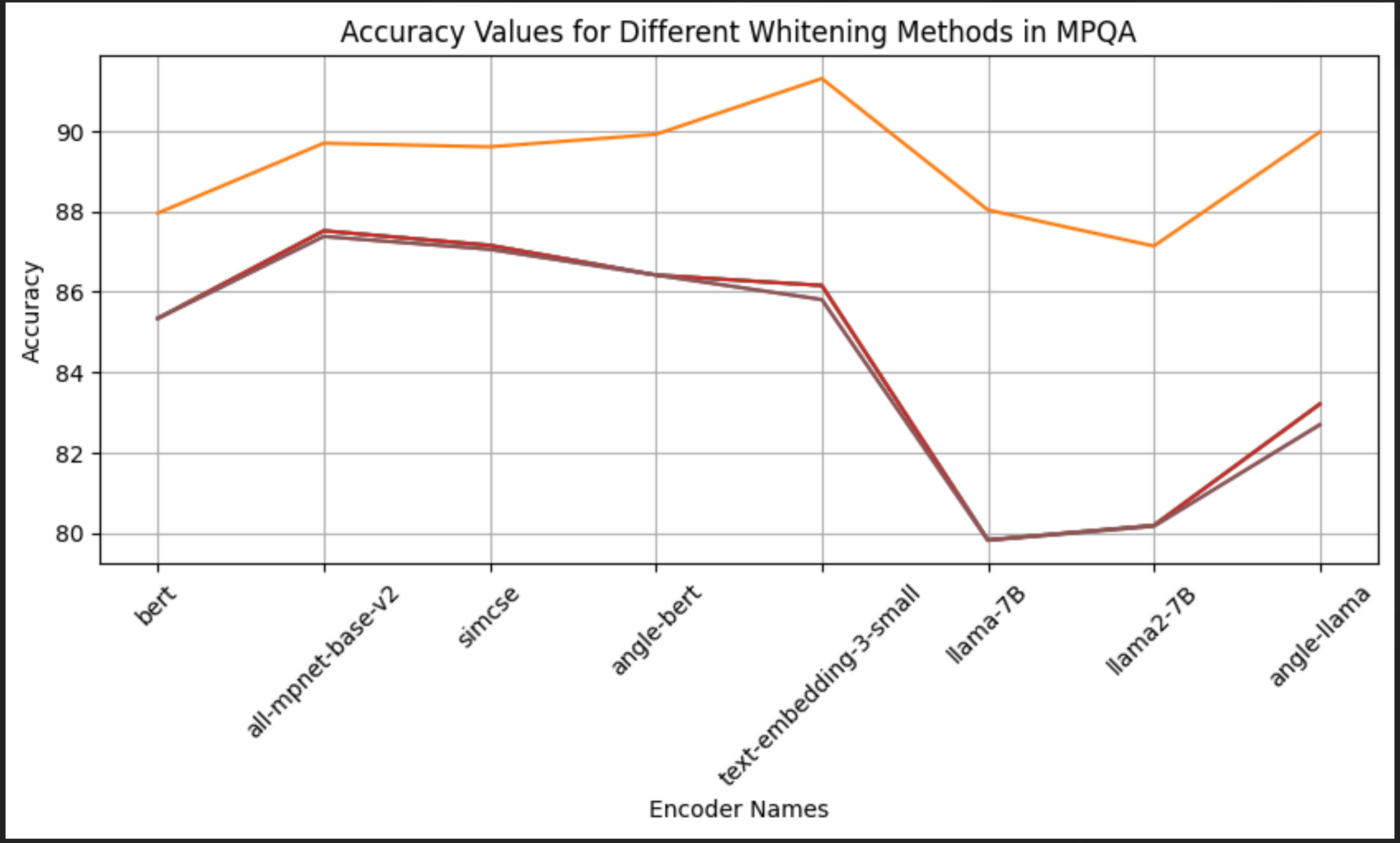


CLASSIFICATION RESULTS

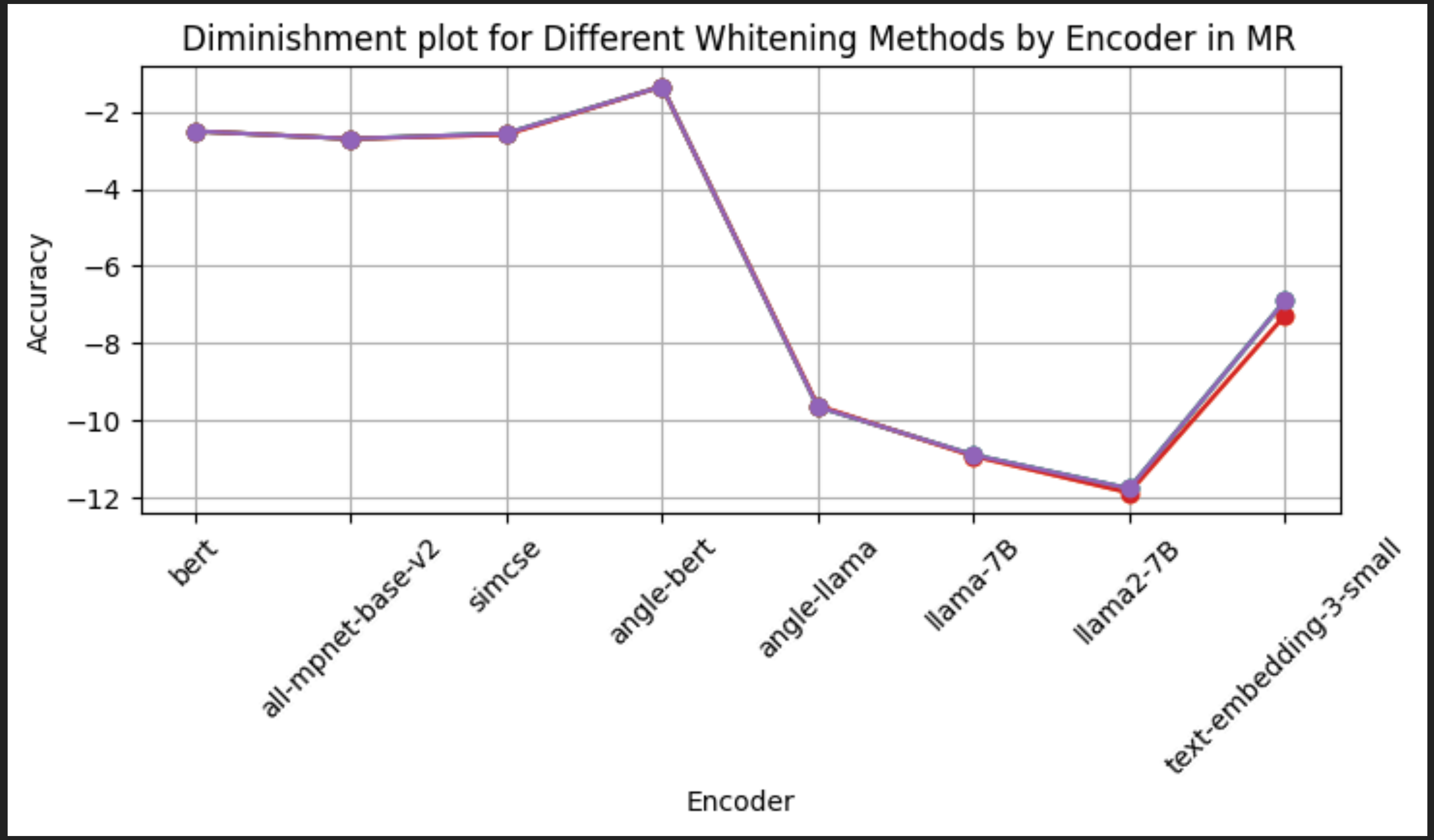


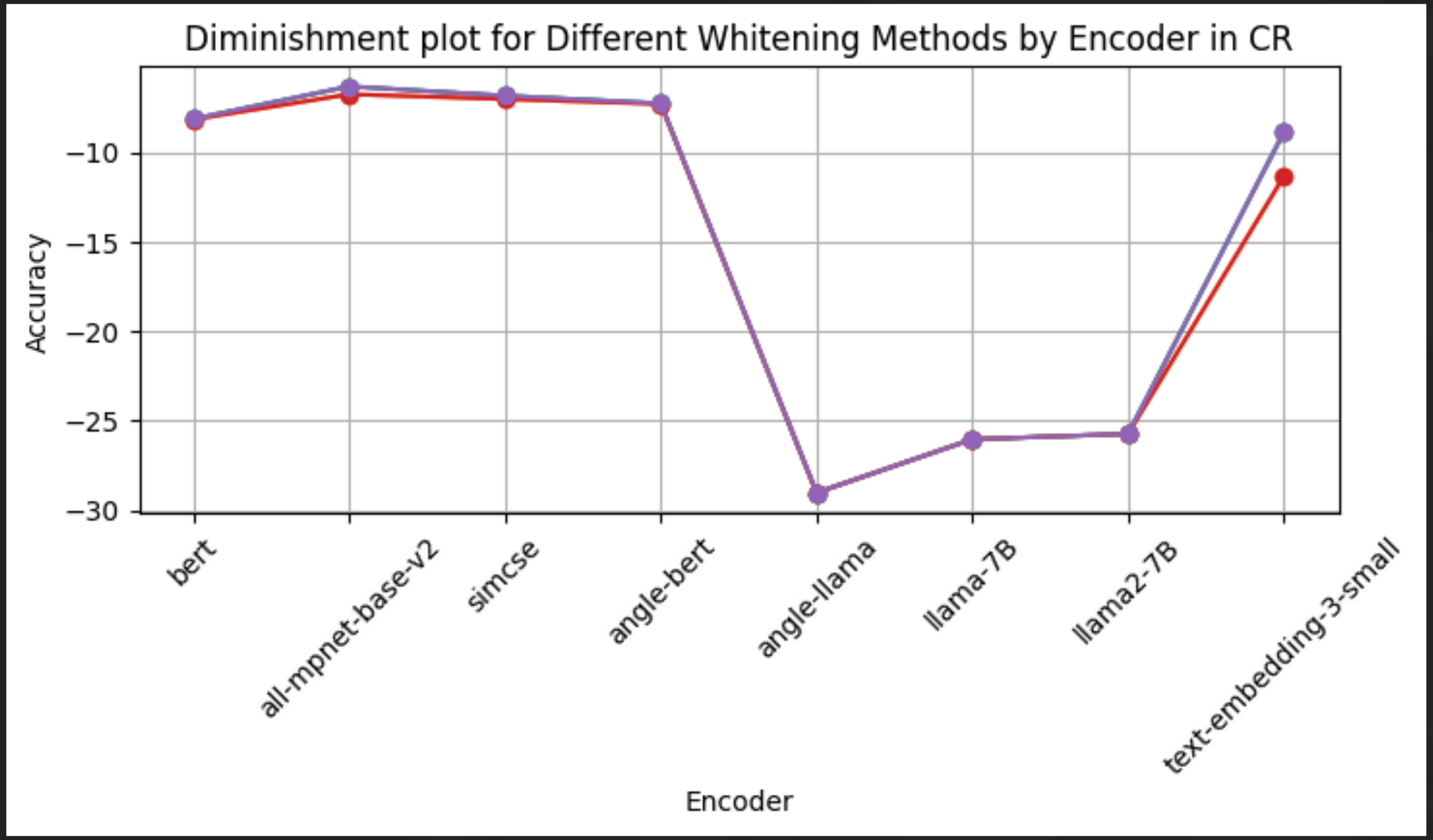


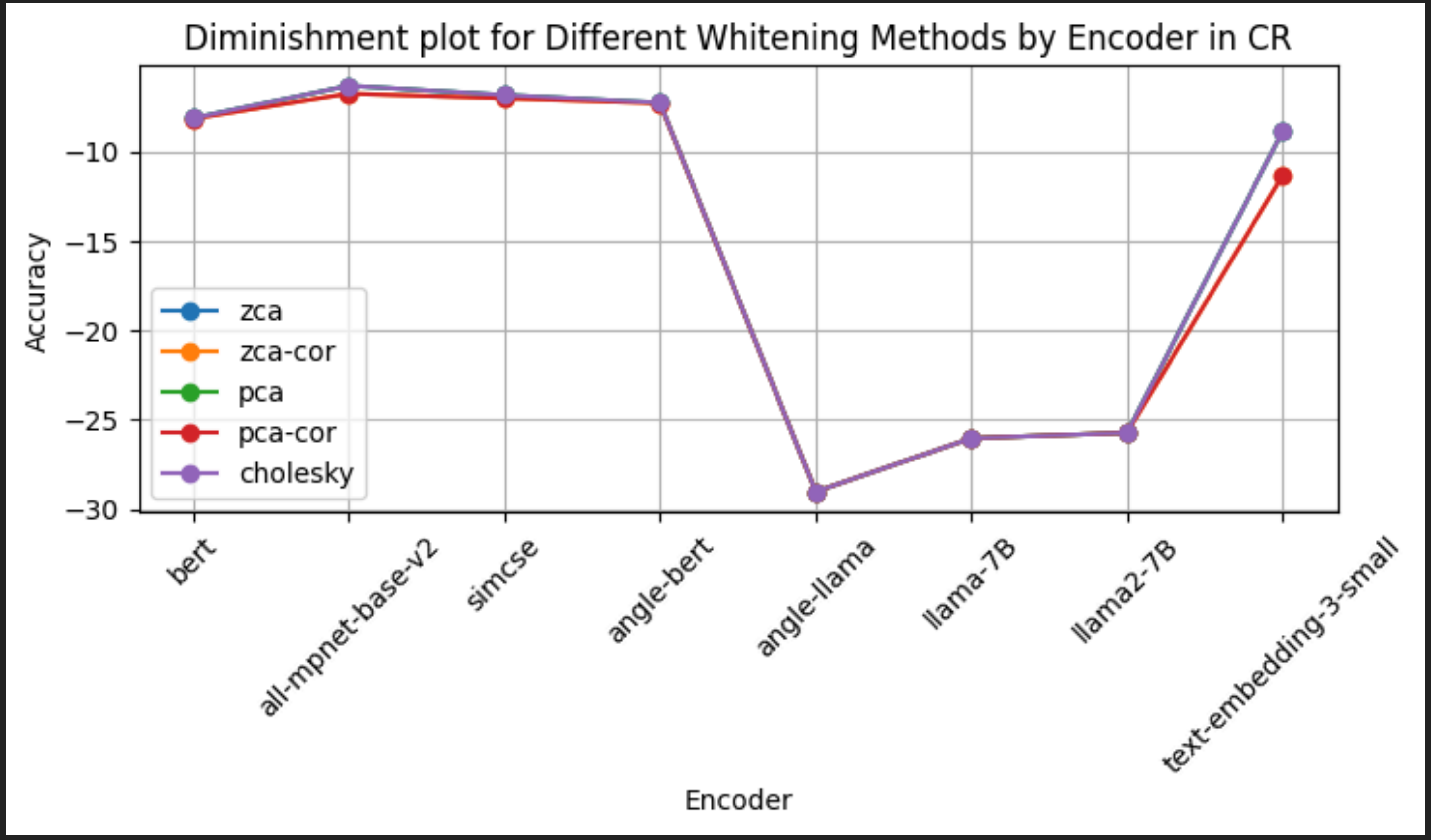


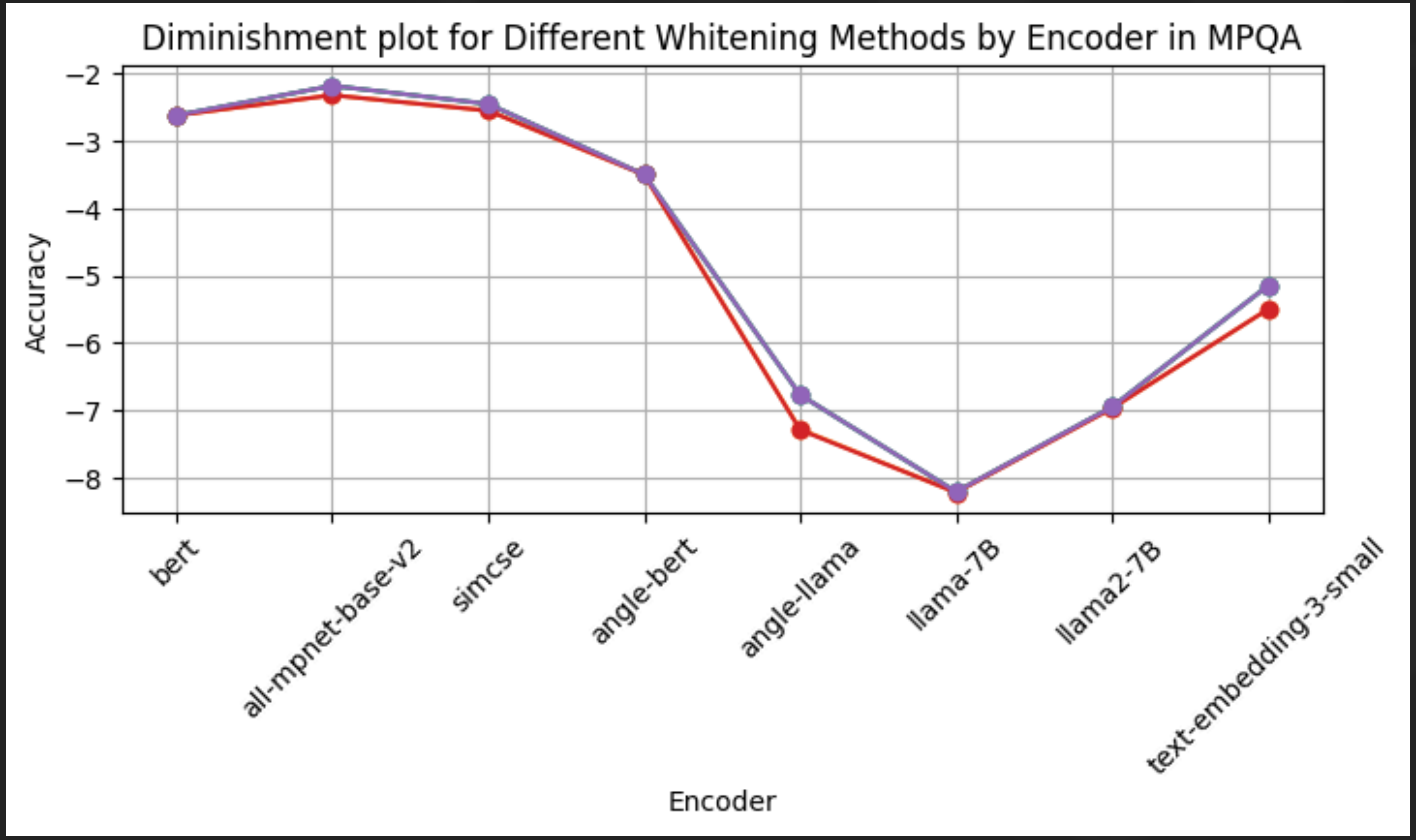


IMPROVEMENT / DIMINSHMENT PLOTS

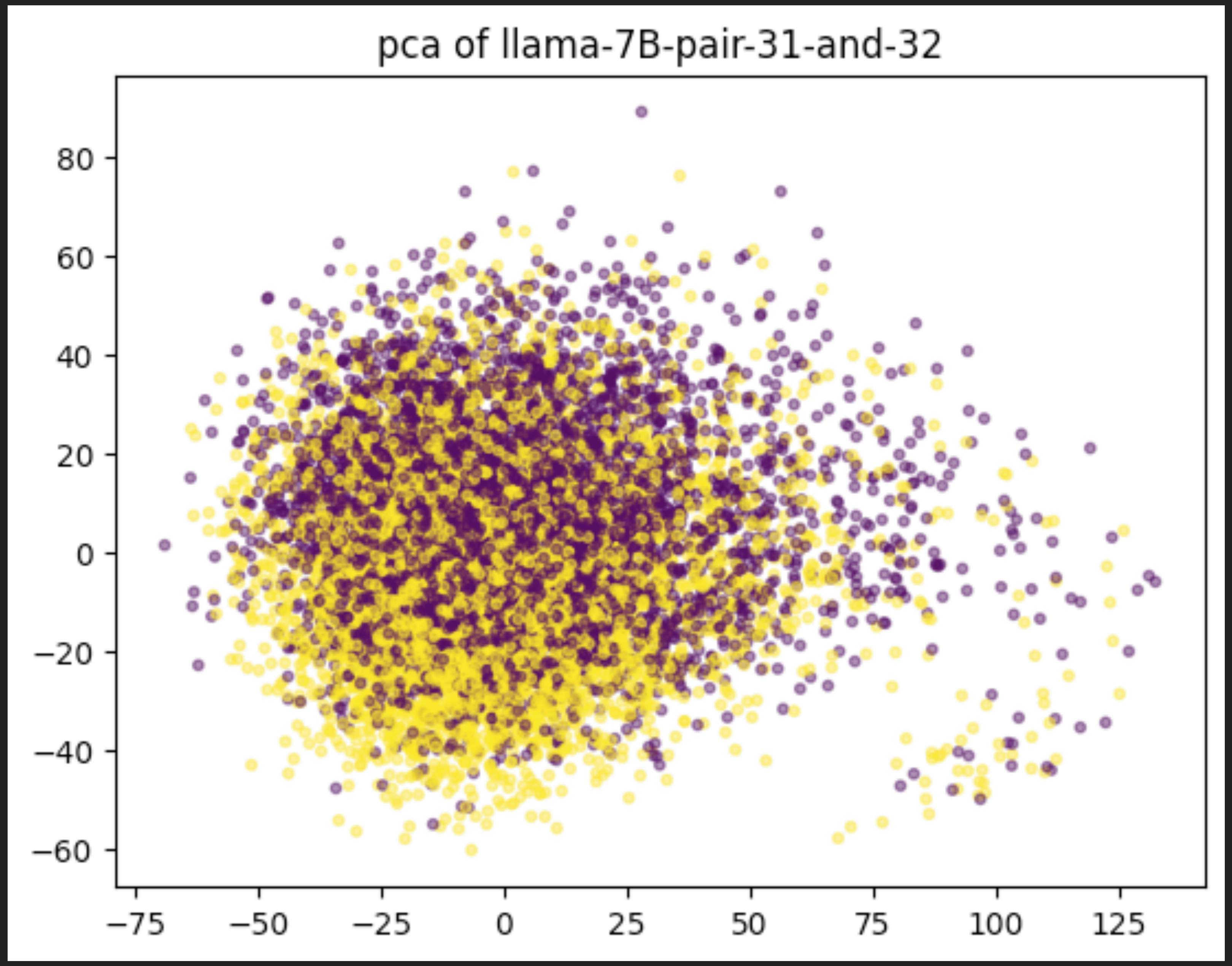


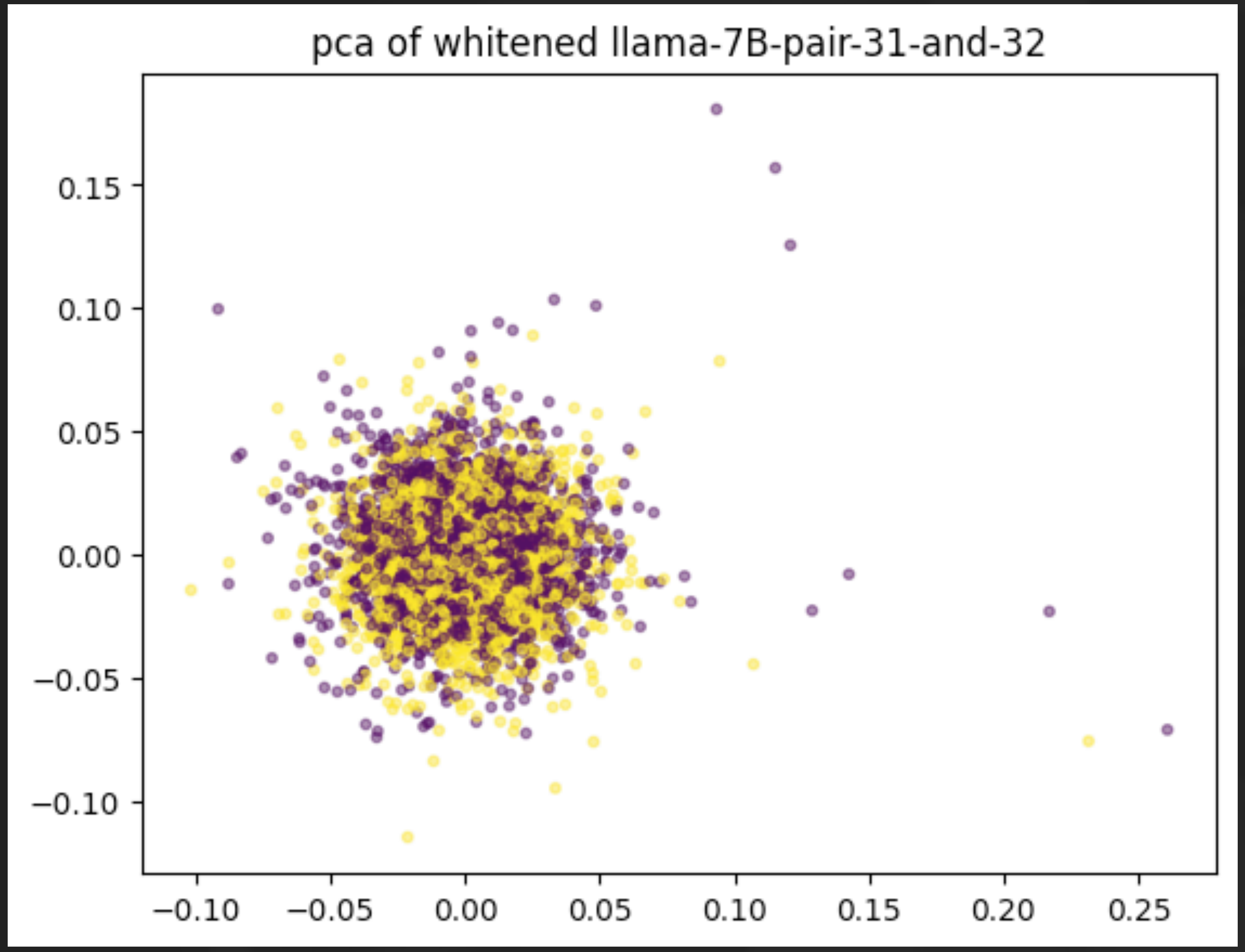


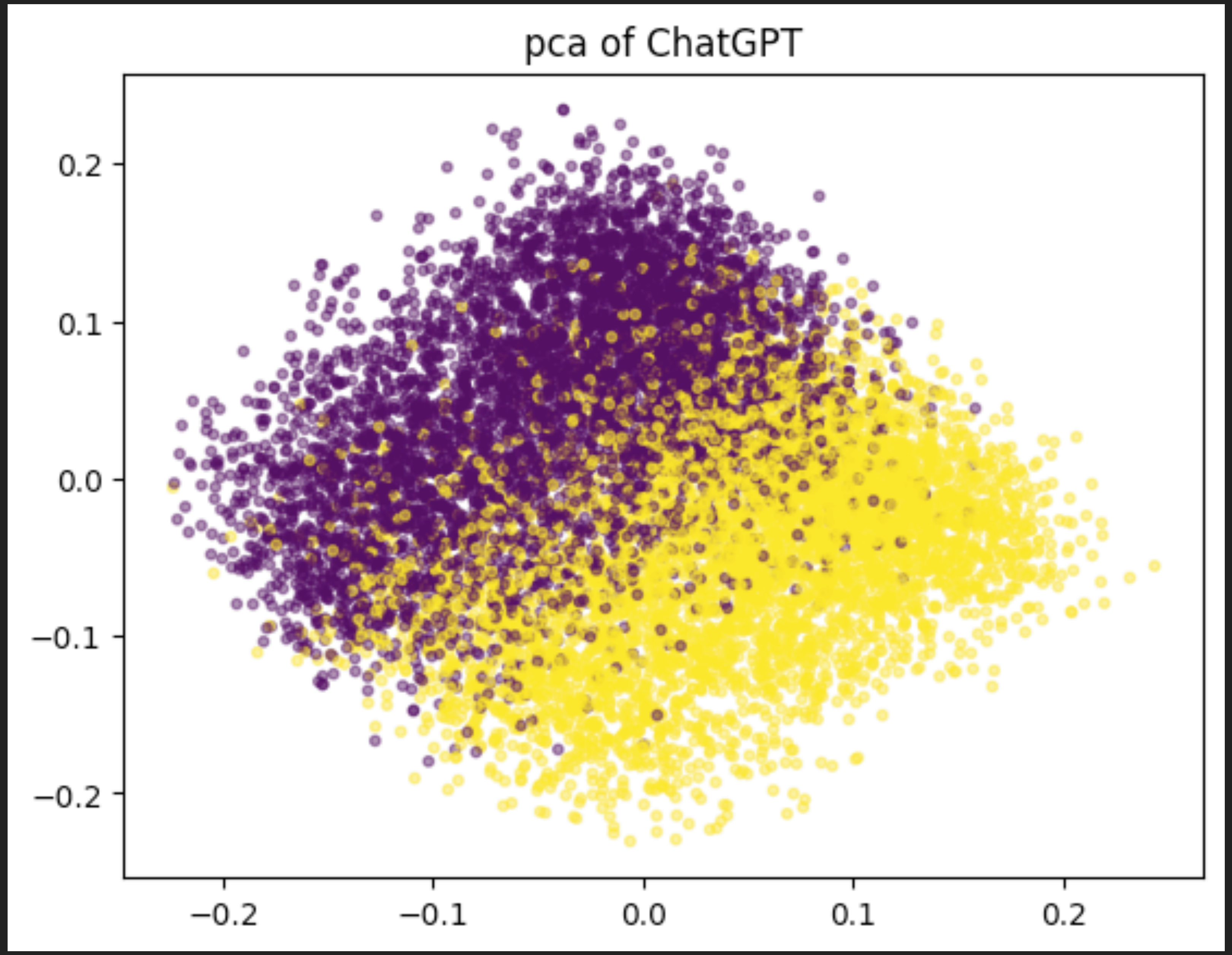


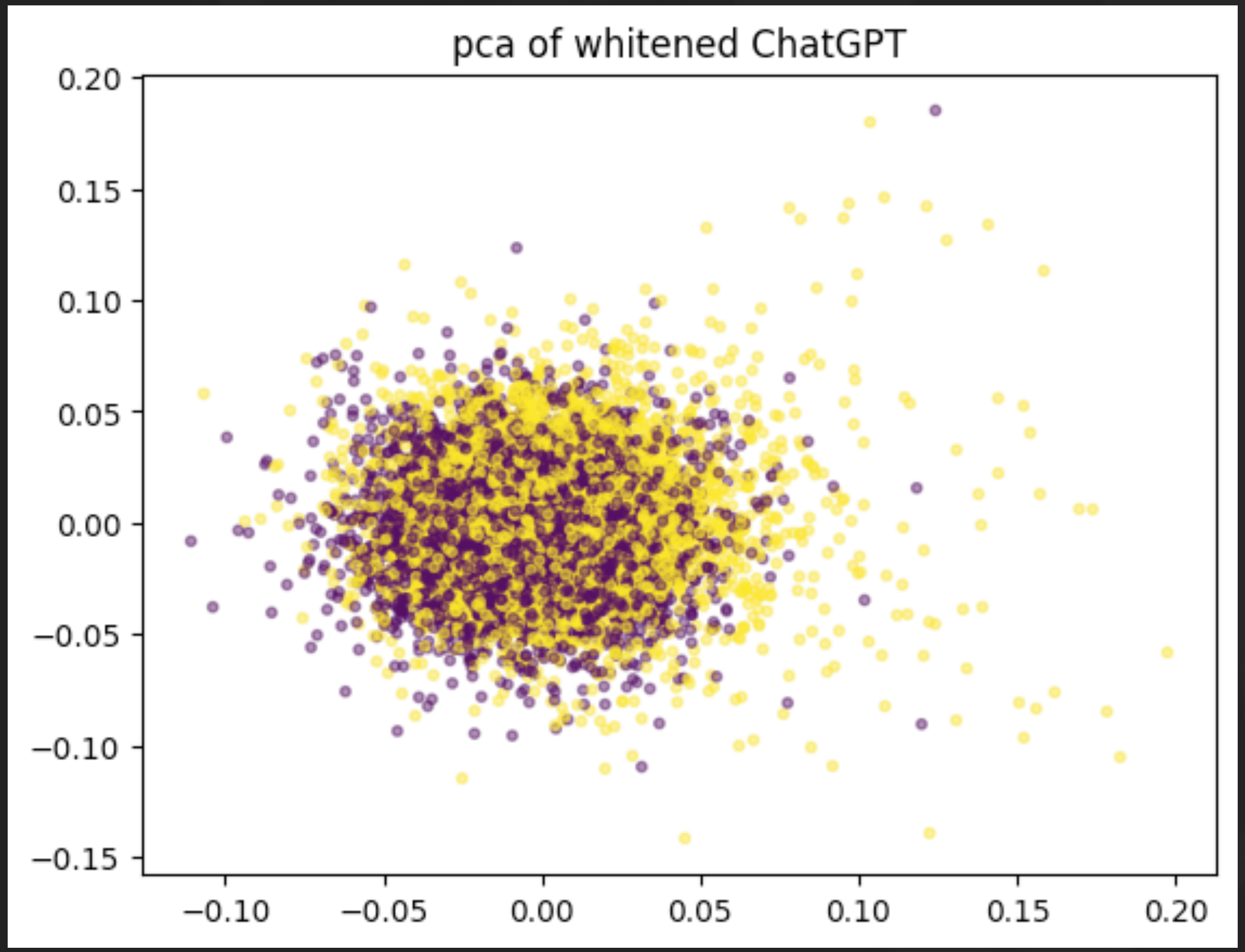


PCA BEFORE AND AFTER WHITENING

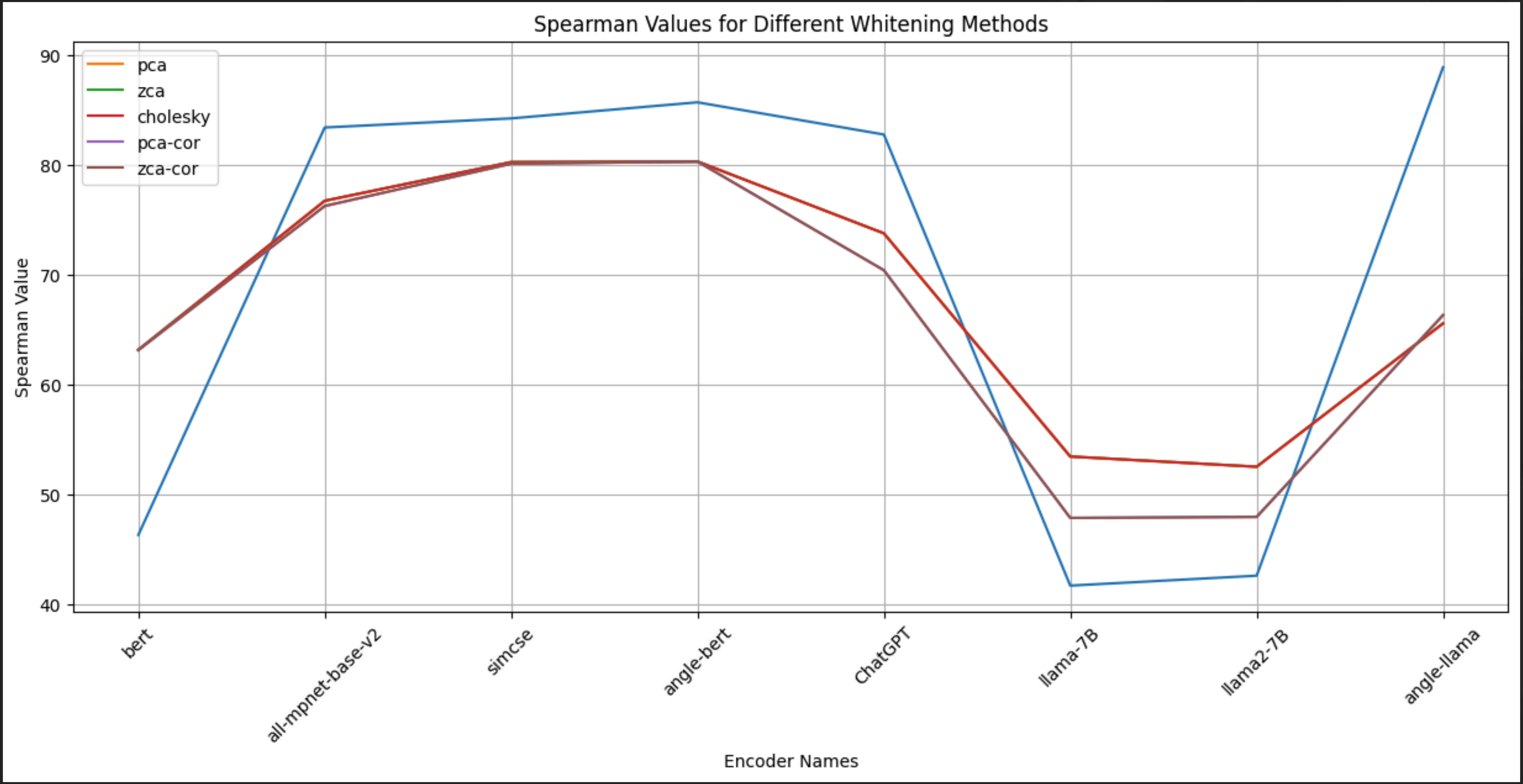




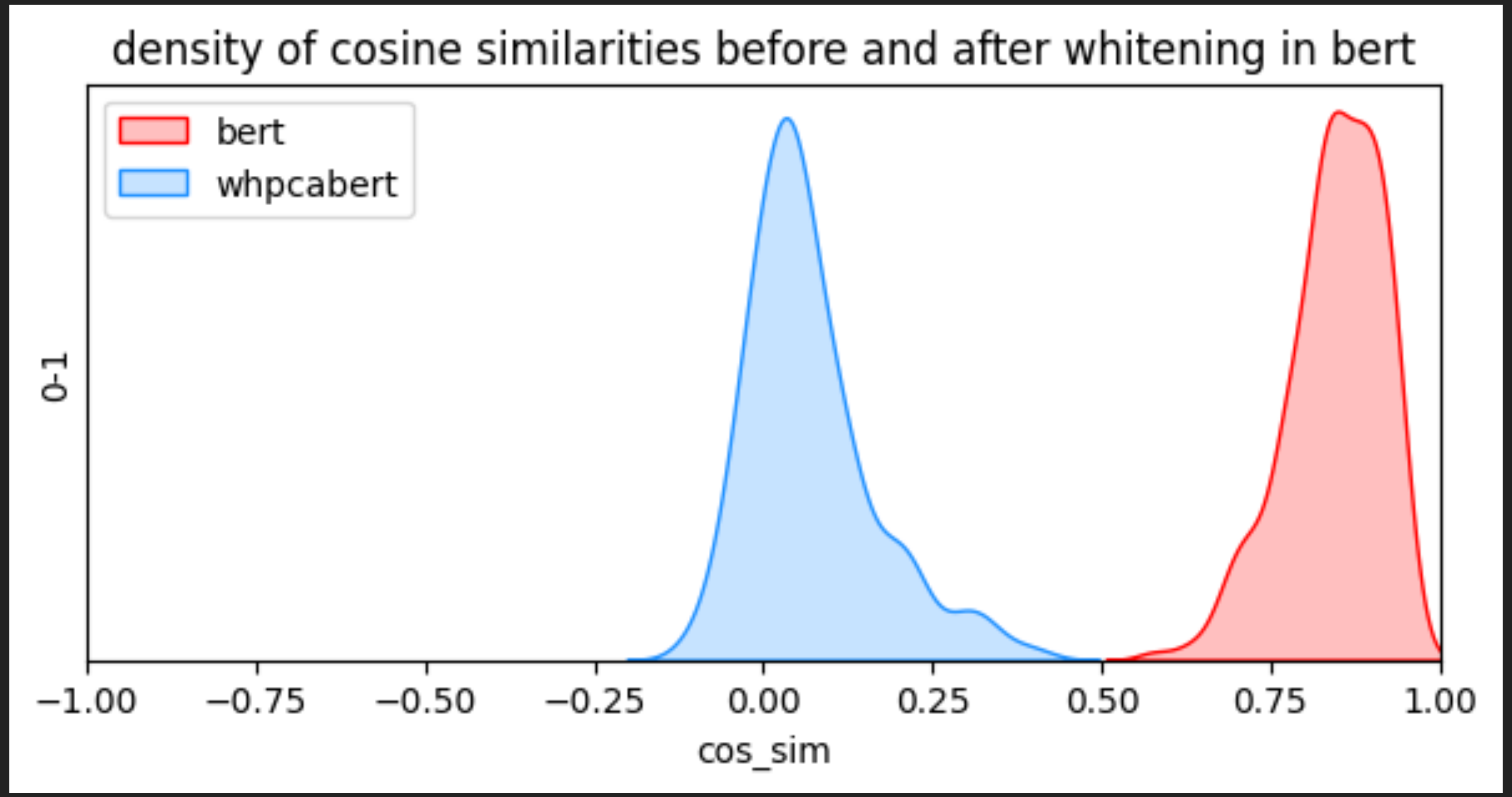


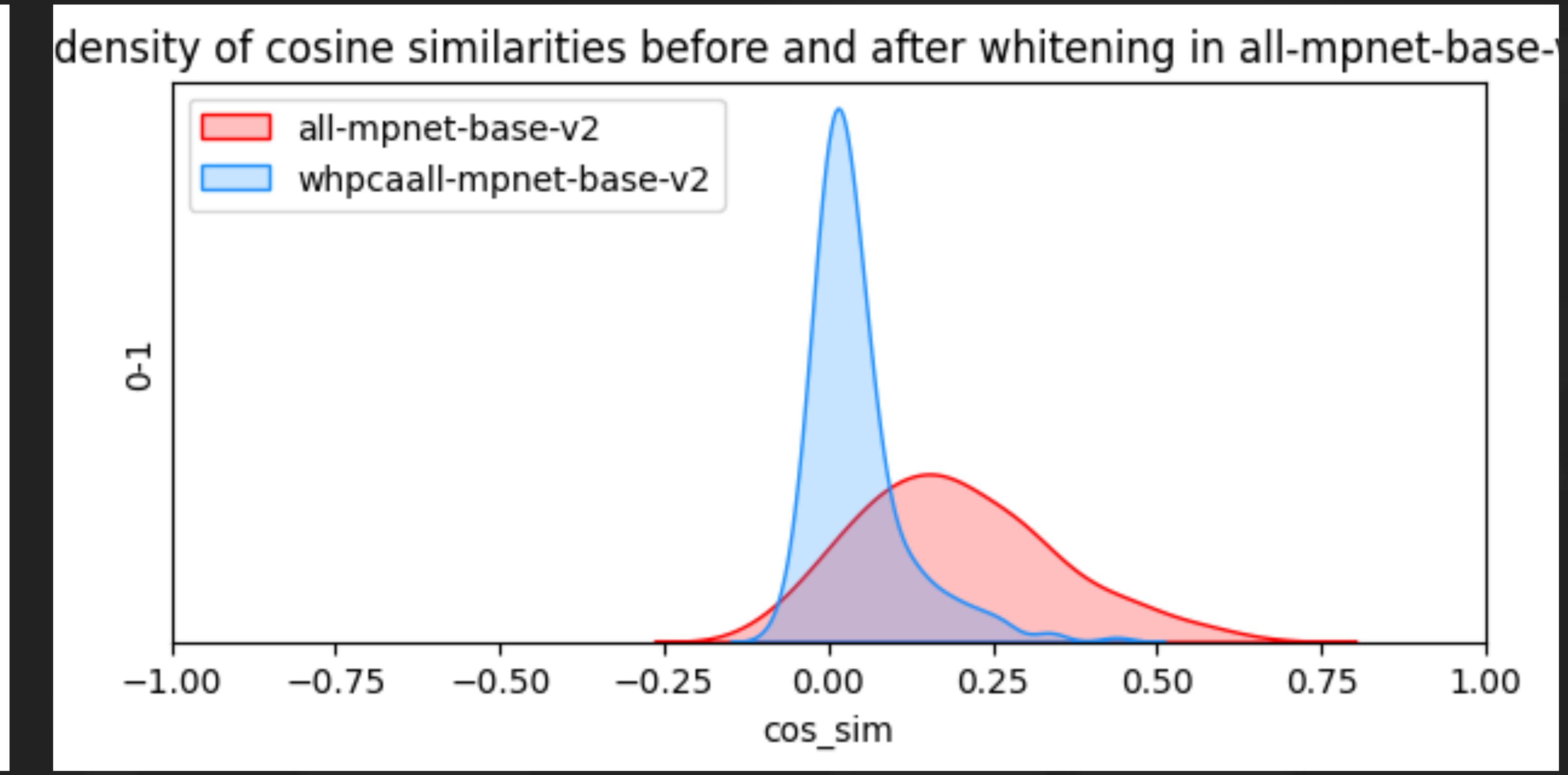


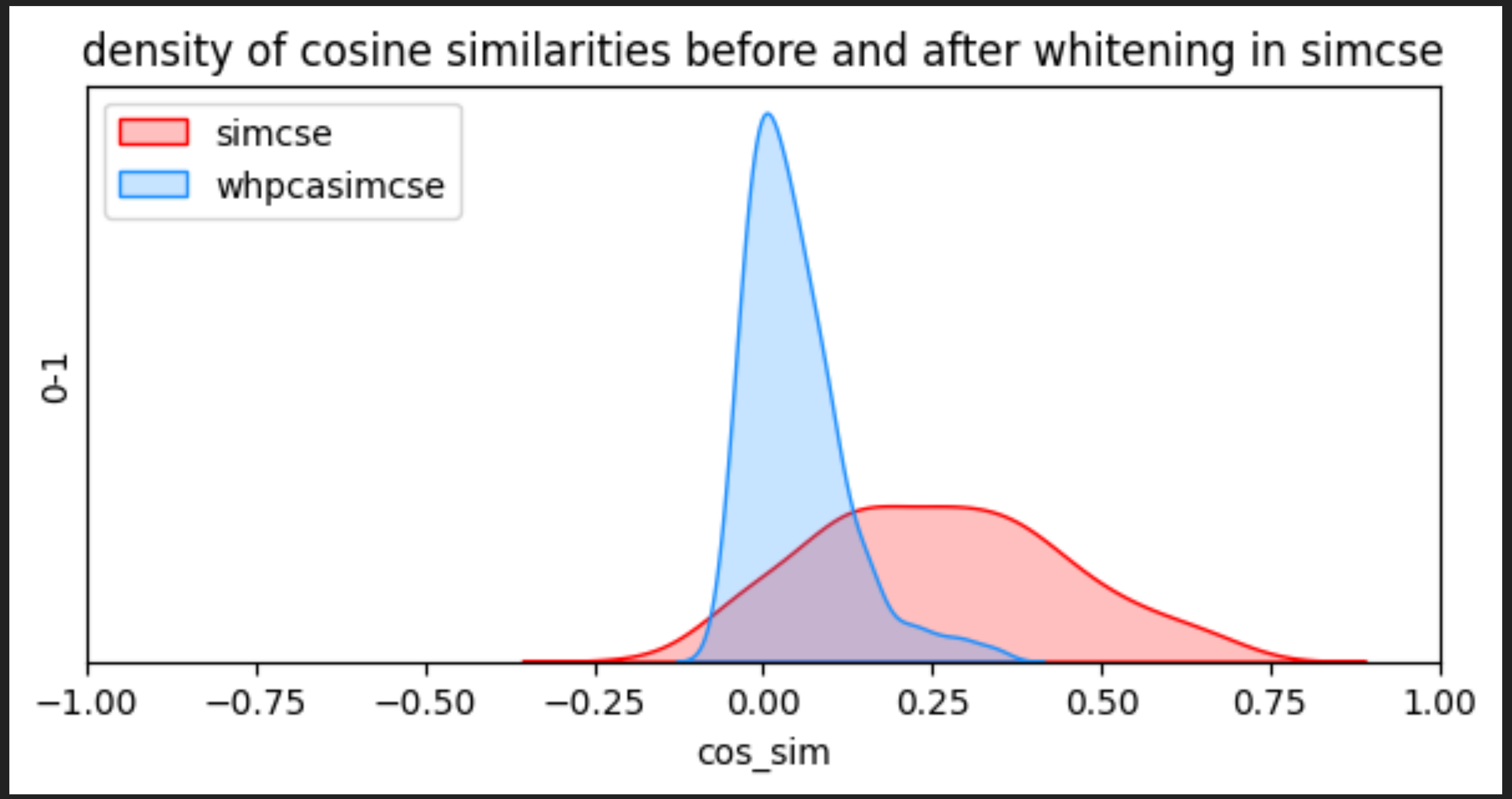
EFFECT OF WHITENING ON TEXT CLASSIFICATION

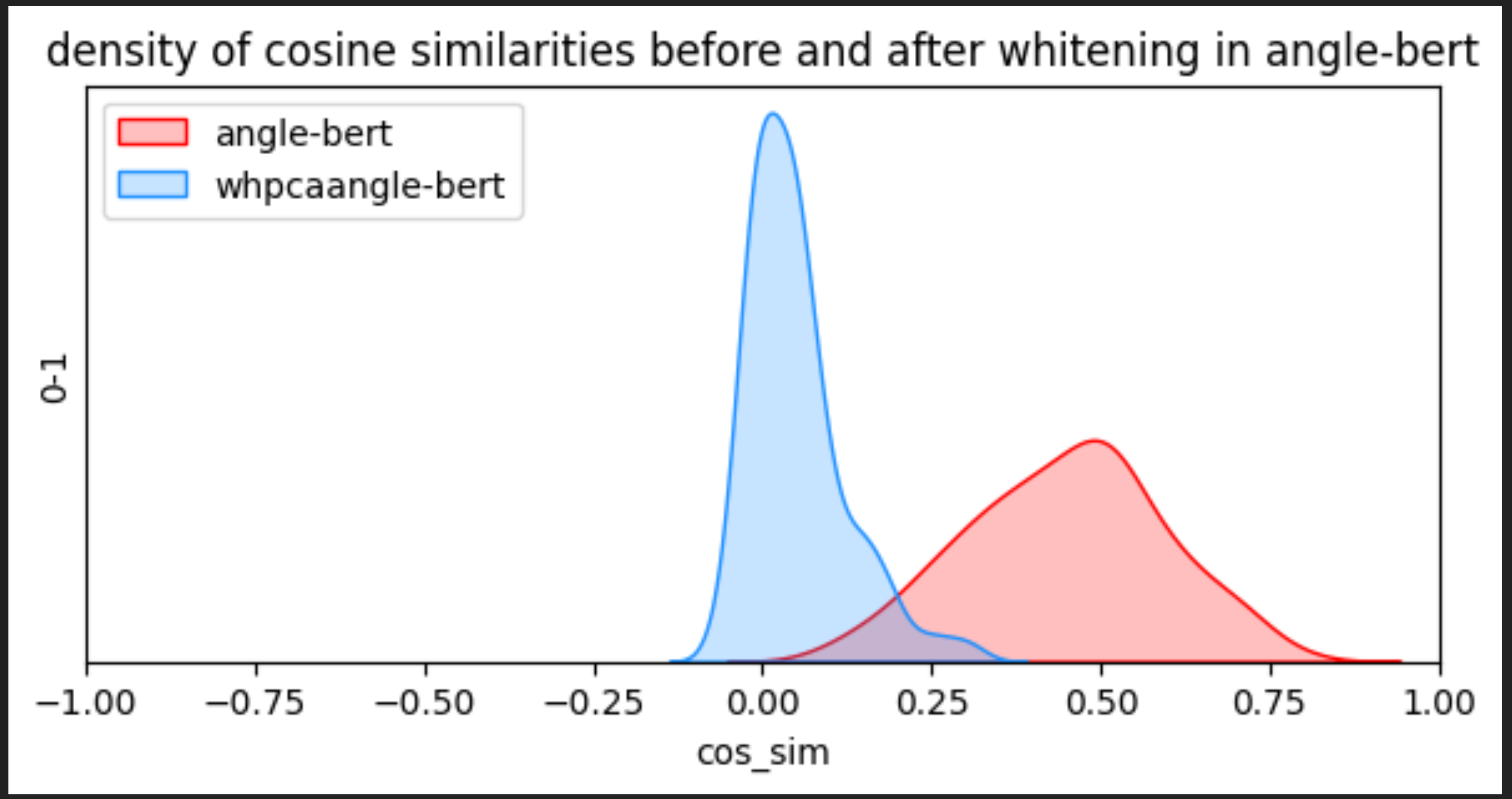


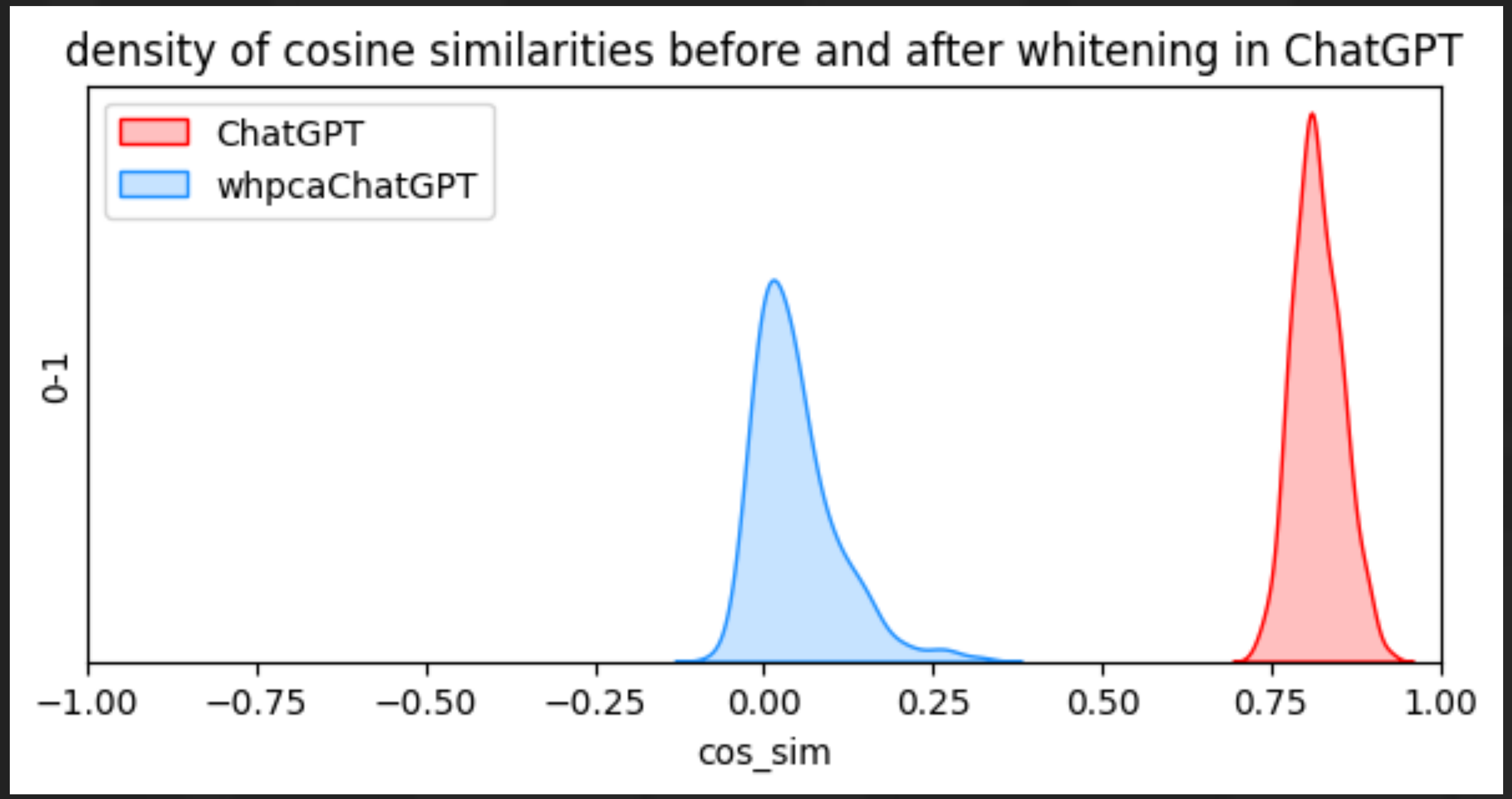
COSINE SIMILARITY DENSITY BEFORE AND AFTER PCA WHITENING

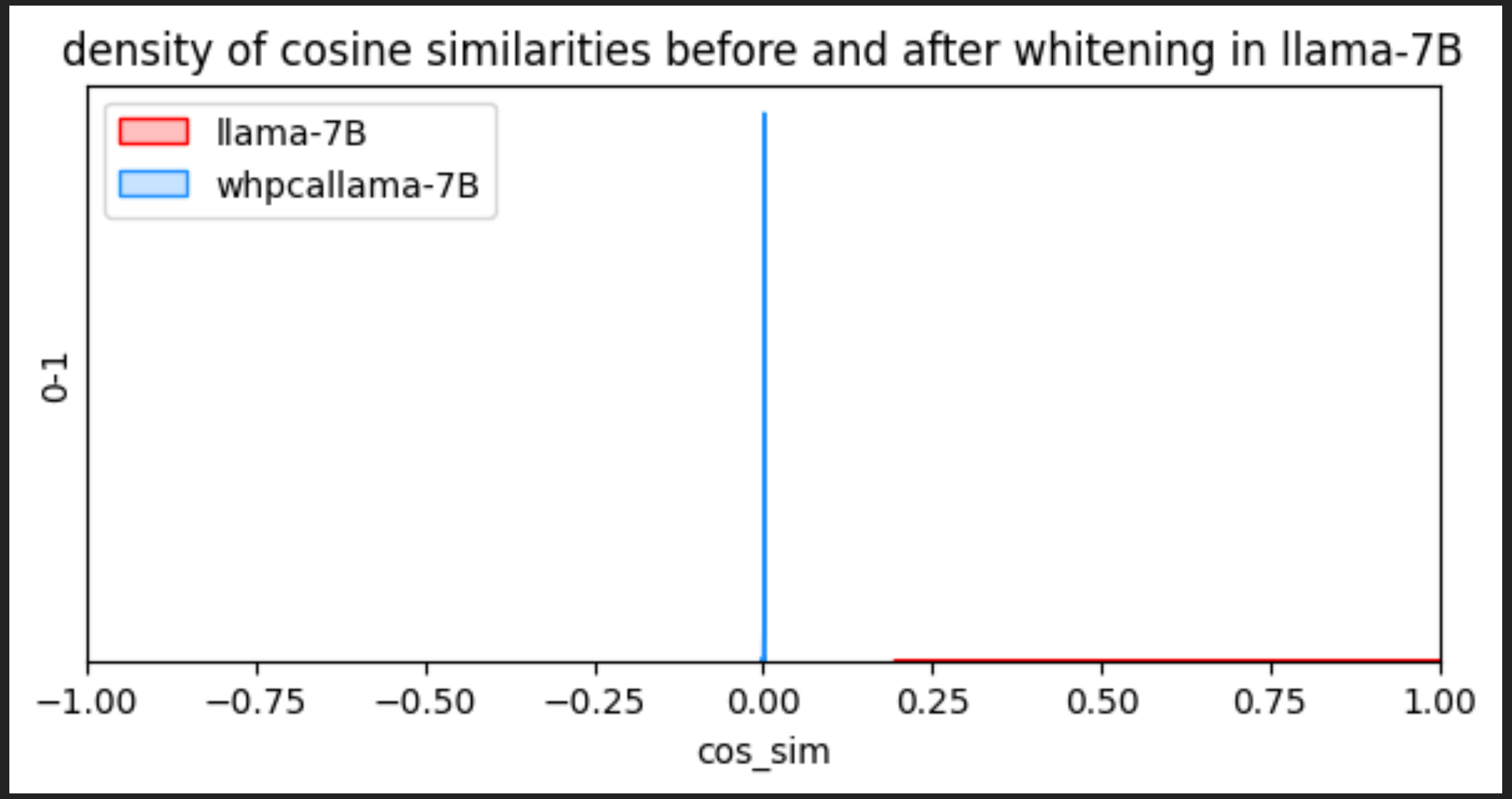


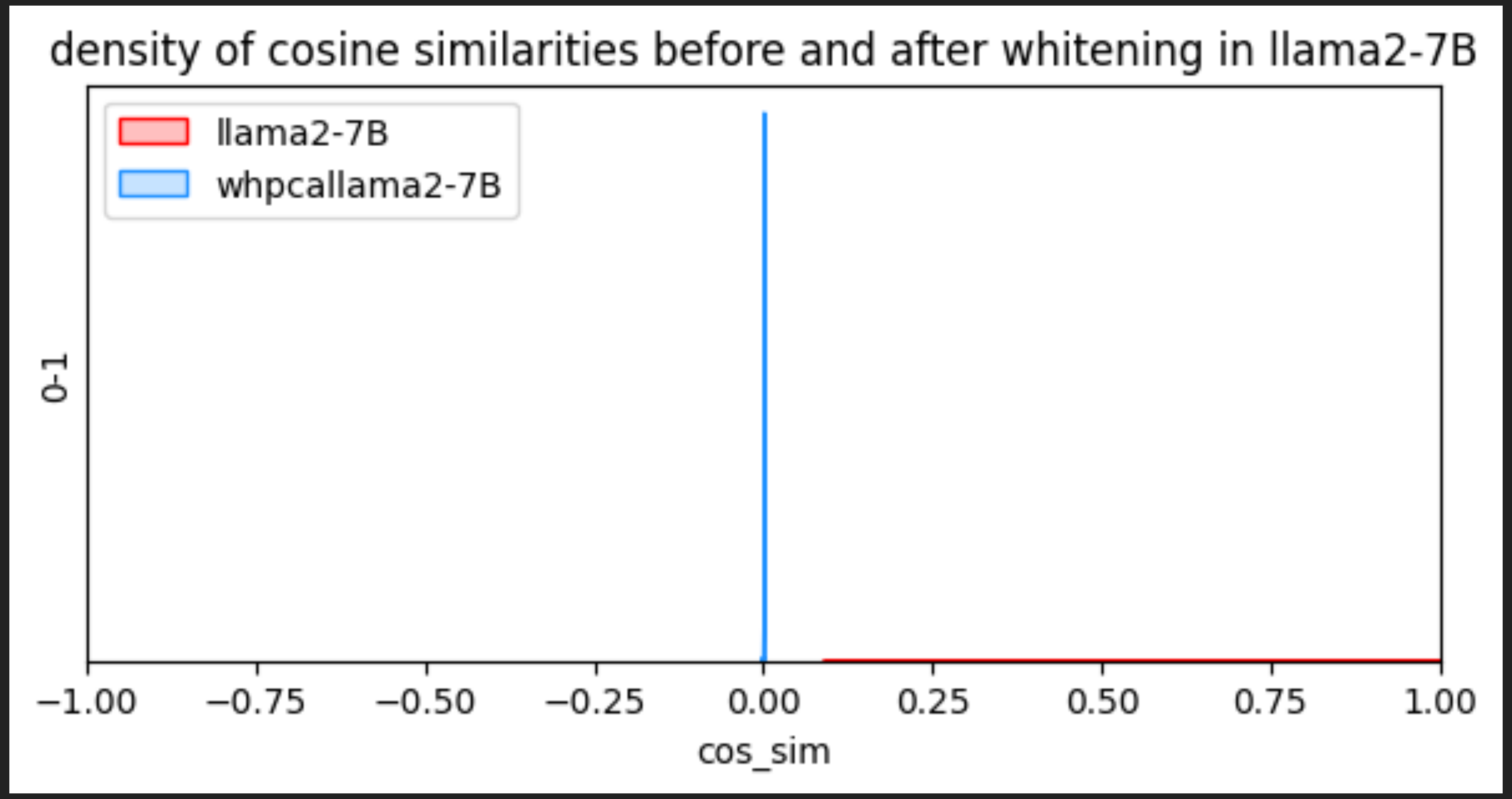


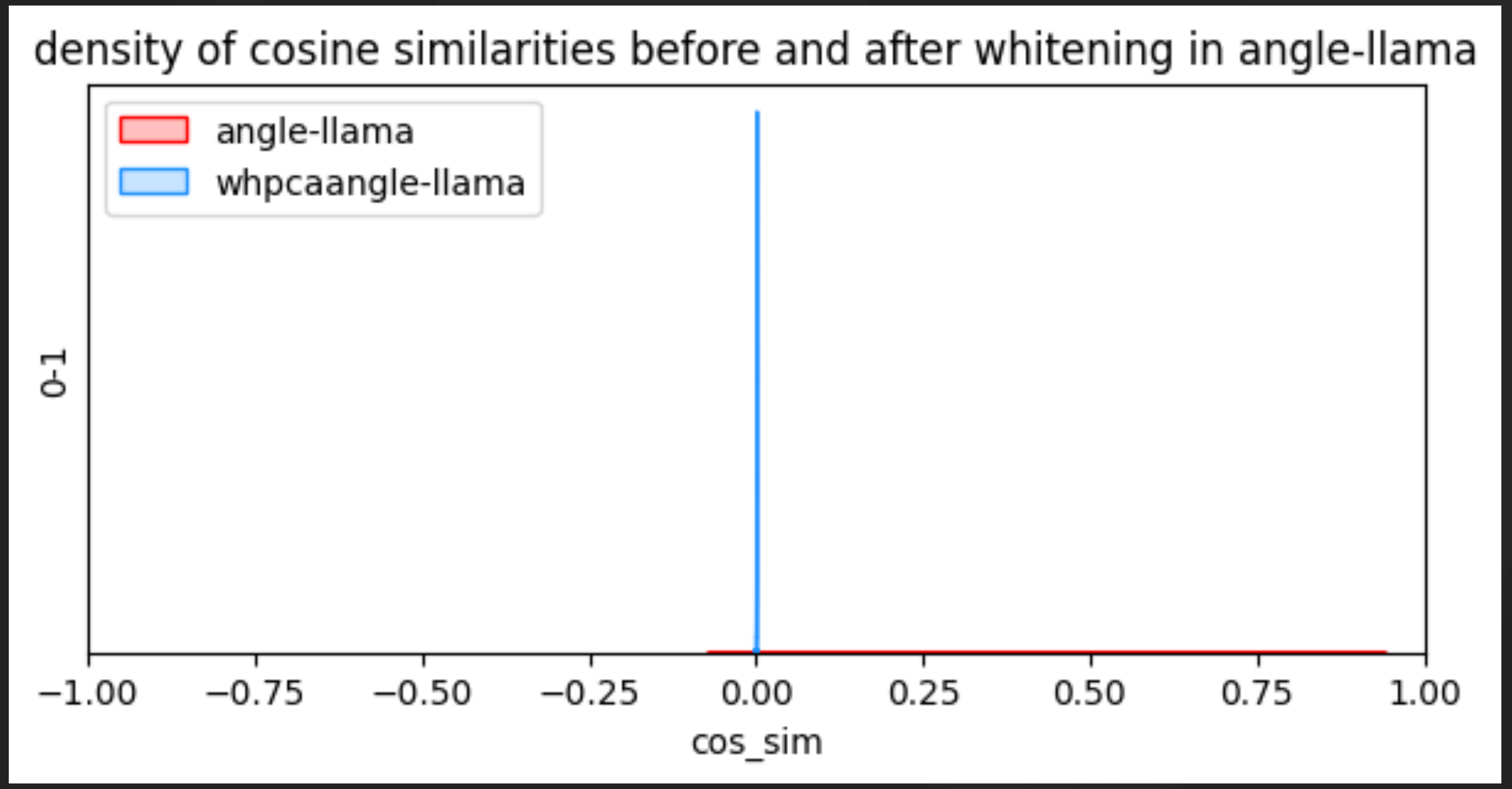




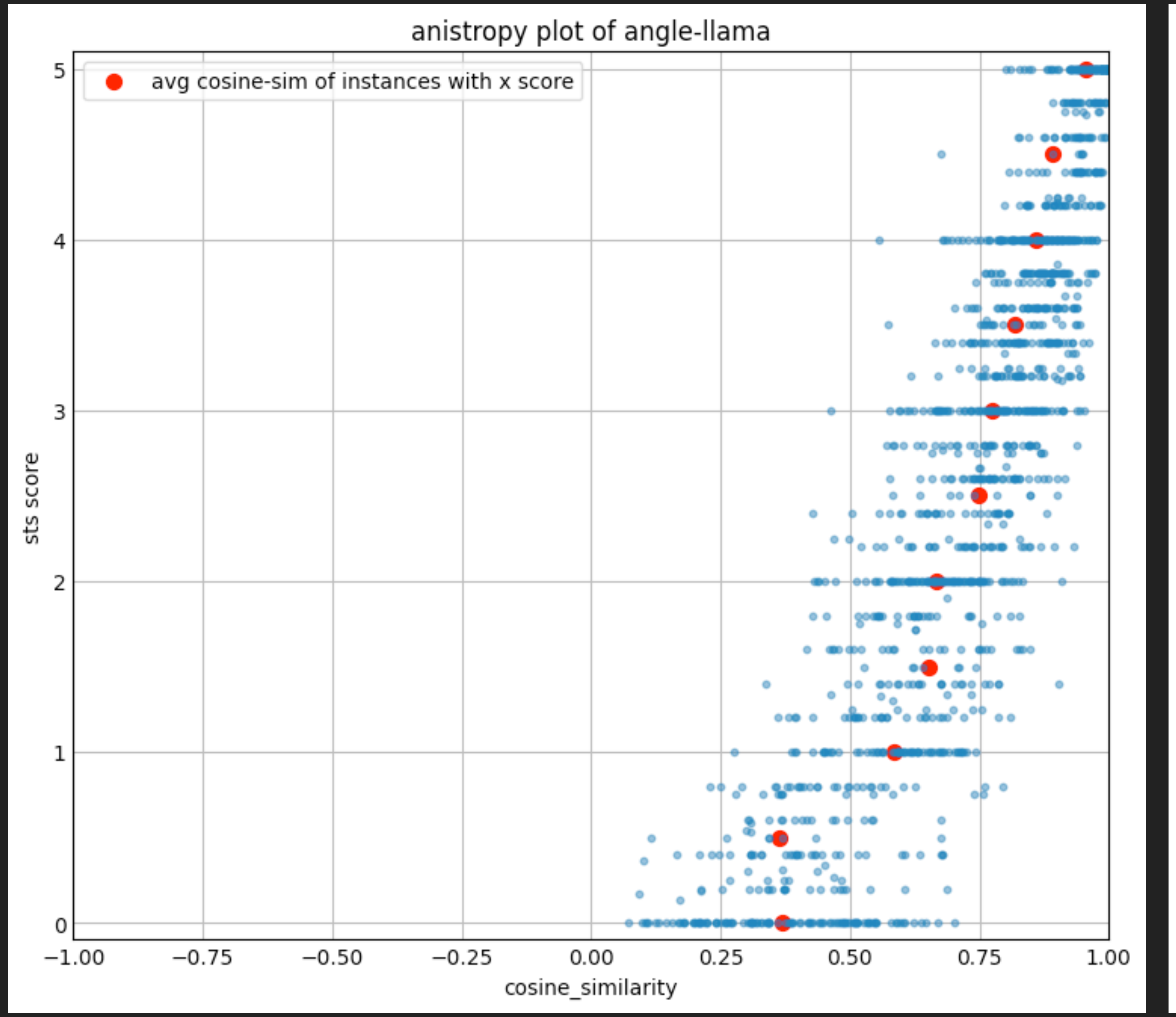


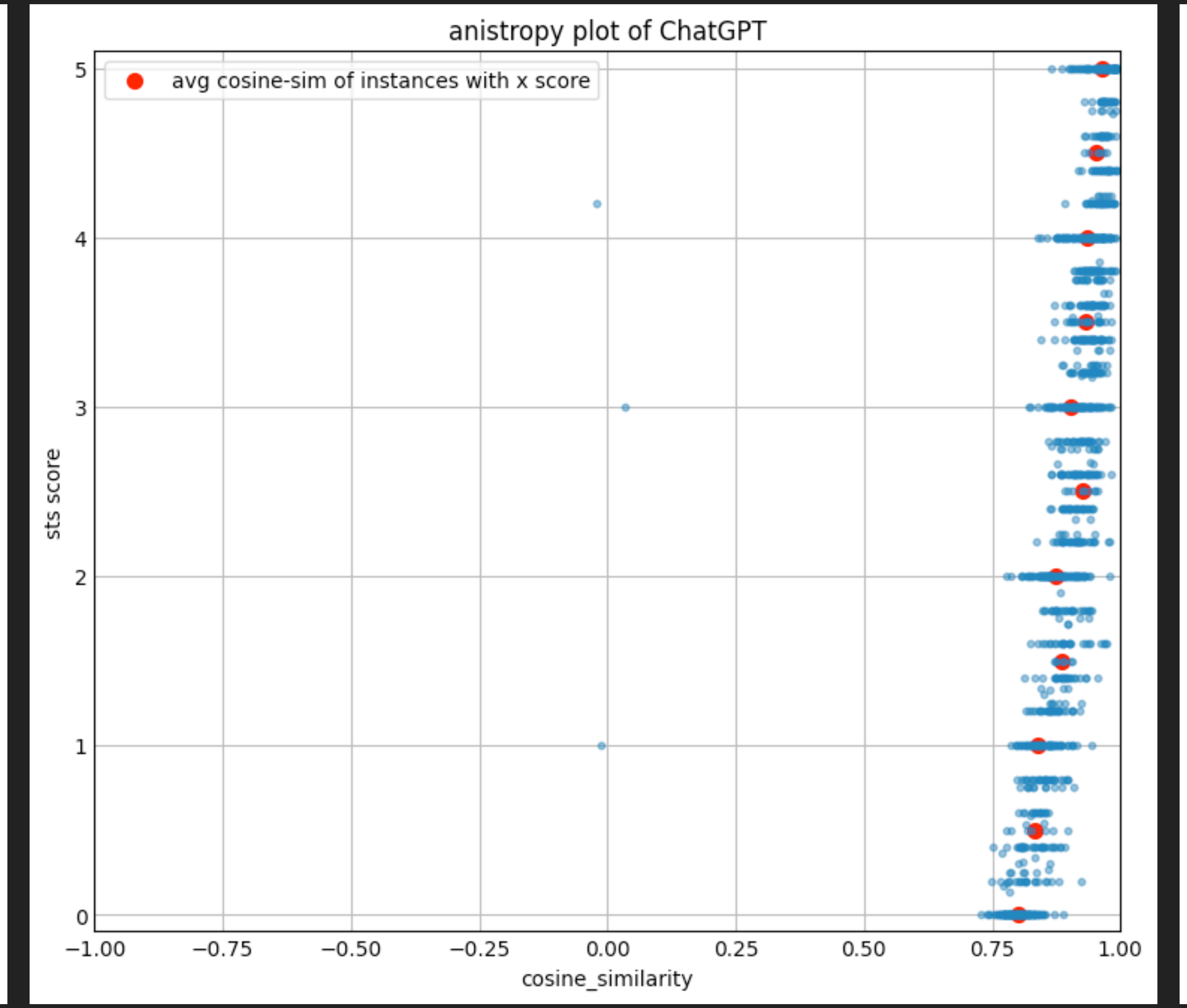


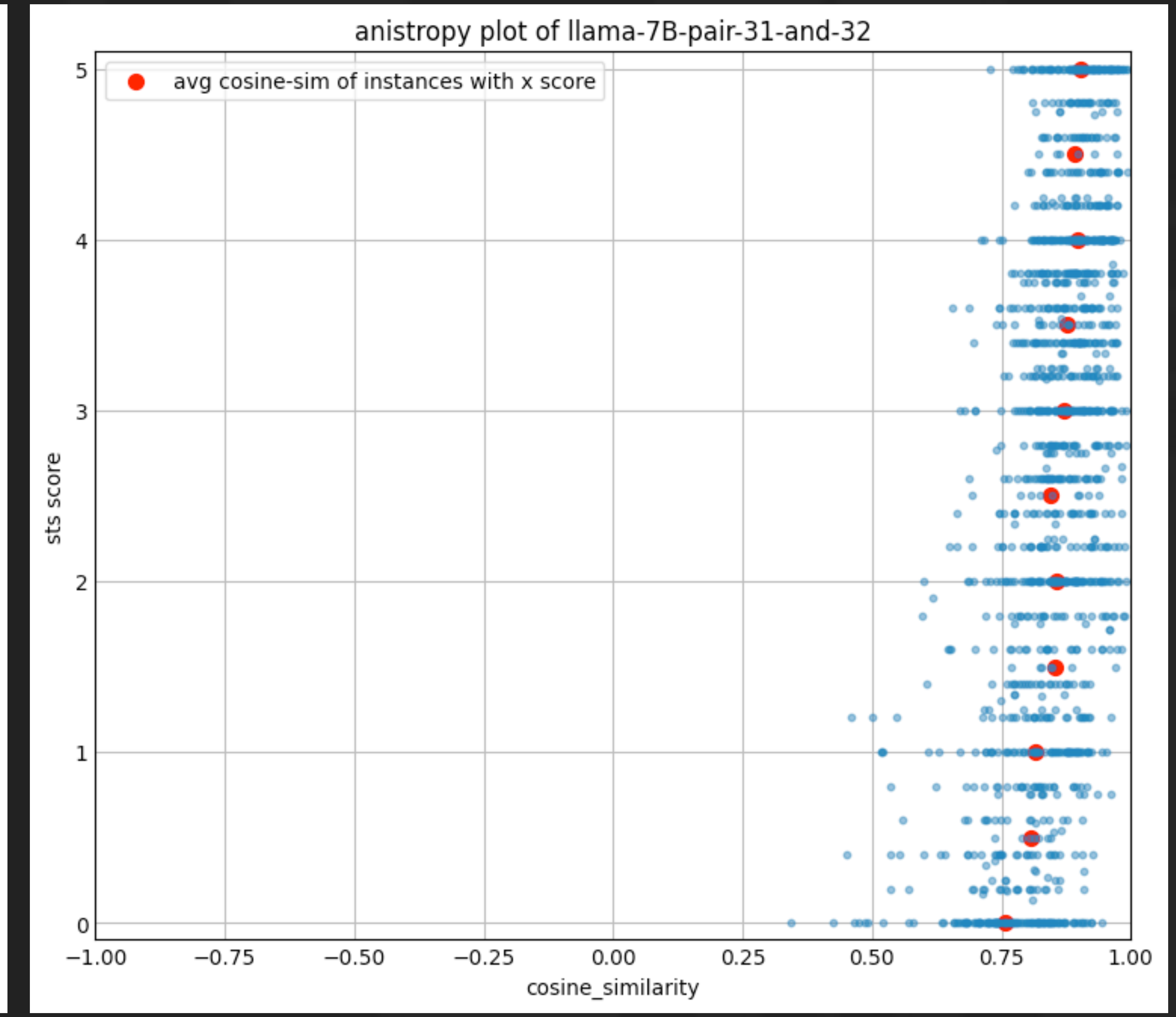




ANISTROPY PLOTS


anistropy plot of angle-llama
● avg cosine-sim of instances with x score
sts score
cosine_similarity


anistropy plot of ChatGPT
● avg cosine-sim of instances with x score
sts score
cosine_similarity


anistropy plot of llama-7B-pair-31-and-32
● avg cosine-sim of instances with x score
sts score
cosine_similarity