Steps taken to disable KV caching:

1. **In model.py modify Attention.forward():**
   Before:

```
bsz, seqlen, _ = x.shape
xq, xk, xv = self.wq(x), self.wk(x), self.wv(x)

xq = xq.view(bsz, seqlen, self.n_local_heads, self.head_dim)
xk = xk.view(bsz, seqlen, self.n_local_kv_heads, self.head_dim)
xv = xv.view(bsz, seqlen, self.n_local_kv_heads, self.head_dim)

xq, xk = apply_rotary_emb(xq, xk, freqs_cis=freqs_cis)

if self.kv_caching:
    self.cache_k = self.cache_k.to(xq)
    self.cache_v = self.cache_v.to(xq)

    self.cache_k[:bsz, start_pos : start_pos + seqlen] = xk
    self.cache_v[:bsz, start_pos : start_pos + seqlen] = xv

    keys = self.cache_k[:bsz, : start_pos + seqlen]
    values = self.cache_v[:bsz, : start_pos + seqlen]
else:
    pass
```

   After:

```
bsz, seqlen, _ = x.shape
xq, xk, xv = self.wq(x), self.wk(x), self.wv(x)

xq = xq.view(bsz, seqlen, self.n_local_heads, self.head_dim)
xk = xk.view(bsz, seqlen, self.n_local_kv_heads, self.head_dim)
xv = xv.view(bsz, seqlen, self.n_local_kv_heads, self.head_dim)

xq, xk = apply_rotary_emb(xq, xk, freqs_cis=freqs_cis)

if self.kv_caching:
    self.cache_k = self.cache_k.to(xq)
    self.cache_v = self.cache_v.to(xq)

    self.cache_k[:bsz, start_pos : start_pos + seqlen] = xk
    self.cache_v[:bsz, start_pos : start_pos + seqlen] = xv

    keys = self.cache_k[:bsz, : start_pos + seqlen]
    values = self.cache_v[:bsz, : start_pos + seqlen]
else:
    keys = xk
    values = xv
```

2. **In generation.py modify Generation.generate():**

Before:

```
for cur_pos in range(min_prompt_len, total_len):
    with torch.no_grad():
        if kv_caching:
            logits = self(tokens[:, prev_pos:cur_pos], prev_pos)
        else:
            pass
    if temperature > 0:
        probs = torch.softmax(logits[:, -1] / temperature, dim=-1)
        next_token = sample_top_p(probs, top_p)
    else:
        next_token = torch.argmax(logits[:, -1], dim=-1)
```

After:

```
for cur_pos in range(min_prompt_len, total_len):
    with torch.no_grad():
        if kv_caching:
            logits = self(tokens[:, prev_pos:cur_pos], prev_pos)
        else:
            logits = self(tokens[:, : cur_pos], 0)
    if temperature > 0:
        probs = torch.softmax(logits[:, -1] / temperature, dim=-1)
        next_token = sample_top_p(probs, top_p)
    else:
        next_token = torch.argmax(logits[:, -1], dim=-1)
```

In benchmark_inference.py:

Set kv_caching to True for inference with KV cache and set it to False for inference without KV cache.

```
if __name__ == "__main__":

    print("=== KV Caching ===")
    benchmark_inference(batch_size=1, input_len=256, output_len=32, kv_caching=True)
    benchmark_inference(batch_size=8, input_len=256, output_len=32, kv_caching=True)
    benchmark_inference(batch_size=16, input_len=256, output_len=32, kv_caching=True)

    print("\n=== No KV Caching ===")

    benchmark_inference(batch_size=1, input_len=256, output_len=32, kv_caching=False)
    benchmark_inference(batch_size=8, input_len=256, output_len=32, kv_caching=False)
    benchmark_inference(batch_size=16, input_len=256, output_len=32, kv_caching=False)
```

Input len = 256, output len = 32		Batch size = 1	Batch size = 8	Batch size = 16
With KV cache	Peak Mem	3071.57 MB	4495.26 MB	6133.10 MB
	Runtime	1.03 secs	1.82 secs	2.75 secs
Without KV cache	Peak Mem	6133.10 MB	6133.10 MB	8642.33 MB
	Runtime	3.96 secs	28.10 secs	52.82 secs