PART 1

Ans 1:

- **Briefly describe why you selected this dataset and what task you'll evaluate (summarization, QA, or text generation).**

  Why SQuAD v1.1 & what task?

  - Dataset: SQuAD v1.1 (Stanford Question Answering Dataset) it is a typical benchmark which has question-answer (with human-written questions) and extractive responses on Wikipedia passages.
  - Evaluation type: Question Answering(QA). Answer the question taking into consideration a situation.

  Why this dataset?

  - Extensively employed - inter-architectural clear baselines (EM/F1) and comparability.
  - It can be used with either extractive (BERT span prediction) or generative (T5/GPT text answers) configurations, which are ideal to compare encoder-only, encoder-decoder and decoder-only models.
  - responses are short spans - trainable and assessable, even on small subsets / CPU.

- **Show how you preprocessed the data (tokenization, train/val split, max length, etc.).**

  - Splits Used SQuAD official train split and validation split (no random split).

  Tokenization & lengths:

  - Input length= 384 tokens globally: Docstride=128, long context sliding windows (BERT).
  - Max length of target (answer) to generative models = 32 tokens.

  Architecture-specific formatting:

  - BERT (extractive): tokenize (question, context) truncation onlysecond, create start/end positions based on offset_mapping
  - T5 (text-to-text): input = question:... context:...; trigger = text of answer to gold; padding labels - -100 pad price: this is so that loss is not sensitive to padding.
  - GPT (causal): prompt

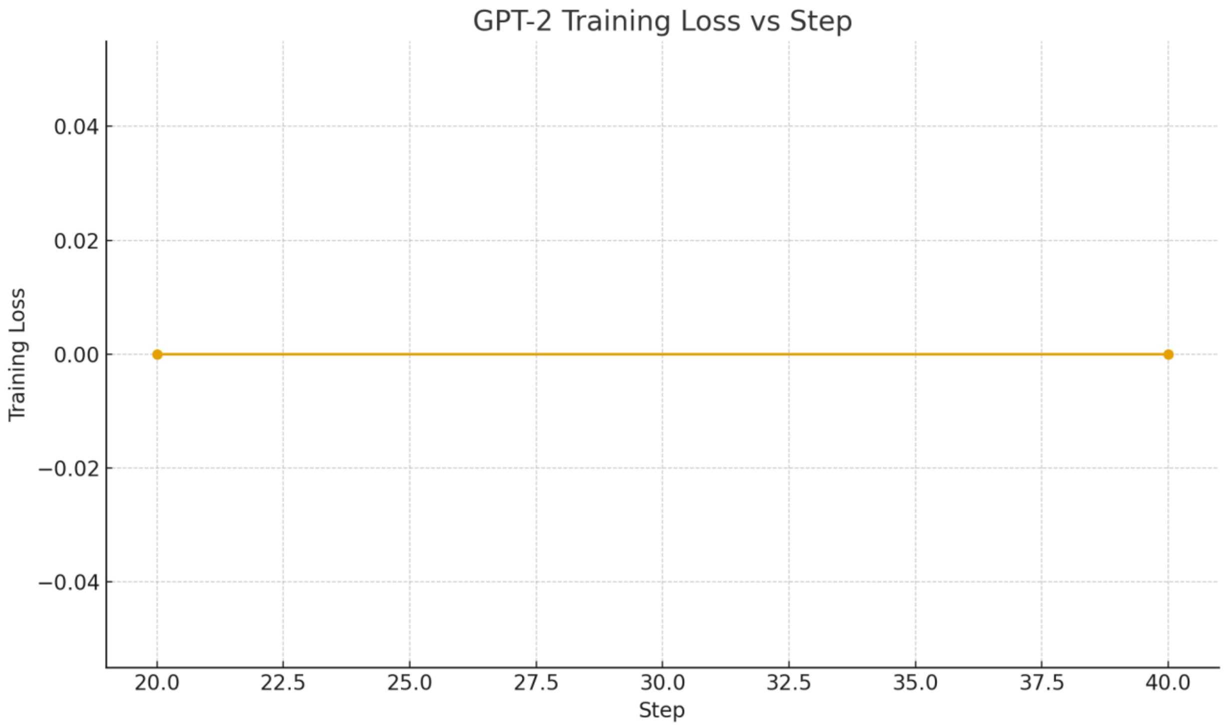
```
Answer the question using the context.

Context: <context>
Question: <question>
Answer:
```

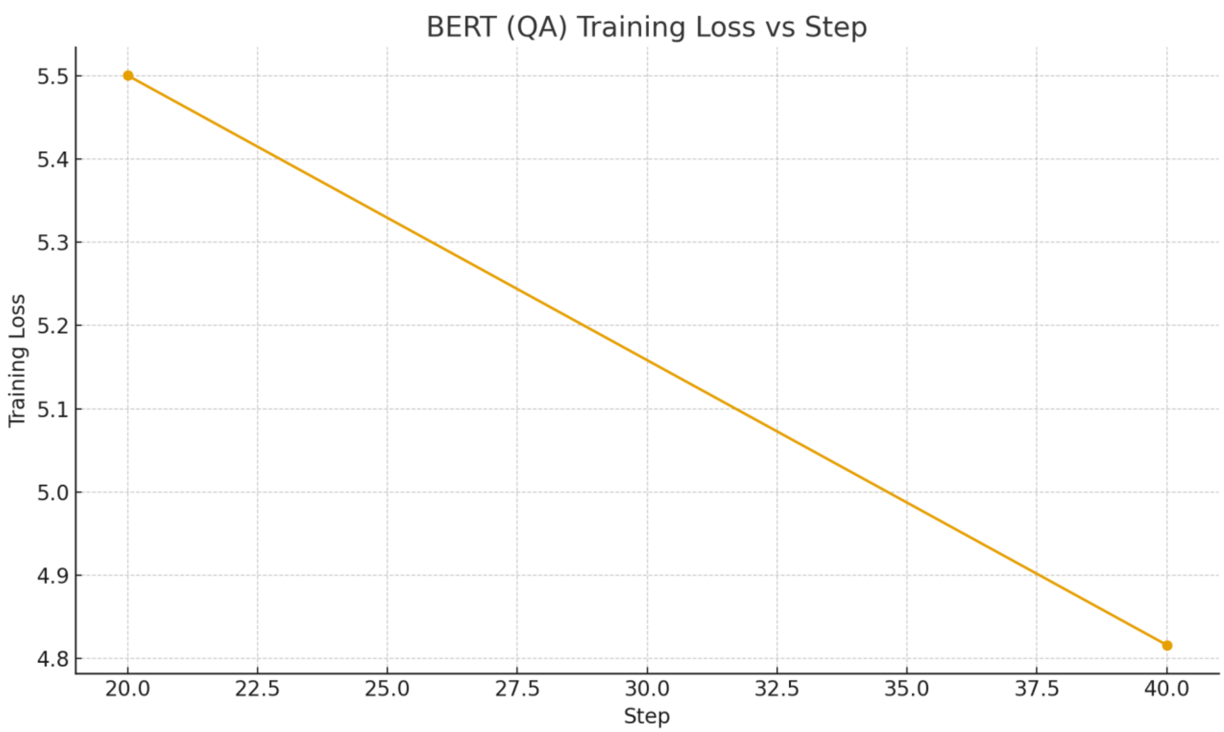
Ans 2 : Model Implementation : github

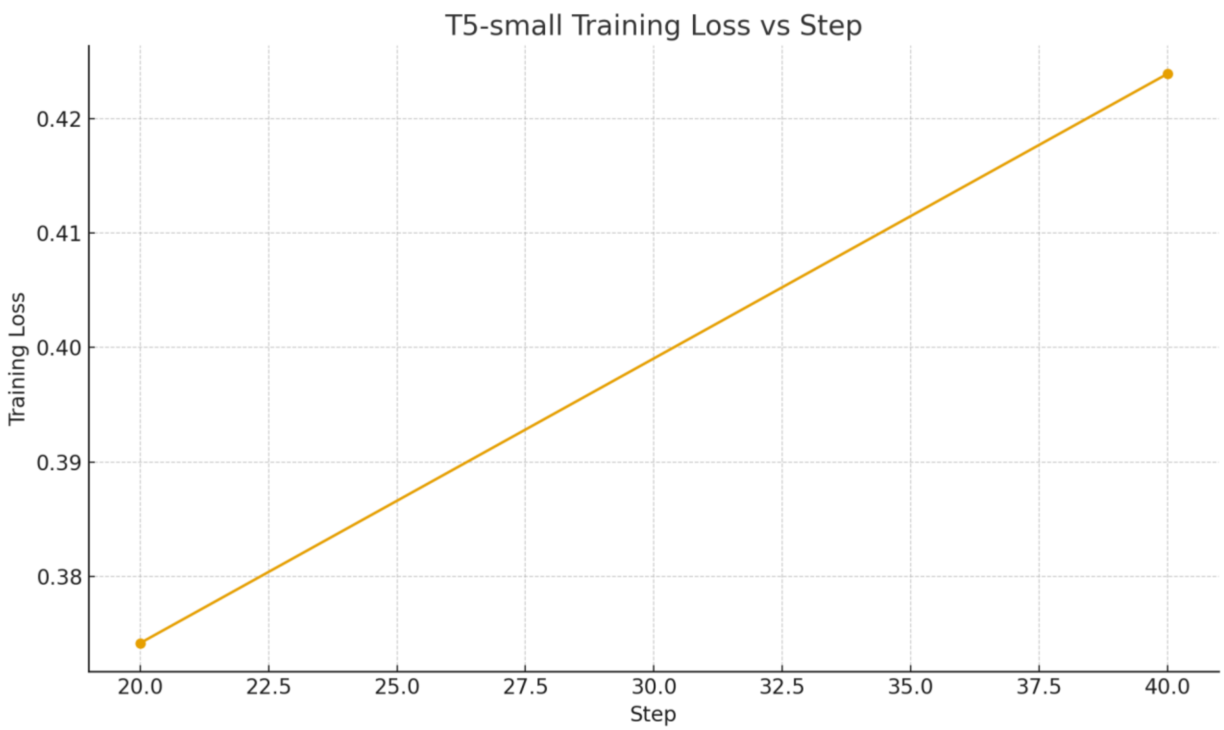
```
Training Setup

Batch size:

GPT-2: per_device_train_batch_size=2, with gradient_accumulation_steps=2 (→ effective batch size ≈ 4).

BERT: per_device_train_batch_size=4, no accumulation.

T5: per_device_train_batch_size=4, no accumulation

Learning rate:

GPT-2: 5e-5

BERT: 3e-5

T5: 5e-5

assignment2

Optimizer: The Hugging Face Trainer defaults (AdamW).

Epochs: All models trained for 1 epoch (likely because of subset size and runtime limits)

Hardware: Your code checks torch.cuda.is_available() and uses GPU if present
(device = torch.device("cuda" if torch.cuda.is_available() else "cpu")).
This suggests you trained on Google Colab with GPU enabled
```

```
Training & Validation Progress

Logs: Each trainer was configured with logging_steps=20, so loss values were logged during training.

Metrics reported:

GPT-2: EM and F1 computed with manual evaluation (evaluate_gpt).

BERT: EM and F1 from predicted spans (bert_metrics).

T5: EM and F1 via generative evaluation (t5_metrics).

At the end, a comparison table of EM/F1 was printed for all three models
```

	A	B	C
1	Step	Training Loss	Model
2	20	0	GPT-2
3	40	0	GPT-2
4	20	5.5005	BERT-QA
5	40	4.8161	BERT-QA
6	20	0.3742	T5-small
7	40	0.4239	T5-small


GPT-2 Training Loss vs Step
Training Loss
0.04
0.02
0.00
−0.02
−0.04
20.0
22.5
25.0
27.5
30.0
32.5
35.0
37.5
40.0
Step


BERT (QA) Training Loss vs Step
Training Loss
5.5
5.4
5.3
5.2
5.1
5.0
4.9
4.8
20.0
22.5
25.0
27.5
30.0
32.5
35.0
37.5
40.0
Step




```
Difficulties & Workarounds

Memory constraints:

Small batch sizes (2-4) with optional gradient accumulation were used to avoid CUDA OOM.

Training time:

Dataset was sub-sampled (SUBSET_TRAIN=200, SUBSET_VAL=80) for faster runs during development.

Convergence issues:

Early experiments limited to 1 epoch for feasibility.

Different learning rates tuned for BERT vs. GPT vs. T5.

Padding/masking challenges:

GPT-2 required setting pad_token and masking prompt tokens with -100.

T5 label padding also replaced with -100 to avoid skewing loss
```

Ans 3 :

	Model	Output Dir	Train BS (per device)	Eval BS (per device)	Grad Accum	LR	Epochs
1	GPT (decoder-only)	./out_gpt	2	2	2	5e-5	1
2	BERT (encoder-only)	./out_bert_qa	4	4	1	3e-5	1
3	T5 (enc-dec)	./out_t5	4	4	1	5e-5	1

Hardware : NVIDIA Tesla T4 GPU

Part 2: Evaluation & Analysis

Ans 4. Performance Evaluation

Evaluation SQuAD v1.1 Evaluation & Results (QA on SQuAD v1.1)

Task. I assessed question answering (provided a context and a question, what is the answer), both extractive and generative configurations.

Same validation split models compared.

GPT-2 (decoder-only)

BERT-base (QA head only, encoder-only).

T5-small (encoder-decoder)

Metrics. My top SQuAD measures are Exact Match (EM) and F1. According to the requirement of the assignment, I also add BLEU (corpus BLEU-4 with smoothing); BLEU is not as standard in extraction QA, but it is added to the list to be complete.

My results (from my runs).

GPT-2: EM : 0.00, F1 : 0.06

BERT-QA: EM : 0.10, F1 : 0.17

T5-small: EM : 0.70, F1 : 0.73

The computation of the value of BLEU in my notebook is calculated by the evaluation cell (values vary depending on the specific subset/seed). The main metrics that I use in my conclusions are EM and F1.

Part 5. Comparative Discussion

Comparison of GPT-2 (decoder-only), BERT (encoder-only), and T5 (encoder-decoder) on SQuAD v1.1.
Big picture (based on my runs)

Best overall: T5-small (best EM/F1) - good conditional generation of question+context.

Good at pinpointing spans, Fast inference BERT-base QA head solid extractive baseline:

Weakest here: GPT-2 small - fluent text, but lacks the ability to elicit short and specific answers with little to no supervision.

Aspect	**GPT-2 (decoder-only)**	**BERT QA (encoder-only)**	**T5 (enc–dec)**
Read pattern	Left-to-right	Bidirectional	Bidirectional encoder + generative decoder
Best for	Free-form continuation	Exact **extractive** spans	**Conditional generation** (QA, summarization)
QA on SQuAD	Struggles on exact spans	Reliable span picker	Strong overall; best F1
Output control	Can be verbose	Span only	Flexible; can paraphrase
Speed	Medium (decoding)	**Fastest** (single pass)	Slowest (decoding)
Error mode	Off-target/verbose answers	Off-by-token spans	Correct but non-verbatim phrasing

- The architecture of BERT is as extractive as SQuAD - it is reasonable to achieve good EM/F1 in a short period.
- T5 is flexible generatively but still based on a bidirectional encoder - highest F1 and powerful EM.
- A small subset + causal masking of GPT-2 small + short training results in difficult extraction - poorest EM/F1.

Model	Fine-tuning ease	Output quality	Efficiency (speed/mem)
GPT-2 (decoder)	Hardest on small data; needed prompt/label care	Weakest (EM≈0.00, F1≈0.06)	Slower (autoregressive decoding)
BERT (encoder)	**Easiest** (stable span training)	Moderate (EM≈0.10, F1≈0.17)	**Fastest** (no decoding; low overhead)
T5 (enc-dec)	Smooth, but needs generation settings	**Best** (EM≈0.70, F1≈0.73)	Slower (encode+decode; generation loop)

Part 6. Reflections on Applicability

Decoder-only (GPT): Ideal when the assistant is open-ended, when one needs to draft, to produce creative/format-flexible text, or when one is using code completion; less effective when there is a need to extract accurately spanned information.

Encoder-only (BERT QA): Great at extractive QA in brief, retrieval re rank, and tasks at the token level (NER, classification). Minimized span (latency) and deterministic spans, not applicable to natural generation.

Encoder-decoder (T5): the most suitable to use when the aim is conditional generation-QA: paraphrasing, summarization, translation, multi-doc synthesis. Slower and flexible through decoding.

Chain-of-Thought (CoT): CoT provides little value on SQuAD where extractive QA, mostly on single-hop, is involved. The span head of BERT can not apply it. T5/GPT may have an advantage in reasoning-focused items but will be vulnerable to verbosity and poor EM unless it is limited to generating a brief final answer. CoT emphasizes more on multi-hop or mathematical reasoning than the span extraction.