SECTION-A

Q 1: Analyze the architectural distinction between the Decoder-Only architecture (e.g., GPT, LLaMA) and the Encoder-Only architecture (e.g., BERT, RoBERTa) by focusing on the self-attention mechanism.

1. Explain precisely how the masking strategy differs between the two, specifically detailing its impact on the Query (Q) and Key (K) matrix multiplication.
2. How does this difference in information flow (bidirectional vs. unidirectional) directly determine the inherent strengths of each architecture for two distinct use cases: Text Classification (Encoder-Only) and Open-Ended Text Generation (Decoder-Only)?

Answer:

Encoder-Only models like BERT utilize a self-attention mechanism that allows attention layers to access **all words** in the input sentence at every stage, creating a bidirectional information flow. Their pre-training objective, often Masked Language Modeling, relies on reconstructing corrupted/masked text based on this full, bidirectional context. In contrast, Decoder-Only models like GPT are auto-regressive, employing masking so that attention layers access **only preceding words** in the sequence. This unidirectional flow makes the Decoder-Only architecture inherently suited for **text generation** and predicting the next token. The inherent bidirectionality enables Encoder-Only models to excel at **language understanding** tasks such as text classification.

Q2: The Encoder-Decoder architecture (e.g., T5, BART) acts as a powerful sequence-to-sequence model.

1. Compare the pre-training objective of T5 (casting all tasks as text-to-text) with the more focused Replaced Token Detection (RTD) objective used in ELECTRA. What is the fundamental difference in the *type of task* (classification vs. generation) each model performs during pre-training, and how does this affect their ability to learn rich bidirectional context?
2. Provide a technical advantage of the Encoder-Decoder design over Decoder-Only models for tasks like Abstractive Summarization and a disadvantage in terms of computational efficiency during inference for standard generative tasks.

Answer:

The T5 (Encoder-Decoder) objective is **generative**, unifying tasks as text-to-text generation, often pre-trained by replacing text spans for prediction. This contrasts with the ELECTRA (Encoder-Only) objective known as Replaced Token Detection (RTD), which is a **classification task** where the model acts as a discriminator to judge if tokens were replaced, utilizing context over all input tokens. An advantage of the Encoder-Decoder design for tasks like Abstractive Summarization is that the encoder can access the **full source text** to create a deep contextual representation, while the decoder is auto-regressive for output generation. Encoder-Decoder models are powerful models suited for generating new sentences conditioned on a given input, such as translation or summarization. However, this dual-component structure (encoder + decoder stack) generally implies a higher computational overhead during inference compared to single-stack Decoder-Only models.

Q3: LLM performance is governed by scaling laws. The landmark Chinchilla finding refined these laws, challenging the prior belief that model size should be prioritized over data size.

1. Explain the concept of optimal scaling, detailing the relationship it establishes between the number of model parameters (N) and the size of the training dataset (D).
2. Based on this principle, provide a quantitative comparison of the training corpus size (in tokens) used for the PaLM 540B model versus the LLaMA2 70B model, and discuss how the LLaMA family's scaling strategy aligns with the Chinchilla recommendations.

Answer:

The concept of optimal scaling, validated by the Chinchilla finding, dictates that for training to be compute-optimal, the model size (number of parameters, N) and the number of training tokens (D) should be scaled **equally**. This means that for every doubling of the parameter count, the training tokens should also be doubled to maximize efficiency under a given compute budget. Quantitatively, the large PaLM 540B model was pre-trained on a corpus of **780 billion tokens**, whereas the much smaller LLaMA2 70B model utilized a substantially larger corpus of **2 trillion tokens**. The LLaMA family's strategy of training a relatively smaller 70B model on a massive 2T token corpus demonstrates alignment with the Chinchilla recommendation to prioritize data scaling for compute-optimal performance.

SECTION-C

Q1: Reinforcement Learning from Human Feedback (RLHF) is the state-of-the-art technique for LLM alignment. Detail the RLHF process by describing the three distinct model components that are trained or utilized, and the purpose of each:

1. The Supervised Fine-Tuning (SFT) Model: What is its input data, and what is the primary purpose of this initial step?
2. The Reward Model (RM): How is it trained using comparative human feedback data, and what is its role in the RL loop?
3. The Policy Model (The final LLM): Explain the function of the Proximal Policy Optimization (PPO) algorithm in this context, specifically mentioning what the PPO objective maximizes, and why a Kullback–Leibler (KL) Divergence penalty is typically included.

Answer:

The Supervised Fine-Tuning (SFT) Model is the initial step, where a pre-trained LLM is fine-tuned on a dataset of labeled demonstrations that outline the desired model behavior. The primary purpose of this SFT stage is to align the language model with user intent across a broad spectrum of tasks. The Reward Model (RM) is trained using comparative human feedback data, specifically a dataset of human-ranked model outputs, allowing it to score different generated responses according to human preferences. The RM's critical role in the RL loop is to provide this alignment-based feedback, which is subsequently used to further tune the Policy Model. The Policy Model (the final LLM) is refined using Reinforcement Learning algorithms, such as Proximal Policy Optimization (PPO), which maximizes the reward signal derived from the Reward Model, and a Kullback–Leibler (KL) Divergence penalty is typically included to constrain the new policy from diverging too far from the original SFT model, promoting stability during training.

Q2: Evaluation metrics must match the task complexity. Compare the use of Exact Match (EM) and F1-score for the SQuAD/Natural Questions datasets with the requirements of reasoning benchmarks like MATH and GSM8K.

1. Why are the EM and F1-score metrics fundamentally inadequate for assessing the type of multi-step, symbolic, and procedural reasoning required by MATH and GSM8K?
2. The MATH dataset uses a 1-to-5 AoPS difficulty scale. Explain what the maximum rating of '5' represents, and detail the specialized language (beyond standard text) that is used in the MATH dataset for presenting both the problems and their solutions, which models must interpret.

Answer:

Exact Match (EM) and F1-score are widely used metrics for datasets like SQuAD and Natural Questions, where EM counts a prediction as correct only if it exactly matches a predefined reference text, and F1 measures lexical overlap. These metrics are fundamentally inadequate for assessing the type of multi-step, symbolic, and procedural reasoning required by benchmarks such as MATH and GSM8K, which focus on evaluating problem-solving ability and logical correctness. EM and F1 primarily detect similarity based on simplistic features like N-Gram overlap, which fails to verify the accuracy of multi-step reasoning processes or the calculated final answer derived through symbolic steps. Regarding the MATH dataset, the difficulty of each problem is rated according to the AoPS standards on a scale from 1 to 5, where the maximum rating of '5' represents the most difficult problems within that subject. Moreover, models must interpret specialized language, as both the problems and their step-by-step solutions are presented using **LATEX** and the **Asymptote vector graphics language**.

Q3: Bias is a significant safety and fairness challenge in LLMs.

1. Explain how pre-training data collection (e.g., the reliance on vast amounts of web data) inevitably introduces and propagates societal and demographic biases into LLMs.
2. Identify and describe the design structure of a specific Bias Evaluation Benchmark mentioned in the survey (e.g., BOLD or an alternative). How does this benchmark specifically measure bias across different demographic or identity groups, and what output metric (e.g., stereotype score, sentiment differential) is typically used to quantify the problem?

Answer:

Large Language Models (LLMs) are pre-trained on massive, often web-scale, text corpora which serve as the foundation for their language understanding capabilities. Pre-training data collection inevitably introduces and propagates societal and demographic biases because the corpora, sourced from the internet, books, and other resources, inherently reflect existing human biases and stereotypes (often western cultures). This propagation is a critical concern, as LLMs trained this way often exhibit unintended behaviors, including generating biased content. While addressing ethical concerns, including bias, is noted as an active area of research to ensure models are fair and unbiased,

BOLD (Bias in Open-Ended Language Generation Dataset) evaluates bias by forcing the model to *generate* text, not just rank or classify it. It measures bias by **identity-conditioned prompts** where Each prompt explicitly references a demographic or identity group. **Matched counterfactuals**, where BOLD includes parallel versions of prompts where only the identity term changes. And **open-ended generation,** where model completes the prompt freely. It relies on sentiment-based differentials computed over generated text and does not use a handcrafted stereotype list.

SECTION-B

Q1: The Retrieval-Augmented Generation (RAG) framework addresses a fundamental limitation of base LLMs.

1. Identify the two primary, interconnected limitations of a standard, statically-trained LLM that RAG is designed to mitigate (one related to knowledge currency/recency, the other related to faithfulness/source attribution).
2. Describe the RAG process pipeline, detailing the function of the Retriever component (e.g., how it queries external data) and the Generator component (how it uses the retrieved context).

Answer:

The Retrieval-Augmented Generation (RAG) framework addresses two major, interconnected limitations of standard, statically-trained LLMs: first, their inherent **lack of up-to-date knowledge or access to use-case-specific information**, which is static based on their pre-training data. Second, RAG mitigates the issue of **hallucination**, which is the generation of content that is often untruthful, nonsensical, or unfaithful to any provided source material, a problem rooted in the LLM's fundamental probabilistic nature. In the RAG process pipeline, the system starts by extracting a query from the user's input prompt. The **Retriever component** uses this query to search and retrieve relevant information from an external knowledge source, such as a knowledge graph or search engine. This relevant information is then utilized by the **Augmentation component** to modify the original prompt by injecting the retrieved context. Finally, the **Generator component** (the LLM) processes this augmented prompt and produces a final response grounded in the external information, which directly helps in minimizing hallucination risks.

Q2: The modern RAG pipeline critically relies on Vector Databases (e.g., Faiss, Milvus, Qdrant).

1. Explain the concept of vector embedding in this context: How is external knowledge (documents, text chunks) prepared and stored within a vector database? What is the role of an Embedding Model in this process?
2. Describe the core function of a vector database, which is similarity search. Why is the use of dense vectors and Approximate Nearest Neighbors (ANN) search algorithms essential for the low-latency retrieval of relevant context in large-scale RAG systems?

Answer;

Vector embedding, in this context, relies on the principle that words mapped to low-dimensional continuous vectors define a hidden space where the **semantic similarity** between inputs can be calculated based on the distance between their vectors. While the sources mention that external knowledge is handled, it chunks the content in to fixed size tokens (as retrieval works at chunk-level), then it is passed to embedding model which converts these chunks into vectors and then they are stored as numerical vectors with optional ids in DB.

Vector databases such as Faiss, Milvus, and Qdrant are specifically built to power embedding **similarity search**. For instance, Faiss specializes in providing **efficient similarity search** and clustering of dense vectors. This reliance on **dense vectors** and specialized search (such as for large-scale, high-dimensional data) is essential for achieving the **low-latency retrieval** of relevant context necessary to effectively augment the LLM's response.

Q3: While RAG augments the LLM externally, Prompt Engineering guides its internal behavior.

1. Differentiate between Zero-Shot and Few-Shot prompting techniques, and explain how the latter technique leverages the model's in-context learning (ICL) capability.
2. Connect RAG with effective Prompt Engineering. When a retrieved document chunk is inserted into the prompt as *context*, which category of prompting (Zero-Shot or Few-Shot) does this augmented prompt structurally resemble, and why is this combination highly effective for grounding LLM outputs?

Answer:

Zero-Shot prompting involves giving the LLM an instruction or question without any specific examples, sometimes instructing the model to "think step by step". Conversely, **Few-Shot prompting** is achieved by providing the model with a small set of examples illustrating the desired task or reasoning process directly within the prompt. The latter technique effectively leverages the LLM's **in-context learning (ICL)** capability, enabling the model to learn a new task solely from these prompt examples at inference time. When a retrieved document chunk from RAG is inserted into the prompt as context, the resulting augmented prompt structurally resembles the **Few-Shot** approach, as the chunk acts as critical input data or guiding material, much like an example. This combination is highly effective because grounding the LLM's generation in factual, external knowledge explicitly mitigates the risks of hallucination.