Exercise 1 - Advanced Natural Language Processing - Moral Bootbool - 208716217

Question 1-

1. QA
    a. **Boolq** - QA over Wikipedia paragraphs with yes/no answers. **Why intrinsic**: long context, sentence semantics and understanding logic to get the right answer. https://huggingface.co/datasets/google/boolq
    b. **Cosmos_qa -** Cosmos QA is a large-scale dataset of 35.6K problems that require commonsense-based reading comprehension, formulated as multiple-choice questions. **Why intrinsic:** Common sense based but also requires vase understanding. https://huggingface.co/datasets/allenai/cosmos_qa.
    c. **DPR (Discrete Reasoning Over Paragraphs) -** Multi-hop reasoning across multiple documents. **Why intrinsic:**combine word meanings, understand the flow of ideas, and make sense of information across sentences.https://www.kaggle.com/datasets/jeromeblanchet/drop-requiring-discrete-reasoning-over-paragraphs
2. Inference-time scaling
    a. Methods: In some of the parallelizable answers - I answered as can we perform several tasks parallel but I'm not sure this was the meaning. Some models like Few shots - we run only one prompt and we check the result - we can run some few shots with different examples and do it parallel than compare the results. But in some examples like Self consistency decoding we can generate each answer independently and compare them afterwards which is more parallel in my opinion.
        i. Prompt Engineering:
            1. Description: Optimizing the writing of the prompt - so the model would understand the task better.
            2. Advantages: No need to retrain the model.
            3. Bottleneck: Need manual optimization.
            4. Parallelizable: Yes - can test multiple prompts in parallel - no dependencies.
        ii. Few-shots:
            1. Description: Writing within the prompt few examples of how to solve the task.
            2. Advantages: No training - the model adapt to the task from the examples.
            3. Bottleneck: The context may be long.
            4. Parallelizable: Yes - different inputs with examples can run parallel.
        iii. Chain-of-Thought (COT) prompting:
            1. Description: Ask the model for a step-by-step explanation of its reasoning before giving a final answer.

2. Advantages:Good for more complex reasoning tasks - like math, the step by step explanation holding the model “account” for each step.
3. Bottleneck: Longer outputs and increased runtime.
4. Parallelizable: Yes - for different queries.

iv. Self-consistency Decoding:
1. Description: Run multiple COT and return the most frequent answer..
2. Advantages: Less random and more accurate (more tries).
3. Bottleneck: Running the model many times - expensive.
4. Parallelizable: Yes - can ran multiple runs at the same time.

v. Tool Augmentation:
1. Description: allowing the model to use third parties tools like calculators or search engines.
2. Advantages:Models more stable and less hallucinations..
3. Bottleneck:Longer running time.
4. Parallelizable: Depends on the paralleliability of the tool.

vi. Retrieval-Augmented Generation (RAG)
1. Description: Retrieve information from a knowledge base and feed it to source.
2. Advantages: :Models more stable, reliable and less hallucinations.
3. Bottleneck: More runtime and memory use.
4. Parallelizable: Yes - for different queries.

b. Suppose you must solve a complex scientific task requiring reasoning, and you have access to a single GPU with large memory capacity. Which method would you choose, and why?
I would use COT with Self consistency decoding : The COT suitable for the scientific problem, which requires a step by step reasoning process. COT encourage the model to “think aloud”, which can mimic human reasoning and improve problem solving (lecture 3). Adding the Self consistency make the results more stable, and since we have large GPU and no runtime limit we can run the queries on the single GPU.

Question 2 -
On the validation and test we get -

1. epoch_num: 3, lr: 5e-05, batch_size: 16, eval_acc: 0.8701, test_acc: 0.3351
2. epoch_num: 2, lr: 3e-05, batch_size: 8, eval_acc: 0.8652, test_acc: 0.3351
3. epoch_num: 1, lr: 0.0001, batch_size: 32, eval_acc: 0.7328, test_acc: 0.6649
4. epoch_num: 2, lr: 0.03, batch_size: 64, eval_acc: 0.6838, test_acc = 0.3350

We can see that though model 1 had the highest accuracy on validation, it had low accuracy on test and the model 3 with mid performance on the validation had a high test accuracy. This suggests that there is similarity between the train and validation set and that the models might overfit to the train, model 3 with the lowest epoch.

I looked into the samples where model 3 succeed and model 1 fails - I noticed it happened in sentences with words synonyms - for example - the model failed to understand that “gain” and “increase” has the same meaning, it also struggled with slight word order change.
For example -
S1: “The increase reflects lower credit losses and favorable interest rates.”
S2: “The gain came as a result of fewer credit losses and lower interest rates.”
Model 3:Paraphrase
Model 1: Not paraphrase