Ex1 - Stav Perez

Tuesday, April 29, 2025 6:31 PM

Github Link:
https://github.com/stavperez/anlp_ex1.git

Open questions

Part 1 Q1

1. SQuAD: contextual comprehension
SQuAD measures a model's ability to understand a single passage and locate a precise answer span within it. This requires core skills like syntactic parsing, semantic role labeling, and coreference resolution — all intrinsic to language understanding.

2. HotpotQA: Multi-hop reasoning
HotpotQA requires reasoning across multiple documents to answer a question. This tests the model's ability to track entities, infer implicit connections, and chain together multiple facts — fundamental capabilities for deep semantic comprehension.

3. DROP (Discrete Reasoning Over Paragraphs): Numerical reasoning and discrete operations
DROP evaluates a model's ability to perform operations like addition, subtraction, sorting, and comparison based on textual cues. This tests symbolic reasoning and interpretation of quantitative language — intrinsic to language understanding where numbers and logic are embedded in text.

Part 1 Q2 a
Self-Consistency
- Description: Self-consistency involves sampling multiple outputs from a model using stochastic decoding and then aggregating the results by majority vote.
- Advantages: Reduces the impact of randomness or hallucinations, improves robustness and reliability of answers in reasoning tasks.
- Computational Bottlenecks: Repeated forward passes for multiple outputs significantly increase inference time and GPU use.
- Parallelization: Yes, each forward pass is independent and can be parallelized easily across GPUs or cores.

Verifiers
- Description: Uses a secondary model (a "verifier") to assess or critique the answer generated by the main model.
- Advantages: Can catch subtle logical or factual errors, allows for dynamic feedback and answer filtering.
- Computational Bottlenecks: Adds extra computation per generation, requires a well-tuned verifier model, possibly trained separately.
- Parallelization: Verifier checks can be done in parallel but only after generation.

Many Outputs from Small Models
- Description: Instead of relying on a single large model to do complex reasoning, use many smaller models to answer sub-questions or vote.
- Advantages: Better results for specific tasks.
- Computational Bottlenecks: Coordination overhead between models, total compute may still be high depending on how many small models are used.
- Parallelization: Yes, all small models can run concurrently on different sub-tasks.

O1 Model: Planning, Backtracking, and Self-Evaluation
- Description: A model that plans an answer strategy, decomposes complex problems into sub-questions, delegates to another model, backtracks when answers don't cohere, performs self-evaluation after each step.
- Advantages: Mimics human reasoning, dynamic - adjusts strategy based on errors, and flexible because

it can handle multi-hop and multi-context tasks.
- Computational Bottlenecks: Requires a looped execution and high token usage due to planning and memory of previous steps.
- Parallelization: Partial. Specific tasks of atomic questions may be parallelized, but the whole process steps are inherently sequential due to dependencies.

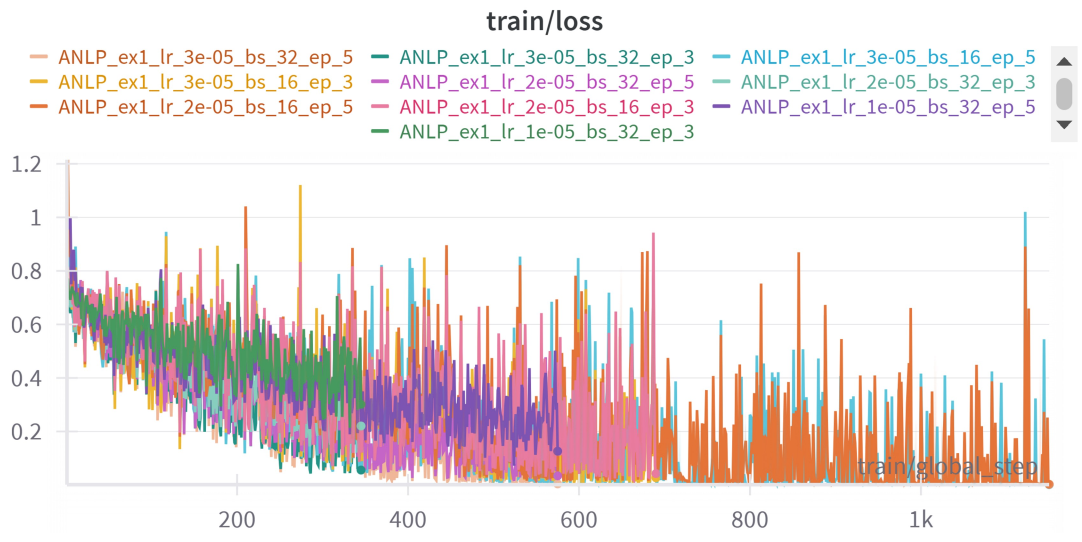
Part 1 Q2 b

If I had to choose one method to solve a complex scientific task using a single GPU with a lot of memory, I would go with the O1 model approach. This method lets the model break down the big problem into smaller parts, plan how to solve them, and fix mistakes along the way. It's more flexible and works better for reasoning through multi-step tasks than just generating one answer in a single shot. Even though it's slower because the steps depend on each other, the high memory allows it to keep track of everything and make smarter decisions. Plus, it can still use elements from other methods, like double-checking answers or using multiple outputs when needed.

Part 2 Q2

- After testing multiple combinations of hyperparameters, I compared how well each model performed on both the validation and the test sets. The configuration that achieved the highest validation accuracy did not achieve the highest test accuracy. Two models reached the top validation accuracy of 87.25%: one trained for 3 epochs with a learning rate of 5e-5 and batch size 16, and another trained for 4 epochs with a learning rate of 3e-5 and batch size 32. However, in the test set both of them only reached a test accuracy of 82.61%, which was not the best. The best test performance came from a model trained for 3 epochs with a learning rate of 3e-5 and a batch size of 32, that achieved a test accuracy of 85.51%.

- The best-performing configuration on the test set was trained for 3 epochs with a learning rate of 3e-5 and a batch size of 32. The worst-performing configuration in the test accuracy was trained for 3 epochs with a higher learning rate (5e-5) and a smaller batch size (16). This difference likely reflects the impact of both learning rate and batch size on how well the model generalizes. A smaller learning rate and larger batch size can lead to more stable and generalizable learning, while a larger learning rate can sometimes cause the model to overfit or learn noisy patterns.


train/loss
ANLP_ex1_lr_3e-05_bs_32_ep_5
ANLP_ex1_lr_3e-05_bs_32_ep_3
ANLP_ex1_lr_3e-05_bs_16_ep_5
ANLP_ex1_lr_3e-05_bs_16_ep_3
ANLP_ex1_lr_2e-05_bs_32_ep_5
ANLP_ex1_lr_2e-05_bs_32_ep_3
ANLP_ex1_lr_2e-05_bs_16_ep_5
ANLP_ex1_lr_2e-05_bs_16_ep_3
ANLP_ex1_lr_1e-05_bs_32_ep_5
ANLP_ex1_lr_1e-05_bs_32_ep_3
1k
train/global_step