Gal Cesana 318510633

Advanced NLP course 67664 The Hebrew University of Jerusalem

ANLP – Exercise #1

Section 1 – Open Questions

1. QA datasets that annotate *intrinsic* concepts:

- **SNLI (Stanford Natural Language Inference)**: Given a pair of sentences—a premise and a hypothesis—decide whether the hypothesis is entailed by, contradicts, or is neutral with respect to the premise. Success requires the model to grasp the underlying semantic relations (entailment, contradiction, neutrality) between two texts, demonstrating an understanding of logical connections between sentences.
- **Coreference Datasets (e.g. the Winograd Schema Challenge):** Identify the referent of a pronoun or noun phrase in context (e.g. in "The Dog ate the kid's homework because he was hungry" determine whether "he" refers to the dog). Solving coreference needs deep semantic interpretation of how entities relate in text, showing the model can link pronouns or mentions to the correct entities purely from linguistic cues.
- **Entity-Linking Datasets (e.g. Wikidata-based)**: Questions require resolving intra-paragraph coreference ("What did *he* do after the game?"), so success hinges on the intrinsic ability to track entity mentions across clauses. Accurately selecting and linking entities tests the model's ability to map textual mentions to semantic entities—requiring both precise text understanding and alignment with external knowledge, a core aspect of language comprehension.

2. a. Inference-time scaling methods

i. Self-Consistency

- Description:
  Instead of taking a single chain-of-thought (CoT) trace, you sample many independent reasoning paths and then take the most frequent final answer (majority vote).
- Advantages:
  - Smooths over "unlucky" chains that go off-track.
  - Often yields higher accuracy than a single greedy CoT trace.
- Computational Bottlenecks:
  - Inference cost: you must run N full forward passes to get N chains.
  - Memory & I/O: storing all intermediate CoT outputs before voting.
- Parallelizability:
  - Yes. Each chain sample is independent, so you can batch or shard them across GPUs/TPUs.

ii. **Verifiers**

- Description:
  Generate one or more candidate answers (with or without CoT), then feed each (question + answer) into a secondary "verifier" model that scores its correctness. Finally, pick the answer with the highest verifier score.
- Advantages:
  - Catches mistakes that the generator makes.
  - Can be fine-tuned specifically to spot logical or factual errors.
- Computational Bottlenecks:
  - Double inference: one pass to generate, another to verify.
  - Model switching overhead: moving data between generator and verifier.
- Parallelizability:
  - Yes - Partially. You can verify multiple candidates in parallel, but you still need to wait for both generator and verifier outputs.

iii. **Chain of Thought (CoT) prompting**

- Description:
  You prepend a few exemplars in your prompt where each example shows not only the question and answer but also its intermediate reasoning steps ("chain of thought"). At inference, the model is encouraged to emulate that step-by-step reasoning before producing a final answer.
- Advantages:
  - Dramatically improves performance on tasks requiring multi-step or compositional reasoning.
  - Often yields more interpretable model outputs, since you can inspect the generated rationale.
- Computational Bottlenecks:
  - Context-window usage**:** CoT exemplars consume significant prompt space, reducing room for longer questions or more examples.
  - Token generation cost: You generate extra tokens (the reasoning steps) for every example in both prompt construction and model output.
- Parallelizability:
  - No. The reasoning steps must be generated sequentially—each step conditions on the previous one—so you cannot parallelize the token-by-token chain.

2. b. The method I would choose for solving a complex scientific task on a single GPU with large memory is <u>Self-Consistency</u>.
While CoT reasoning alone can guide the model to break down complex problems step by step, it remains vulnerable to errors from a single sampled reasoning path.
Verifiers, though helpful in evaluating answer correctness, add architectural and computational complexity by requiring an additional model and a two-stage pipeline.
In contrast, Self-Consistency leverages multiple independent CoT reasoning paths and selects the most frequent answer, which statistically improves robustness and accuracy.

This method makes efficient use of a large-memory GPU by allowing batch processing of multiple reasoning chains, enhancing both speed and performance without the need for additional model components.
Therefore, Self-Consistency offers the best trade-off between accuracy, efficiency, and implementation simplicity in this setting.

Section 2 – Programming Exercise

GitHub Repo Link

Each model's accuracies:

ep3_lr5e-05_bs32

- val acc = 0.867647
- test acc = 0.8382608695652174

ep2_lr2e-05_bs16

- val acc = 0.828431
- test acc = 0.8347826086956521

ep2_lr0.00015_bs16

- val acc = 0.845588
- test acc = 0.8249275362318841

As we can see, the **ep3_lr5e-05_bs32** configuration achieved both the best accuracy on the validation set and the test set but only by a small margin.

Comparing the best and worst performance models:

We can see that the weaker **ep2_lr2e-05_bs16** model consistently stumbles on cases that demand more than surface matching—deep paraphrases with little word overlap, numerical or factual inference, subtle shifts in which entity or clause is in focus, and multi-step entailments or negation nuances—whereas the stronger **ep3_lr5e-05_bs32** configuration, with its larger batch size and slightly higher learning rate, generalizes better to these complex semantic patterns and so achieves higher accuracy on such edge cases.