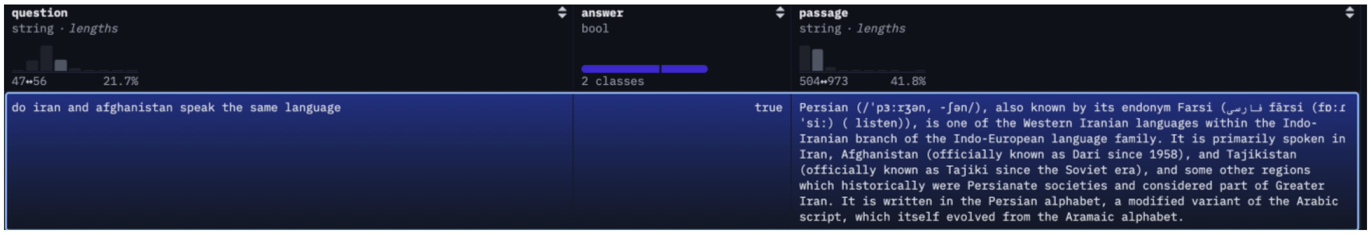
ANLP – Ex 1

By Yuval Cohen (208305052)

Open Questions

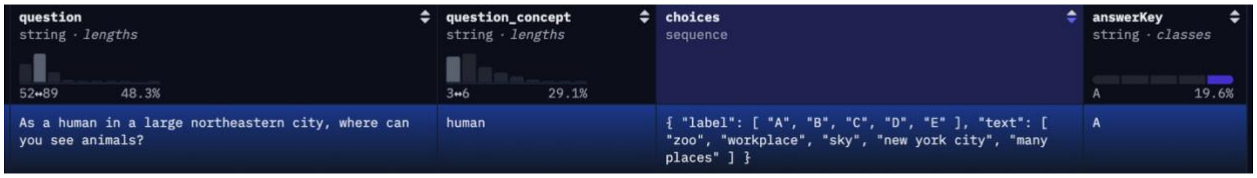
1. QA datasets that use QA to annotate intrinsic concepts:
    a. BoolQ - This dataset involves yes/no questions based on a passage, where the model must determine if the passage supports the question's premise.



question string · *lengths* 47↔56 21.7%	answer bool 2 classes	passage string · *lengths* 504↔973 41.8%
do iran and afghanistan speak the same language	true	Persian (/ˈpɜːrʒən, -ʃən/), also known by its endonym Farsi (فارسی fārsi (fɒːɾ ˈsiː) (listen)), is one of the Western Iranian languages within the Indo-Iranian branch of the Indo-European language family. It is primarily spoken in Iran, Afghanistan (officially known as Dari since 1958), and Tajikistan (officially known as Tajiki since the Soviet era), and some other regions which historically were Persianate societies and considered part of Greater Iran. It is written in the Persian alphabet, a modified variant of the Arabic script, which itself evolved from the Aramaic alphabet.

It tests semantic understanding and textual entailment as intrinsic properties because it evaluates the model's ability to comprehend and reason about the meaning and consistency of text without external knowledge.

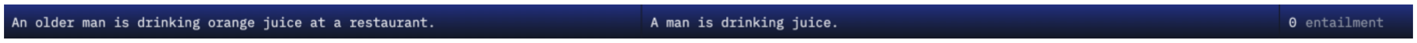
    b. CommonsenseQA - This dataset contains multiple-choice questions about everyday scenarios requiring commonsense knowledge.



question string · *lengths* 52↔89 48.3%	question_concept string · *lengths* 3↔6 29.1%	choices sequence	answerKey string · *classes* A 19.6%
As a human in a large northeastern city, where can you see animals?	human	{ "label": ["A", "B", "C", "D", "E"], "text": ["zoo", "workplace", "sky", "new york city", "many places"] }	A

It tests semantic inference as an intrinsic property since the model must reason about implicit knowledge embedded in language to understand social and physical relationships in the question's context.

    c. SNLI - This dataset requires determining if a hypothesis entails, contradicts, or is neutral with respect to a premise.



An older man is drinking orange juice at a restaurant.	A man is drinking juice.	0 entailment

It assesses semantic entailment as an intrinsic property by requiring the model to analyze sentence meanings and their logical relationships within the provided text.

2. (a) *As scaling model-sizes slowed, Chain-of-Thought (CoT) methods gained prominence, enabling performance improvements during inference through inference-time scaling. CoT is central to these methods, often underpinning or inspiring them. Therefore, I will cover all methods from the lecture as required, sometimes noting their ties to CoT, and provide descriptions, advantages, computational bottlenecks, and parallelization potential.*

   Chain-of-Thought (CoT) Prompting:
   CoT prompting involves instructing the model to generate step-by-step reasoning to solve a problem before providing the final answer, such as breaking down a mathematical or logical query into intermediate steps. A key aspect of CoT is that the length of the reasoning chain directly scales inference-time compute, as longer chains require more tokens and processing time. Its advantages include improved accuracy on complex tasks by making the model's thought process explicit and enabling better handling of multi-step problems. The computational bottleneck is the increased token generation, which consumes significant memory and processing power, especially for long chains. CoT cannot be easily parallelized, as each reasoning step depends on the previous one, requiring sequential computation.

   Self-Consistency:
   Self-consistency involves generating multiple reasoning paths for the same prompt, often using CoT, and selecting the most frequent or consistent answer through majority voting. Its advantages include enhanced robustness and accuracy, particularly for problems with multiple valid reasoning approaches, by mitigating errors from single-path reasoning. The computational bottleneck is the need for multiple forward passes through the model, significantly increasing compute and memory usage. This method can be parallelized, as each reasoning path can be generated independently on separate processors.

   Verifiers:
   After generating multiple CoT reasoning chains, verifiers evaluate these chains using mechanisms like rule-based checks or trained verifier models to select the most accurate answer, often through rejection sampling or best-of-N approaches. The advantages include higher accuracy by filtering out incorrect reasoning and the ability to incorporate domain-specific validation (e.g., unit tests for code). The computational bottleneck lies in generating multiple chains and running the verification process, which adds significant compute overhead, especially for complex verifiers. This method can be parallelized, as generating chains and verifying them can be distributed across multiple processors.

Advanced Reasoning Strategies:
Advanced reasoning strategies, as demonstrated by the o1 model, encompass planning, backtracking, and self-evaluation, which collectively enhance reasoning by iteratively refining the model's approach during inference. Planning involves structuring a solution approach before solving, increasing compute by adding early reasoning. Backtracking includes revisiting and recomputing incorrect steps, such as correcting a mathematical derivation, which scales compute through iterative retries. Self-evaluation involves the model assessing its own reasoning, such as evaluating multiple-choice options, adding compute via additional checks. These strategies, often built on CoT, offer the advantage of improved reasoning accuracy by mimicking human-like problem-solving, particularly for complex tasks. The computational bottleneck is the iterative nature of these processes, which significantly increases token generation and processing time. These strategies are generally sequential and difficult to parallelize, as each step depends on prior reasoning outcomes.

(b) I would choose Advanced Reasoning Strategies, which include planning, backtracking, and self-evaluation. As described in section (a), these strategies excel by iteratively revisiting incorrect steps and evaluating reasoning, making them highly effective for complex scientific problems like mathematical derivations or logical analysis. The large memory capacity of the single GPU supports the increased token generation and iterative computations required, accommodating longer reasoning chains and multiple iterations without memory constraints. Self-Consistency and Verifiers require generating multiple chains, which are less efficient on a single GPU due to sequential processing limitations, despite their parallelization potential. In contrast, Advanced Reasoning Strategies are sequential, aligning well with a single GPU as parallelization offers little benefit. They balance computational demands with deep reasoning, making them the best fit for a complex scientific task on a single GPU.

Programming Exercise

1. Configuration Comparison:
   Below are the configurations I selected, along with their accuracy scores on the validation and test sets (as recorded in res.txt):
   a. epoch_num: 3.0, lr: 2e-05, batch_size: 16, eval_acc: 0.8309
   b. epoch_num: 4.0, lr: 3e-05, batch_size: 8, eval_acc: 0.8529
   c. epoch_num: 2.0, lr: 1e-05, batch_size: 32, eval_acc: 0.7745
   d. epoch_num: 5.0, lr: 3e-05, batch_size: 8, eval_acc: 0.8578

As required, I computed their accuracies on the test set:
a. Test accuracy for model_epochs_3.0_lr_2e-05_batch_16: 0.8284
b. Test accuracy for model_epochs_4.0_lr_3e-05_batch_8: 0.8539
c. Test accuracy for model_epochs_2.0_lr_1e-05_batch_32: 0.7641
d. Test accuracy for model_epochs_5.0_lr_3e-05_batch_8: 0.8394

While Config d outperformed Config b by 0.0049 on the validation set, the test set results showed a more significant difference, with Config b surpassing Config d by 0.0145. This suggests that the additional epoch in Config d might have led to slightly reduced generalization on the test set, causing it to perform worse than Config b. In contrast, Config b's test accuracy was slightly higher than its validation accuracy, indicating that its hyperparameters might be better balanced for this task compared to the other configurations.

2. Qualitative Analysis:
I compared the best performing configuration – epoch_num=5.0, lr=3e-05, batch_size=8, (as Config d), with the worst - epoch_num=2.0, lr=1e-05, batch_size=32 (as Config c).

To make the comparison easier, I created the script *qualitative_analysis.py*, which produced the following results:

"Found 55 examples where the best model succeeded but the worst failed:
Overpredictions by worst model (predicted 1, true label 0): 47
Underpredictions by worst model (predicted 0, true label 1): 8

- Sentence 1: Magnarelli said Racicot hated the Iraqi regime and looked forward to using his long years of training in the war .
  Sentence 2: His wife said he was " 100 percent behind George Bush " and looked forward to using his years of training in the war .
  Ground Truth: 0 (not equivalent)
  Config c : 1 | Config d: 0
  … "

First, It seems like the False Positive count (predicted 1 when the true label was 0) is the much more dominant error, greater by almost 6 times the other one (47 vs. 8), showing a clear trend in Config c's mistakes.
Let's examine some examples of this kind of error to understand why Config c struggles:

(all example can be found under '*differing_examples.txt*')
Example 11:

Sentence 1: " Sanitation is poor ... there could be typhoid and cholera , " he said .

Sentence 2: " Sanitation is poor , drinking water is generally left behind . . . there could be typhoid and cholera . "

True Label: 0

Best Model Prediction: 0

Worst Model Prediction: 1

Sentence 2 adds “drinking water is generally left behind,” which changes the meaning, but Config c might have focused on the shared “Sanitation is poor… there could be typhoid and cholera” and ignored the extra detail.

Example 4:

Sentence 1: No dates have been set for the civil or the criminal trial .

Sentence 2: No dates have been set for the criminal or civil cases , but Shanley has pleaded not guilty.

True Label: 0

Best Model Prediction: 0

Worst Model Prediction: 1

Once again, Sentence 2 adds “but Shanley has pleaded not guilty,” which introduces new information not present in Sentence 1. Config c might have focused on the shared “No dates have been set for the criminal or civil cases” and ignored the extra clause.

Example 104:

Sentence 1: " I don 't know if the person I 'm talking to now may end up being someone else at another time that may not follow the rules , " Parrish said .

Sentence 2: " I don 't know whether the person I 'm talking to now may end up being someone else , " Parrish said .

True Label: 0

Best Model Prediction: 0

Worst Model Prediction: 1

Sentence 2 omits the latter part (“at another time that may not follow the rules”), changing the meaning. Config c likely focused on the shared beginning and ignored the omission.

From these examples, we can suspect that to make the worse config fail, it’s better to add or omit a certain detail while keeping both sentences largely similar in structure or wording. This seems to make it hard for Config c to catch the differences, as it tends to focus on surface-level similarities and overlooks subtle changes in meaning. Config c’s limited training (only 2 epochs) and lower learning rate (1e-05) compared to Config d’s 5 epochs and higher learning rate (3e-05) might explain why it struggles to detect these nuances, leading to a higher rate of False Positives.