Open Questions:

1) Examples of QA Datasets for Intrinsic Language Understanding Tasks:

1. QA-SRL (Question-Answer Driven Semantic Role Labeling)

Data: Sentences annotated with natural language questions and answers capturing the semantic roles of predicates (e.g., who did what to whom, when, and where).

Task: Answer predicate-specific questions such as identifying agents, patients, and temporal or locative information within sentences.

Why intrinsic: It evaluates a model's ability to understand the underlying semantic structure of sentences, linking verbs to their arguments. These abilities reflect core intrinsic language skills because they rely on grasping syntactic and semantic relationships within text, independent of external factual knowledge.

---

2. Quoref

Data: Wikipedia paragraphs with questions requiring resolution of coreference links between expressions (e.g., pronouns and their referents).
Task: Answer questions that require identifying which entity a pronoun or noun phrase refers to.
Why intrinsic: It challenges a model to resolve coreference — that is, to determine which entities different pronouns or noun phrases refer to across a passage. This is an intrinsic skill because it requires understanding the internal structure and coherence of the text, tracking entities over multiple sentences, and interpreting meaning based on linguistic cues, not external world knowledge.

---

3. SQuAD (Stanford Question Answering Dataset)

Data: Paragraphs from Wikipedia paired with fact-seeking questions and annotated answer spans indicating the correct location of the answer within the text.

Task: Given a passage and a question, extract the exact span of text that answers the question.
Why intrinsic: SQuAD evaluates a model's ability to perform contextual comprehension, textual inference, and information extraction — all intrinsic skills. The task requires understanding the internal linguistic structure of the passage to identify answer boundaries, make inferences, and distinguish relevant vs. irrelevant information, all without relying on external world knowledge.

---

2. a) Inference-Time Scaling Methods

Inference-time scaling refers to improving a model's performance at inference time without retraining, by allocating more computational resources — such as generating multiple responses, extending reasoning chains, or applying verification. The goal is to enhance answer quality through smarter inference strategies.

In the lecture, we discussed three main approaches to inference-time scaling:

---

1. Self-Consistency

Description:
The model generates multiple diverse reasoning chains via Chain-of-Thought (CoT) prompting — using temperature or sampling — and selects the most frequent final answer across them. This method builds directly on CoT by aggregating over multiple valid reasoning paths.

Advantages:

- Leverages the intuition that correct answers often reappear across different valid reasoning chains.
- Reduces the effect of individual hallucinations or flawed generations.
- Empirically shown to improve performance on complex reasoning tasks.

Computational Bottlenecks:

- Requires multiple forward passes per input (one per reasoning chain).
- Requires memory to store and tally multiple candidate outputs.

Parallelizable?
Yes — each reasoning chain is independent, so generations can be computed in parallel.

---

2. Verifiers

Description:
After generating multiple candidate outputs, each is passed through one or more verification modules — either rule-based (e.g., regex, unit tests) or learned — to assess correctness. The final output is the one that passes the most verifiers.

Advantages:

- Suitable for evaluating unstructured outputs such as code, math solutions, or free text.
- Improves interpretability by allowing inspection of which outputs pass which checks.
- More targeted than majority voting, especially when correctness is objective.

Computational Bottlenecks:

- Running multiple verification modules can be expensive depending on their complexity.
- Requires management of verification logic and scoring for each candidate.

Parallelizable?
Yes — both generation and verification can be parallelized across candidates, assuming independent modules.

---

3. Improved Chain-of-Thought Reasoning (Planning, Backtracking, Self-Evaluation)

Description:
The model reasons in a flexible and reflective manner — it can plan ahead,

backtrack to correct earlier mistakes, and try alternative reasoning paths when needed. This mimics more robust, human-like problem solving.

Advantages:

- Reduces wasted computation by salvaging partially correct reasoning chains.
- Enables deeper, more adaptive reasoning.
- Effective for multi-step tasks where intermediate steps may be flawed but recoverable.

Computational Bottlenecks:

- Requires longer and more dynamic sequences involving retries, conditional logic, and re-evaluations.
- Higher inference cost due to internal loops and dependency tracking.

Parallelizable?
Not easily — reasoning is **sequential and state-dependent**, which limits parallelization. Some partial steps or branches might be parallelized in advanced setups.

b) When solving a complex scientific reasoning task on a single GPU with large memory, it's crucial to choose a method that makes the most of serial compute without depending on parallel execution. The best choice in this setting is **Improved Chain-of-Thought Reasoning**. Unlike Self-Consistency or Verifier-based approaches, which rely on generating and evaluating multiple reasoning paths in parallel, Improved CoT concentrates computation on a **single reasoning trajectory** — allowing the model to plan ahead, detect and fix mistakes, and adaptively explore alternatives.

This approach avoids redundant forward passes, better leverages the available memory for long and detailed sequences, and delivers strong performance with minimal overhead. Its introspective, iterative structure makes it especially suitable for deep, multi-step scientific problem-solving in resource-constrained environments.

GitHub Repository URL for Section 2:

https://github.com/DorRegevStudent/ANLP-ex1

Programming Exercise:

2) a) Yes. The configuration with the best validation accuracy also achieves the best test accuracy. The results I got was:
config1 accuracy: validation: 0.8799, test: 0.8330

Config2 accuracy: validation: 0.8480, test: 0. 7948

config1 accuracy: validation: 0.8353, test: 0.8133

b) After comparing the validation examples where the best-performing configuration succeeded and the worst-performing configuration failed, several patterns emerge:

1. **Paraphrase Sensitivity (Semantic Similarity)**
   Many examples where the worst model failed involve paraphrases that use **different surface wording but preserve meaning**, e.g.:
   - Example: *"called as a witness at a pretrial hearing"* vs. *"will be called as a witness Wednesday in a pretrial hearing"*
   - Example: *"solid chance to become the first Triple Crown winner since Affirmed in 1978"* vs. *"looking to become horse racing's first Triple Crown winner in a generation"*

→ The best model successfully recognizes these as **non-paraphrases or paraphrases depending on subtle context**, while the worst model misclassifies them, likely because it struggles with **lexical variation and paraphrastic inference**.

2. **Numerical and Factual Details**
   Several examples include **numbers, percentages, or factual details** that change the meaning:
   - Example: *"1% to 3%"* vs. *"2.5%"* for trans fat, and *"14%"* vs. *"11–12%"* for saturated fat.
   - Example: *"bomb that destroyed the Murrah building"* vs. *"4,000-pound fuel-and-fertilizer bomb that destroyed the Murrah building"*.

→ The worst model struggles with **understanding the factual weight of numeric changes or added specifics**, while the best model correctly picks up on these details.

3. **Entity and Coreference Resolution**
   A few examples require resolving **who or what is being referred to**:
   - Example: Terri Schiavo's case is described from slightly different angles but refers to the same context.
   - Example: A subtle difference in phrasing around identity and rules.

→ The best model seems better at **tracking entities across sentences**, while the worst model fails when pronouns, names, or events are phrased differently.