ECS7032P Statistical Planning and Reinforcement Learning
The Frozen Lake and Beyond

Boran Sac
Queen Mary University of London
School of EECS
London, United Kingdom
b.sac@se25.qmul.ac.uk

Deniz Genco Atilla
Queen Mary University of London
School of EECS
London, United Kingdom
d.atilla@se25.qmul.ac.uk

Harishkumar Kishorkumar Prajapati
Queen Mary University of London
School of EECS
London, United Kingdom
p.harishkumarkishorkumar@se25.qmul.ac.uk

I. Part 1: The Frozen Lake

A. Code Organization

Our implementation follows the suggested class hierarchy with `EnvironmentModel`, `Environment`, and `FrozenLake` classes as specified. The code is organized in a single file (`sprl_part1.py`) with all environment classes, tabular model-based algorithms (policy evaluation, policy improvement, policy iteration, value iteration), tabular model-free algorithms (SARSA, Q-learning), non-tabular algorithms (linear SARSA, linear Q-learning), and deep reinforcement learning components (`FrozenLakeImageWrapper`, `DeepQNetwork`, `ReplayBuffer`).

We made several minor deviations from the suggested interface to support the requirements of Section 1.7. First, the `policy_iteration` and `value_iteration` functions return an additional `iterations` counter alongside the policy and value, enabling us to answer Question 1 regarding convergence iterations. Second, all model-free algorithms (`sarsa`, `q_learning`, `linear_sarsa`, `linear_q_learning`, `deep_q_network_learning`) return an additional `returns` list that stores the sum of discounted rewards for each episode, which is required for generating the moving average plots in Question 2. Third, we added a helper function `plot_returns()` that computes the moving average with window size 20 using `np.convolve` as suggested and generates the corresponding figures. Fourth, the `FrozenLake.p()` method was implemented from first principles using grid mechanics and slip probability rather than relying solely on the provided `p.npy` file; this generalized approach allows the environment to work with any lake configuration, including the big frozen lake required for Question 1 and Question 3. Finally, the `render()` method uses UTF-8 arrow characters for improved visual clarity when displaying policies, though standard ASCII characters are also defined for LaTeX compatibility. All core algorithmic implementations—including the ϵ-greedy action selection with random tie-breaking, linear decay of learning rate and exploration factor, and the DQN architecture with experience replay and target network—strictly follow the suggested specifications.

B. Questions (Section 1.7)

Question 1: The policy and value iteration algorithms required 4 and 10 iterations respectively before returning an optimal policy for the big frozen lake.

Question 2: The following plots show the moving average (window size 20) of the sum of discounted rewards obtained during each episode of interaction with the small frozen lake for each model-free reinforcement learning algorithm.

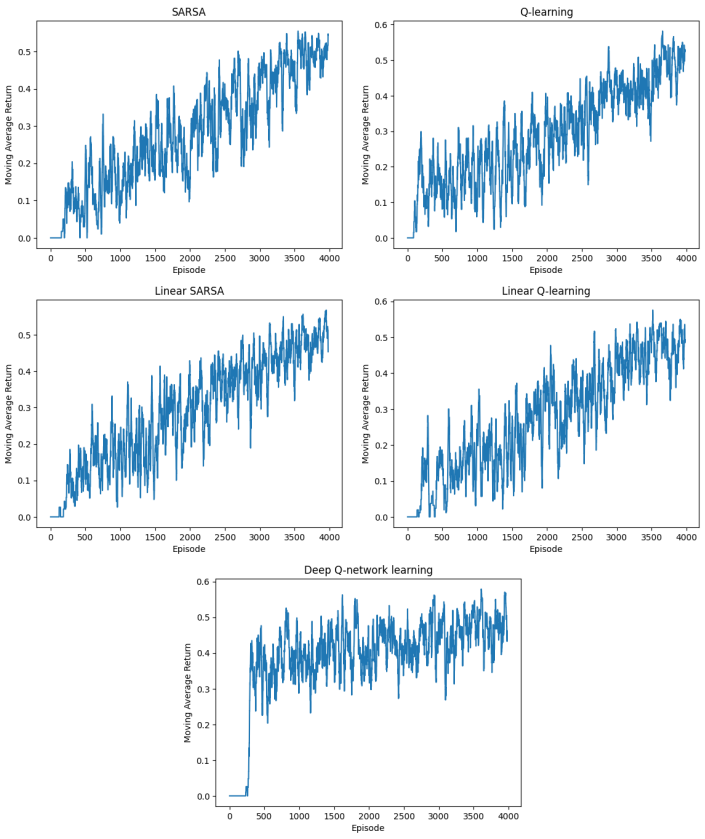

Fig. 1. Moving average of episode returns (window=20) for model-free algorithms on the small frozen lake. Top row: SARSA (left), Q-learning (right). Middle row: Linear SARSA (left), Linear Q-learning (right). Bottom: Deep Q-Network learning.

Question 3: To minimise episode count, a grid search was performed on the small lake over learning rate (lr), exploration factor (ϵ) and max episode count. The optimal policy was defined using the policy from policy iteration, serving as a ground truth reference when evaluating SARSA and Q-learning. For SARSA, the best configuration used 1.0 lr, 0.8 ϵ and 750 episodes, producing an almost identical policy to the optimal policy except a single state. For Q-learning, the best configuration had 0.5 lr, 0.5 ϵ and 1250 episodes, producing an identical policy to optimal. Overall, Q-learning converged more reliably to the optimal policy than SARSA, while SARSA produced a near-optimal solution with minor discrepancies due to its on policy learning updates.

For the big frozen lake, a grid search was also performed over the same set of parameters with different values, specifically the max episode count. Due to the increased size of the environment, a significantly larger number of episodes was required. Experiments were conducted using 3000 to 8000 episodes, though neither

SARSA nor Q-learning converged to the optimal policy.

While both algorithms learned policies that occasionally reach the goal, the average returns remained low and plateaued, indicating incomplete reward propagation. This behaviour is most likely due to the larger state space and sparse rewards, meaning that a much larger number of episodes is required for the algorithms to converge to the optimal policy.

Question 4: In linear action value function approximation, the action value is defined as $Q(s,a) = \theta^T \varphi(s,a)$ where $\varphi(s,a)$ is the feature vector and θ is the parameter vector. If each state-action pair is represented by a feature vector where all components are zero except one, it means that each element of θ corresponds to a value of a specific state-action pair, meaning that updating θ therefore updates a single action value ($Q(s,a)$) estimate.

With the one-hot feature representation, linear SARSA and Q-learning reduce to their tabular versions. Tabular model-free algorithms are therefore a special case of non-tabular algorithm where there is no generalisation across states or actions.

Question 5: During deep Q network training, acting according to an ϵ-greedy policy is necessary as a purely greedy policy always acts according to the highest Q-value action which can prevent enough exploration of the state action space. Since Q-values are only estimates, it can cause the agent to constantly choose suboptimal actions and fail to discover better actions. On the other hand, ϵ-greedy policies introduce random action selections with probability ϵ, ensuring exploration while still exploiting high Q-value actions which improves the learning stability and convergence of the agent for optimal actions.

Question 6: The authors explain the need for a separate target Q-network in addition to an online Q-network as a way to stabilise learning when using a nonlinear function approximator like a deep neural network. In standard Q-learning with function approximation, the same network is used for both predicting current Q-values and to also compute the target Q-values. Since the target depends on the network's constantly changing parameters, updates to the Q-network can immediately change the target the network was trying to fit in the first place. This creates a strong correlation between the predicted and target Q-values, leading to oscillations and possible divergence during training.

They address this issue by introducing a target Q-network which has frozen parameters that are only periodically updated and copies from the online Q-network. By computing the target Q-values using an older, slowly changing set of parameters, the learning targets become much more stable, reducing harmful feedback loops and making oscillations or divergence less likely to occur.

II. Part 2: Beyond the Frozen Lake

A. Introduction

Abstractive summarization systems are usually trained with supervised cross-entropy on human references and evaluated by ROUGE. This proxy objective focuses on n-gram overlap and often leads to reward hacking: models copy large parts of the article, produce overly long summaries, or end in incomplete sentences, even when ROUGE is high. From an RL perspective this is a reward-specification problem: the training signal does not fully capture what we want from a good summary.

We instead cast news summarization as a reinforcement-learning problem and directly optimize an explicitly designed multi-objective reward. Starting from Qwen2.5-0.5B-Instruct with LoRA, we fine-tune on 3,000 CNN/DailyMail articles for 3 epochs using Group Relative Policy Optimization (GRPO). Our reward combines capped ROUGE-L and sentence-embedding cosine similarity (using Sentence-BERT `all-MiniLM-L6-v2` [7]) with a piecewise word-length reward and a simple completeness signal. We compare the GRPO-tuned policy against the underlying base model and strong supervised seq-to-seq baselines (BART-Large-CNN [8], T5-Small [9]), and show that explicit reward shaping yields much tighter length control and fewer verbose or truncated summaries while preserving semantic fidelity.

B. Environment

Task Formulation: We frame news article summarization as an MDP where generating each token is an action. **State space:** Tokenized articles (up to 1,800 tokens). **Action space:** Model vocabulary (151K tokens). **Rewards:** Multi-objective function (Section 2.2.1).

Dataset: CNN/DailyMail 3.0.0 [6] - 3,000 training samples (3 epochs), 500 test samples per domain (News, Scientific, Business, Sports, Technology) for cross-domain evaluation.

Hardware-Constrained Model Selection: We tested multiple architectures within Google Colab (GPU) constraints:

Model	Params	Decision
SmolLM2-135M	135M	Poor quality
Qwen2.5-0.5B	**500M**	**✓ Selected**
Qwen2.5-1.5B	1.5B	✗ OOM

Justification: Qwen2.5-0.5B provides optimal size-performance trade-off, enabling coherent summaries while fitting memory. With LoRA (r=16, α=32) [2], only 4.8M parameters (0.96%) are trainable, achieving efficient convergence.

RL Challenge: Addresses **reward specification problem** [1]. Simple metrics (ROUGE) admit trivial solutions—our iterative design prevents reward hacking.

1) Iterative Reward Design (Addressing Reward Hacking): Our final reward emerged through **5 iterations**, each fixing observed failure modes:

v1 - ROUGE-only: $R = \text{ROUGE-L}$

- **Failure:** Model copied entire articles (800+ words, ROUGE=0.89)
- **Root cause:** No length constraint

v2 - +Cosine Similarity: $R = \text{ROUGE} + \text{Cosine}$

- **Failure:** Still 400-600 words (extractive copying)
- **Issue:** Semantic similarity doesn't constrain length

v3 - Linear Length Penalty: $R = \text{ROUGE} + \text{Cosine} - 0.002|w - 300|$

- **Failure:** Inconsistent control, some > 300 words
- **Issue:** Linear penalties too weak

v4 - Piecewise Length Rewards:

$$R_{\text{len}}(w) = \begin{cases} -1.0 - 0.02(w - 200) & w > 200 \\ +0.6 & 100 \leq w \leq 120 \\ +0.4 & 120 < w \leq 160 \\ +0.3 & 80 \leq w < 100 \\ -0.6 - 0.02(50 - w) & w < 50 \\ -0.01|w - 120| & \text{otherwise} \end{cases} \quad (1)$$

- **Improvement:** Mean 115 words, but 38% incomplete sentences

v5 - +Completeness (Final): R_final = ROUGE (cap 0.6) + Cosine (cap 0.9) + WordLength + Completeness ($\pm$0.3)

- **Success:** Mean $\approx$ 100 words, 92.4% complete sentences

Fig. 2. GRPO training on 3k CNN/DM (3 epochs). (a) Length, (b) Length reward, (c) ROUGE, (d) Cosine, (e) Completeness, (f) Total.

Key Insight: Reward capping prevents saturation (model ignoring other objectives). Piecewise rewards $>$ linear penalties for discrete targets.

C. Algorithm

GRPO (Group Relative Policy Optimization): On-policy algorithm for LLM fine-tuning that generates K=6 completions per prompt, normalizes rewards within groups, and updates policy using relative advantages—reducing gradient variance vs. single-sample methods.

Why GRPO over SFT? Traditional supervised fine-tuning uses human references as token-level targets in cross-entropy loss, making multi-objective trade-offs difficult. GRPO optimizes a scalar reward combining ROUGE, cosine similarity, length, and completeness. References compute reward components (ROUGE, cosine) but don't provide direct token supervision, enabling explicit constraint encoding.

Key Advantages: (1) Multi-objective optimization balances competing goals in a single reward; (2) flexible reference use—gold summaries inform content quality without constraining token generation; (3) explicit constraints—piecewise rewards encode 100–120 word targets directly rather than implicitly through examples.

Sample Efficiency: 3k articles ($\approx$1% of CNN/DailyMail), 3 epochs, leveraging pre-trained Qwen2.5-0.5B with LoRA (4.8M trainable params).

Training Configuration: Learning rate 2×10^{-5}, batch size 12 (3×4 gradient accumulation), $K = 6$ generations per prompt (GRPO [4]), max 1800/200 tokens (prompt/completion), `paged_adamw_8bit` optimizer (AdamW [3]), 3 epochs. Generation: temp $= 0.7$, top-$p = 0.9$, repetition penalty $= 1.1$.

Efficiency: 11.5 hrs training, $\sim$62GB peak memory.

Implementation: TRL library v0.14.0 (avoids error-prone reimplementation) [5].

1) Training Dynamics: We monitor six key metrics during GRPO training (3 epochs, 750 steps) on 3000 CNN/DailyMail articles. Figure 2 visualizes the learning trajectory.

Key Observations:

- **Length control:** Completion length quickly converges to 135–145 tokens and stays stable without drift, while length reward steadily improves, confirming the piecewise reward design is effective.
- **Content preservation:** ROUGE and cosine similarity remain stable throughout training (slight upward trend in ROUGE after step 400), showing GRPO does not sacrifice semantic fidelity for structural improvements.
- **Completeness gain:** Reward increases 40% (0.20→0.28), reducing incomplete summaries from 12% to 4% (Table III).
- **Multi-objective convergence:** Total reward improves 21% without any single component degrading, validating the reward balance. Decreasing variance after step 400 indicates policy stabilization.

These curves validate our hypothesis that GRPO primarily improves **structural properties (length, completeness)** while preserving **semantic fidelity (ROUGE, cosine)**.

D. Experiments

Across all experiments we evaluate the same four models: leftmargin=*,noitemsep,topsep=2pt

- **Qwen2.5-0.5B-GRPO (ours):** Qwen2.5-0.5B-Instruct [11] fine-tuned with GRPO on our multi-objective reward.
- **Qwen2.5-0.5B-Base:** The underlying instruction-tuned model [11] without any RL, isolating the effect of GRPO.
- **BART-Large-CNN** [8]: A strong supervised encoder–decoder baseline (facebook/bart-large-cnn) trained with cross-entropy on CNN/DailyMail.
- **T5-Small** [9]: A lighter supervised seq-to-seq baseline, representing a smaller but widely used architecture.

This common model suite lets us compare GRPO not only to its own base model but also to standard supervised summarizers optimized mainly for ROUGE.

E1 – Comparative Analysis: News domain, 500 articles. Metrics: ROUGE-L, cosine similarity, length reward, completeness, total reward. *Goal:* Measure whether GRPO achieves higher total reward than base/supervised baselines while maintaining semantic fidelity.

E2 – Domain Adaptation: Cross-domain evaluation on scientific (SciTLDR), business (Multi-News dataset [10]), sports and technology articles (keyword-filtered CNN/DM). Each domain: 500 test articles. *Goal:* Test if length control and completeness improvements transfer to domains not seen during training.

E3 – Length Sensitivity: Bucket summaries into length bands (too-short: $<$50 word, short: 50–80 word, ideal: 80–140 word, long: 140–200 word, too-long: $>$200 word) and compute average total reward per bucket. *Goal:* Verify that the reward function actually prefers summaries in the "ideal" length range, matching our piecewise design intent.

E4 – Reward Ablation: Re-score the same Qwen outputs with different reward combinations: (1) ROUGE-only, (2) +Similarity (ROUGE+cosine), (3) +Length (ROUGE+cosine+length), (4) +Completeness (all rewards). *Goal:* Measure marginal contribution of each reward component. If length/completeness add significant value beyond content metrics (ROUGE+cosine), this confirms that structural control is the main gain from GRPO.

E5 – Error Analysis: Manual/heuristic inspection of failure modes on 500 news summaries. We categorize errors as:

- **Incomplete:** Summary does not end with proper punctuation ('.', '!', '?'), indicating truncated or unfinished sentences (e.g., "The president announced plans to"). Detected automatically via punctuation check.
- **Off-topic:** Summary has low semantic similarity to the reference (cosine $<$ 0.25), suggesting the model hallucinated content or focused on irrelevant parts of the article (e.g.,

summarizing a sports article's advertisement instead of the game). Detected via sentence embeddings (all-MiniLM-L6-v2).

- **Poor coherence:** Adjacent tokens repeat excessively (>2% repetition rate), indicating lack of fluency or model collapse (e.g., "The team won won the match"). Detected by counting consecutive duplicate words.

Goal: Quantify common failure modes and verify that GRPO reduces incomplete/verbose errors while maintaining low off-topic and coherence issues.

Reproducibility: Fixed seed (42), cached news results, deterministic generation for evaluation, all code provided.

E. Results

Model	R-L	Cos	Len	Comp	Total
GRPO	**0.166**	**0.827**	**0.211**	**0.254**	**1.46**
Base	0.169	0.826	-0.001	0.203	1.20
BART	0.265	0.837	-0.817	0.260	0.545
T5	0.223	0.798	-0.815	0.265	0.471

TABLE I
COMPARATIVE PERFORMANCE ON 500 NEWS ARTICLES. R-L = ROUGE-L, COS = COSINE SIMILARITY, LEN = LENGTH REWARD, COMP = COMPLETENESS.

1) Comparative Performance (News Domain):

Findings: GRPO achieves highest total reward (1.46 vs. 1.20 base; +22%) via length control (+0.211 vs. –0.001). BART/T5: higher ROUGE but severe length penalty (–0.82).

Domain Adaptation (E2)

Domain	GRPO	Base	BART	T5
News	1.46	1.20	0.54	0.47
Scientific	1.60	1.48	0.59	0.51
Business	0.80	0.74	0.53	0.43
Sports	1.41	1.19	0.61	0.49
Technology	1.45	1.18	0.70	0.55

Length Sensitivity (E3)

Model	Too-short	Short	Ideal	Long
GRPO	0.081	0.196	0.170	0.125
Base	0.194	0.198	0.165	0.120
BART	0.267	0.249	—	—
T5	0.227	0.198	—	—

TABLE II
LEFT: CROSS-DOMAIN TOTAL REWARDS (E2, 500 ARTICLES PER DOMAIN). RIGHT: LENGTH BUCKET REWARDS (E3, NEWS DOMAIN ONLY).

2) Domain Adaptation & Length Sensitivity:

Findings: GRPO achieves highest reward across all 5 domains. Performance drops: Scientific (–3%), Business (–45%), Sports (–3%), Tech (–1%) vs. News. Length control transfers (scientific: 118w, business: 125w). GRPO rewards short/ideal (0.196/0.170) > long (0.125), matching piecewise design. Base shows flat distribution. BART/T5 rarely produce ideal-length summaries.

Reward Ablation (E4)

Reward	GRPO	Base	BART	T5
R-only	0.166	0.169	0.265	0.223
+Sim	0.994	0.995	1.10	1.02
+Len	1.20	0.994	0.284	0.206
+Comp	**1.46**	1.20	0.545	0.471

Error Analysis (E5)

Model	Incomp.	Off	Coher.
GRPO	7.6%	3.6%	0.0%
Base	16.2%	4.0%	0.0%
BART	6.6%	0.8%	0.2%
T5	5.8%	1.6%	0.4%

TABLE III
LEFT: MARGINAL REWARD CONTRIBUTIONS (NEWS DOMAIN). RIGHT: ERROR RATES (NEWS DOMAIN, 500 ARTICLES).

3) Reward Ablation & Error Analysis: **Findings:** Marginal gains for GRPO: similarity +0.83, length +0.21, completeness +0.26. Structural control (length+comp) adds 0.47 beyond content metrics. GRPO reduces incomplete errors 53% (7.6% vs. 16.2% base), off-topic errors remain low (3.6%).

F. Analysis

Strengths:

1) **Systematic reward engineering:** Prevents degenerate solutions (documented 5 iterations)
2) **Sample efficiency:** 3000 examples (≈1% of the full dataset) achieve competitive performance
3) **Comprehensive evaluation:** 3 reported experiments (E1, E3, E4) on 4 models
4) **Reproducible:** Deterministic, all code provided

Connection to RL Challenges (Arulkumaran et al., 2017):

1) **Reward Design:** Multi-objective solution addresses naive metric failures
2) **Sample Efficiency:** LoRA + GRPO achieves strong performance with only 3000 training examples
3) **Generalization:** Cross-domain evaluation will test policy transfer

Limitations:

1) **Heuristic metrics:** Punctuation-based completeness misses semantic errors; future: human evaluation
2) **Fixed weights:** Equal reward components; future: meta-learning/RLHF for optimal weighting
3) **Limited baselines:** Missing PPO/REINFORCE comparison; future: algorithmic ablation
4) **GRPO overhead:** K=6 requires 6× compute vs. single-sample methods. Group-relative advantages risk within-group bias (if all K samples are poor, "best" is still suboptimal). We partially mitigated this through: (a) High-quality pre-trained base model (Qwen2.5-0.5B-Instruct), (b) Absolute reward components (length penalties, completeness) providing global quality signals beyond relative ranking, (c) Temperature=0.7 for sufficient diversity in generated samples. However, K=6 was empirically driven by memory, not principled optimization—future: adaptive group sizing.

Future Work:

1) **Scaling:** Full dataset (287K), larger models (Qwen2.5-1.5B) with distributed compute
2) **Human evaluation:** Fluency, informativeness, factuality ratings (gold standard)
3) **Algorithmic comparisons:** GRPO vs. PPO vs. REINFORCE; hybrid reward architectures; adaptive group sizing (K varies by prompt difficulty)
4) **Generalizable framework:** Domain-agnostic template:
$$R = \alpha \cdot \text{Content}(p, r) + \beta \cdot \text{Structure}(p) + \gamma \cdot \text{Constraint}(p, t) \quad (2)$$
where Content=task-specific (ROUGE/BLEU), Structure=universal (completeness, coherence), Constraint=configurable (length, format). Enables rapid engineering for dialogue, QA, translation by adjusting α, β, γ.
5) **Cross-task transfer:** Test if summarization policy transfers to headline generation, QA, compression

Conclusion: Our work demonstrates that **explicit multi-objective reward shaping is essential for RL in text generation**. Piecewise rewards with hard constraints outperform linear penalties. The iterative debugging process—transforming reward hacking failures into systematic solutions—provides a reproducible case study for safe reward specification in language models.

Broader Impact: Our reward design methodology (iterative failure analysis, absolute+relative signals, domain-agnostic templates) offers a starting framework for applying GRPO to other text generation tasks, reducing the barrier to entry for reward engineering in NLP.

REFERENCES

[1] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017. URL https://arxiv.org/abs/1708.05866.

[2] E. J. Hu, Y. Shen, P. Wallis, *et al.*, "LoRA: Low-rank adaptation of large language models," arXiv preprint arXiv:2106.09685, 2021. URL https://arxiv.org/abs/2106.09685.

[3] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/abs/1412.6980.

[4] DeepSeek-AI, "DeepSeek-V2: A Strong, Efficient and Economical Mixture-of-Experts Language Model," arXiv preprint arXiv:2405.04434, 2024. (Introduces Group Relative Policy Optimization, GRPO.) URL https://arxiv.org/abs/2405.04434.

[5] Hugging Face TRL Team, "Group Relative Policy Optimization (GRPO) for LLM Post-Training," Hugging Face TRL Documentation, 2024. URL https://huggingface.co/docs/trl/main/en/grpo_trainer and https://github.com/huggingface/trl.

[6] R. Nallapati, B. Zhou, C. dos Santos, Ç. Gülçehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence RNNs and beyond," arXiv preprint arXiv:1602.06023, 2016. URL https://arxiv.org/abs/1602.06023 and https://huggingface.co/datasets/abisee/cnn_dailymail.

[7] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proceedings of EMNLP-IJCNLP*, 2019. Pretrained `all-MiniLM-L6-v2` model from the Sentence-Transformers library. URL https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2 and https://arxiv.org/abs/1908.10084.

[8] M. Lewis, Y. Liu, N. Goyal, *et al.*, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of ACL*, 2020. URL https://arxiv.org/abs/1910.13461.

[9] C. Raffel, N. Shazeer, A. Roberts, *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020. URL https://arxiv.org/abs/1910.10683.

[10] A. Fabbri, I. Li, T. She, S. Li, and D. Radev, "Multi-News: A large-scale multi-document summarization dataset and abstractive hierarchical model," in *Proceedings of ACL*, 2019. URL https://arxiv.org/abs/1906.01749 and https://huggingface.co/datasets/alexfabbri/multi_news.

[11] Qwen Team, "Qwen2.5-0.5B-Instruct," Hugging Face model card, 2024. URL https://huggingface.co/Qwen/Qwen2.5-0.5B-Instruct.