Quantitative Question Answering (QQA)

Abstract

Quantitative Question Answering (QQA) requires numerical reasoning to select correct answers from textual questions with numerical data. This project implements three baseline approaches: a rule-based system, a fine-tuned RoBERTa model, and a zero-shot model using facebook/bart-large-mnli. We enhance the pipeline with preprocessing for scientific notation and feature engineering for contextual cues. Evaluation on the QQA dataset shows fine-tuned RoBERTa outperforms others, emphasizing the role of domain-specific training. Challenges in multi-step reasoning and data variability guide future improvements.

1 Introduction

Quantitative Question Answering (QQA) entails extracting and reasoning with numerical values embedded in text to answer questions accurately. For example, solving "A train travels at 80 km/h for 3 hours; what is the distance?" yields 240 km. This task is pivotal in applications like automated tutoring, financial analysis, and statistical reporting, where numerical insights are critical. Unlike standard question answering, QQA demands precise numerical extraction, contextual understanding, and arithmetic or logical operations, making it a complex NLP challenge.

Our project develops baseline models for QQA, comparing rule-based, fine-tuned, and zero-shot approaches. We incorporate preprocessing to handle scientific notation and feature engineering to capture relevant keywords, addressing gaps in numerical reasoning pipelines. The motivation is to advance intelligent systems that process numerical queries efficiently, reducing manual effort in data-driven domains. For instance, a system answering "Which planet has a diameter of 1.39E6 km?" requires recognizing scientific notation and mapping it to options like "Jupiter" or "Mars." Our goal is to establish robust baselines and identify areas for enhancing QQA performance.

2 Related Work

Numerical reasoning in NLP has seen significant progress. **?** introduced the DROP dataset, focusing on numerical operations like sorting and addition in reading comprehension. **?** proposed NumNet, a graph-based model enhancing numerical reasoning on datasets like DROP. Transformer models, such as RoBERTa (**?**), excel in contextual tasks and are adaptable for QQA with fine-tuning. Zero-shot learning, as in **?**, offers flexibility but often lacks precision in specialized domains. Preprocessing techniques, like those by **?**, emphasize numerical feature extraction, while **?** explored injecting numerical skills into language models.

Our work builds on these by evaluating three baselines on the QQA dataset, integrating preprocessing for scientific notation and keyword-based feature engineering. Unlike studies focusing on single models, we provide a comparative analysis, highlighting trade-offs between simplicity, accuracy, and adaptability. We also address scientific notation, a less-explored aspect in prior QQA research, ensuring compatibility with questions involving large or small numbers.

3 Methodology

Our pipeline includes preprocessing, feature engineering, and three baseline models, detailed below.

3.1 Preprocessing

We preprocess the QQA dataset to extract scientific notation numbers (e.g., 1.2E3), as shown in Listing **??**. The `preprocess.py` script loads JSON data, applies regular expressions, and adds a `numeric_values` field. This step ensures models handle large or small numbers accurately, critical

for questions like those involving planetary distances or molecular weights.

3.2 Feature Engineering

The `feature_engineering.py` script extracts numerical values and keywords (e.g., "speed," "friction") to create a `features` dictionary. Keywords are selected based on domain relevance, such as physics terms for motion or energy. This enhances rule-based reasoning and provides contextual cues for neural models, as shown in Listing **??**.

3.3 Baseline Models

- **Rule-based Baseline**: Extracts numbers using regex and applies rules based on keywords. For example, if "speed" and "time" appear, it calculates distance = speed $\times$ time. The `model.py` script includes rules for physics-related terms and defaults to numerical comparisons, as shown in Listing **??**.
- **Pre-trained Model Baseline**: Fine-tunes a RoBERTa model on the QQA dataset using Hugging Face Transformers. The `main.py` script converts data into multiple-choice format, training for 3 epochs with a learning rate of 2e-5.
- **Zero-shot Learning Baseline**: Uses facebook/bart-large-mnli for classification without fine-tuning. The `model.py` script employs the zero-shot pipeline to select the option with the highest confidence score.

The `main.py` script orchestrates preprocessing, feature engineering, and model execution, while `utils.py` logs progress for transparency.

4 Dataset, Experimental Setup, and Results

4.1 Dataset

The QQA dataset comprises train, dev, and test splits, each with questions requiring numerical reasoning (e.g., calculations, comparisons). Questions include two options, such as "120 km" or "160 km" for a distance query. The dataset varies in complexity, from simple arithmetic to multi-step reasoning involving percentages or scientific notation.

4.2 Experimental Setup

- **Preprocessing**: Extracts scientific notation for all splits using `preprocess.py`.
- **Feature Engineering**: Adds numerical counts and keywords via `feature_engineering.py`.
- **Rule-based**: Applies regex and rules from `model.py`, handling physics and comparison queries.
- **RoBERTa**: Fine-tuned with batch size 8, 3 epochs, and learning rate 2e-5 using `main.py`.
- **Zero-shot**: Uses bart-large-mnli with default pipeline settings in `model.py`.

Accuracy is the primary metric, calculated as the proportion of correct option selections.

RoBERTa achieved the highest test accuracy (0.53), benefiting from fine-tuning. The rule-based baseline was consistent (0.49–0.52) but limited by contextual gaps. The zero-shot model (0.48–0.51) struggled with domain-specific reasoning. RoBERTa's low dev accuracy (0.35) suggests potential overfitting or dataset mismatch, requiring further analysis.

5 Discussion

The results highlight distinct strengths and weaknesses. The rule-based baseline's balanced accuracy (0.49–0.52) suits simple calculations but fails on questions needing contextual interpretation, such as those involving abstract terms like "energy" or "force." For example, a question like "Which material is stronger given thicknesses of 2.5E-3 m and 3.0E-3 m?" exposed rule-based limitations, as it relied solely on numerical comparison without material context.

The zero-shot model's moderate performance (0.48–0.51) reflects its general knowledge but lack of QQA-specific training, particularly for multi-step reasoning. RoBERTa's test accuracy (0.53) underscores fine-tuning's impact, yet its dev accuracy (0.35) raises concerns about generalization. Preprocessing improved handling of scientific notation, but regex errors occasionally missed numbers (e.g., malformed notation). Feature engineering aided rule-based logic by prioritizing keywords, but RoBERTa likely captured these implicitly, reducing feature impact.

Multi-step reasoning was a universal challenge. Questions requiring sequential operations (e.g., calculate speed, then adjust for friction) often led to errors across models. Preprocessing and feature engineering mitigated some issues, but complex ques-

tions exposed gaps in rule-based simplicity and zero-shot adaptability. RoBERTa's performance suggests transformers can bridge these gaps with more training data or advanced architectures.

6 Conclusion and Future Work

This project establishes baselines for QQA, with fine-tuned RoBERTa outperforming rule-based and zero-shot models due to its contextual reasoning. Preprocessing for scientific notation and feature engineering for keywords enhanced performance, particularly for rule-based systems. However, multi-step reasoning and dataset variability remain challenges.

Future work includes:

- **Hybrid Models**: Combine rule-based logic with neural networks for robustness.
- **Dataset Expansion**: Include more multi-step and diverse questions.
- **Advanced Models**: Test models like GPT-4 or T5 for complex reasoning.
- **Preprocessing Refinement**: Improve regex to handle edge cases in notation.
- **Error Analysis**: Conduct detailed analysis of dev-test discrepancies in RoBERTa.

These improvements aim to advance QQA systems, enabling reliable numerical reasoning in real-world applications like education and analytics.