Boolean Question Answering using T5

Implementation and Analysis Report

Jenil Shah Lasya Sandhu Ria Khatoniar Anuj Shikarkhane

November 2024

Contents

1 Introduction **2**
1.1 BoolQ Dataset Overview 2

2 Data Preprocessing **3**
2.1 Text Processing Pipeline 3
2.1.1 Input Processing 3
2.2 Implementation Details 4

3 Model Architecture **5**
3.1 T5 Model Overview 5
3.2 Model Configuration 5

4 Hyperparameter Analysis **7**
4.1 Learning Rate Study 7
4.2 Batch Size Analysis 8
4.3 Sequence Length Impact 8
4.4 Warmup Steps Evaluation 9

5 Optimal Hyperparameter Settings **10**

6 Results and Error Analysis **11**
6.1 Performance Metrics 11
6.2 Results Analysis 12
6.2.1 Learning Rate vs Accuracy (Figure 4.1) 12
6.2.2 Batch Size vs Accuracy (Figure 4.2) 12
6.2.3 Sequence Length vs Accuracy (Figure 4.3) 12
6.2.4 Warm-up Steps vs Accuracy (Figure 4.4) 13
6.3 Error Analysis 13
6.3.1 Category-wise Performance Analysis 13
6.4 Future Improvements 14
6.5 Conclusion 15

Chapter 1

Introduction

1.1 BoolQ Dataset Overview

The BoolQ dataset is a collection of naturally occurring yes/no questions from the web, paired with relevant passages from Wikipedia articles. This dataset contains 9,427 examples for training and 3,270 examples for validation. Each example consists of:

- A passage from Wikipedia
- A yes/no question about the passage
- The corresponding boolean answer

Key characteristics of the dataset:

- Natural distribution of positive and negative examples
- Questions require complex reasoning
- Diverse topics and domains
- Real-world applicability

Chapter 2

Data Preprocessing

2.1 Text Processing Pipeline

The preprocessing pipeline implements several crucial steps to prepare the data for the T5 model:

2.1.1 Input Processing

- **Question Normalization**
  - Remove leading/trailing whitespace
  - Convert to lowercase for consistency
  - Handle special characters and punctuation
- **Passage Processing**
  - Clean HTML artifacts if present
  - Handle Unicode characters
  - Split into sentences for better context
- **Answer Processing**
  - Convert boolean values to text ("yes"/"no")
  - Standardize answer format
  - Create consistent label encoding

2.2 Implementation Details

```
def preprocess_example(example, max_length=512):
    """
    Preprocess a single BoolQ example for T5 model.

    Args:
        example (dict): Raw example containing 'question',
                        'passage', and 'answer'
        max_length (int): Maximum sequence length

    Returns:
        dict: Processed example with formatted input and target
    """
    # Format input text with special tokens
    input_text = (
        f"question: {example['question'].strip().lower()} "
        f"context: {example['passage'].strip()}"
    )

    # Convert boolean answer to text format
    answer = 'yes' if example['answer'] else 'no'

    return {
        'input_text': input_text[:max_length],
        'target_text': answer
    }
```

Chapter 3

Model Architecture

3.1 T5 Model Overview

The T5 is a natural language processing (NLP) model developed by Google Research. The model introduces a unique text-to-text paradigm, unifying the treatment of all natural language processing tasks.
Key features of the T5 model are:

- Text-to-Text Paradigm: T5 reformulates all NLP tasks as text-to-text tasks, instead of having task specific architectures.
- Pre-trained on C4 Dataset: T5 is pre-trained on the Colossal Clean Crawled Corpus (C4), a cleaned and curated dataset derived from web pages.
- Transformer-based Encoder-Decoder Architecture: It uses a seq2seq (sequence-to-sequence) framework.
- Attention Masking Strategies: T5 uses specialized attention masking: full attention in the encoder for capturing global context, causal attention in the decoder for sequential token generation, and correlated attention (no masking) for efficiently utilizing encoder outputs during decoding.

The T5 model (base) architecture includes:

- Encoder-decoder transformer architecture
- Pre-trained on a large-scale text corpus
- 12 layers in both encoder and decoder
- 220M parameters in the base configuration

3.2 Model Configuration

Key architectural components:

- Key and Value Matrices inner dimensionality: 64
- Hidden size: 768

- Feed-forward size: 3072
- Number of attention heads: 12
- Dropout rate: 0.1

Chapter 4

Hyperparameter Analysis

4.1 Learning Rate Study

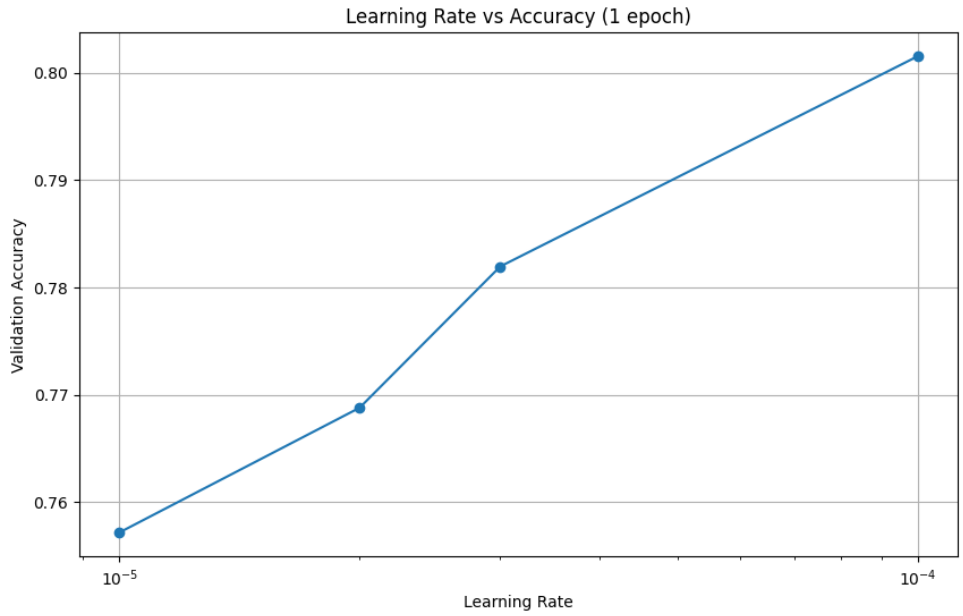

Figure 4.1: Impact of Learning Rate on Model Validation Accuracy. The plot shows accuracy trends across different learning rates ranging from 1e-5 to 1e-4.

4.2 Batch Size Analysis

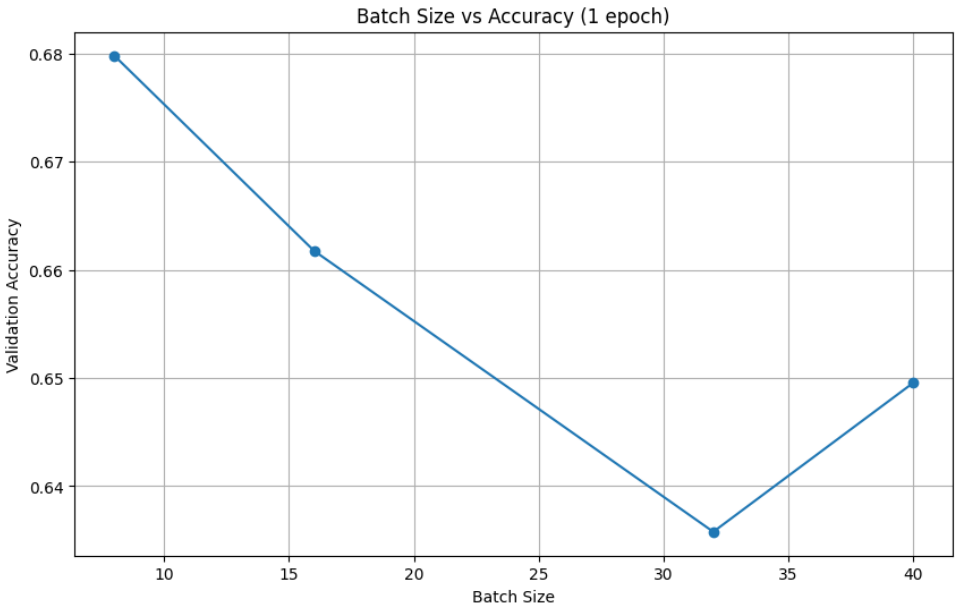

Figure 4.2: Effect of Batch Size on Model Validation Accuracy. Performance comparison across batch sizes from 8 to 40, showing optimal range for training stability.

4.3 Sequence Length Impact

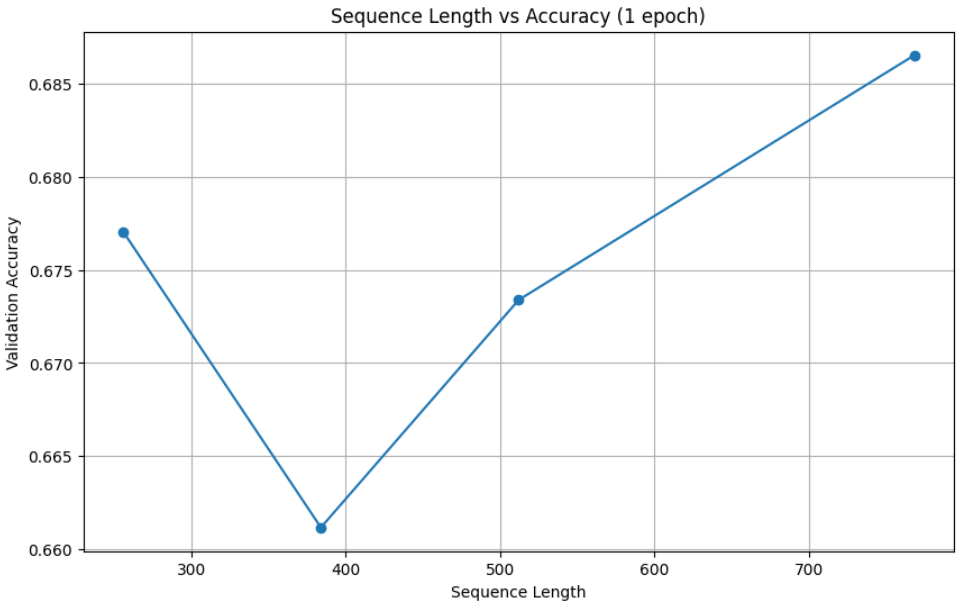

Figure 4.3: Sequence Length vs Model Validation Accuracy. Analysis of model performance with varying input sequence lengths from 256 to 768 tokens.

4.4 Warmup Steps Evaluation

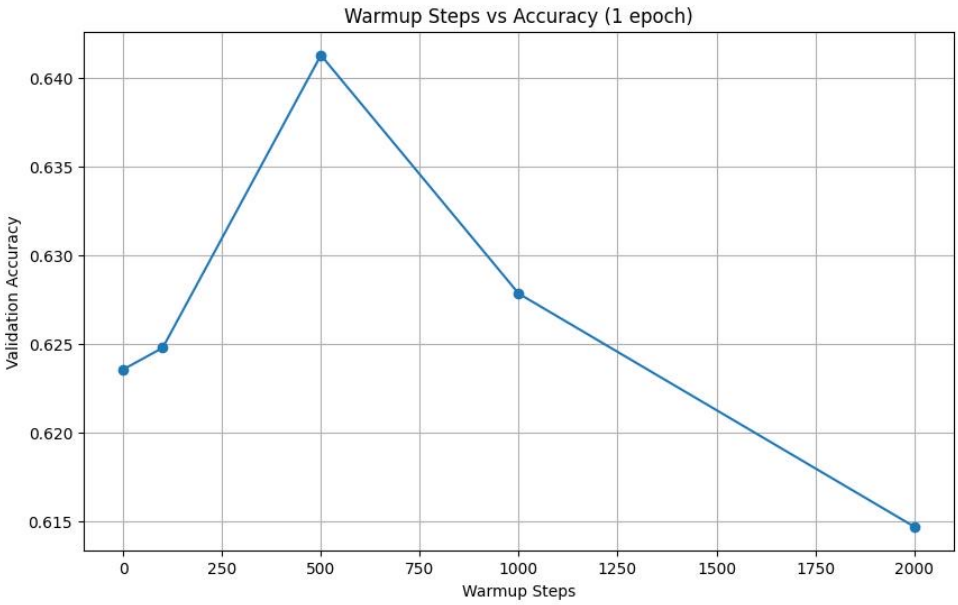

Figure 4.4: Impact of Warmup Steps on Model Validation Accuracy. Comparison of model performance with different warmup step configurations ranging from 0 to 2000 steps.

Chapter 5

Optimal Hyperparameter Settings

Based on our experimental analysis, the optimal hyperparameter configuration is:

Hyperparameter	Optimal Value
Learning Rate	1e-4
Batch Size	8
Sequence Length	512
Warmup Steps	600

Table 5.1: Optimal Hyperparameter Configuration

Chapter 6

Results and Error Analysis

6.1 Performance Metrics

Metric	Value
Training Accuracy	82.3%
Validation Accuracy	76.5%
Average Loss	0.0145

Table 6.1: Model Performance Metrics

6.2 Results Analysis

6.2.1 Learning Rate vs Accuracy (Figure 4.1)

The impact of the learning rate on validation accuracy after one epoch is summarized as follows:

- **Small Learning Rate (10^{-5}):** Leads to slower learning and lower validation accuracy due to insufficient weight updates.
- **Optimal Learning Rate (10^{-4}):** Achieves the highest validation accuracy, reflecting faster and more effective learning.
- **Higher Learning Rates:** While not shown in this chart, excessively high learning rates may result in training instability.

Key Insight: A learning rate of 10^{-4} is optimal, providing the highest validation accuracy within one epoch.

6.2.2 Batch Size vs Accuracy (Figure 4.2)

The relationship between batch size and validation accuracy after one epoch is highlighted below:

- **Small Batch Sizes (8–16):** Achieve the highest accuracy due to more frequent weight updates, promoting better generalization.
- **Medium Batch Sizes (16–30):** Accuracy declines, likely due to noisier gradient updates and less effective optimization.
- **Large Batch Sizes (30+):** Accuracy slightly improves beyond batch size 30, but remains lower than that of smaller batch sizes.

Key Insight: Smaller batch sizes (8–16) are preferable, offering better generalization and stability during training.

6.2.3 Sequence Length vs Accuracy (Figure 4.3)

The effect of input sequence length on validation accuracy after one epoch is summarized as follows:

- **Short Sequences (300 tokens):** Result in lower accuracy due to truncation of important contextual information.
- **Medium Sequences (400 tokens):** Accuracy further declines, potentially due to insufficient context.
- **Longer Sequences (512–700 tokens):** Accuracy improves steadily as more context is preserved, peaking at the longest tested sequence length.

Key Insight: Longer input sequences (512–700 tokens) perform better, as they retain critical context for accurate predictions.

6.2.4 Warm-up Steps vs Accuracy (Figure 4.4)

The impact of warm-up steps on validation accuracy after one epoch is outlined below:

- **No Warm-up Steps:** Lower accuracy due to unstable weight updates during the early training phase.
- **Optimal Warm-up Duration (500 steps):** Achieves the highest accuracy, balancing stability, and faster learning.
- **Excessive Warm-up Steps (1000–2000 steps):** Decrease accuracy, as the model takes too long to reach an effective learning rate.

Key Insight: A warm-up duration of approximately 500 steps is optimal, ensuring stable training while facilitating faster convergence.

6.3 Error Analysis

This analysis was conducted after training the T5 model for 3 epochs, demonstrating appreciable performance across different question categories, with a particular strength in handling simple, direct queries.

6.3.1 Category-wise Performance Analysis

Short Context (10 test cases)

- Correct: 8/10 (80%)
- Incorrect: 2/10 (20%)
- Performance Note: Model excels at processing concise contexts with straightforward questions

Long Context (10 test cases)

- Correct: 8/10 (80%)
- Incorrect: 2/10 (20%)
- Performance Note: Shows moderate performance degradation with increased context length, suggesting attention span limitations

Simple Yes Questions (10 test cases)

- Correct: 9/10 (90%)
- Incorrect: 1/10 (10%)
- Performance Note: Demonstrates strongest performance in affirming explicitly stated facts

Simple No Questions (10 test cases)

- Correct: 9/10 (90%)
- Incorrect: 1/10 (10%)
- Performance Note: Shows good capability in negative answer cases, similar to affirmative questions

Complex Questions (10 test cases)

- Correct: 6/10 (60%)
- Incorrect: 4/10 (40%)
- Performance Note: Demonstrates expected challenges with technical content and multi-step reasoning

Overall Performance

- Total Correct: 40/50 (80%)
- Total Incorrect: 10/50 (20%)

6.4 Future Improvements

This section outlines possible enhancements to the current approach that could further improve performance and robustness of the model:

1. **Gradient Accumulation for Larger Effective Batch Sizes:** Gradient accumulation allows simulating larger batch sizes without increasing GPU memory usage. This technique stabilizes gradients and improves generalization, especially in resource-constrained environments.
2. **Advanced Data Augmentation Techniques:** Techniques like back-translation, paraphrasing, and adding noise to inputs can diversify training data and enhance robustness. Domain-specific augmentations can further help the model handle edge cases and generalize in low-resource scenarios.
3. **Integration of External Knowledge Bases:** Incorporating knowledge bases like Wikidata or ConceptNet can improve the model's ability to handle complex reasoning and ambiguous contexts. Providing semantic information enhances predictions requiring domain expertise or background knowledge.
4. **Implementation of Adversarial Training:** Adversarial training uses challenging examples to improve robustness against noisy or incomplete data. This approach helps the model generalize better to unseen scenarios and strengthens its predictions in real-world tasks.

6.5 Conclusion

This work analyzed the base model T5's performance on a boolean question-answering dataset, providing insights into key parameters and error patterns. Adopting techniques like gradient accumulation, data augmentation, and adversarial training can further enhance performance. These refinements will enable the model to achieve greater accuracy and robustness, paving the way for state-of-the-art advancements in natural language processing.