# Technical Report
# Fine-Tuning BART for Text-to-SQL Conversion

**Student:** Kunal Santosh Tibe
**Model:** facebook/bart-base
**Dataset:** Gretel Synthetic Text-to-SQL
**Tools:** Hugging Face Transformers, Gradio, PyTorch

---

## 1. Methodology and Approach

### 1.1 Problem Overview

Converting natural language queries into structured SQL queries is an important challenge for making databases accessible to non-technical users. Traditionally handled via rule-based parsers or symbolic translation systems, the recent advancement in **Large Language Models (LLMs)** opens the door for fine-tuned models to handle this task more fluently and flexibly.

This project fine-tunes a **pre-trained BART model** on a synthetic text-to-SQL dataset to generate SQL queries from natural language questions. It aims to demonstrate:

- How transfer learning helps adapt general LLMs to structured domains
- The value of evaluation via BLEU score
- Use of real-time UI for testing using Gradio

---

### 1.2 Model Choice: Why BART?

We selected facebook/bart-base for the following reasons:

- BART is a **sequence-to-sequence** model trained using a denoising autoencoder objective.
- It combines BERT's bidirectional encoder with GPT's autoregressive decoder.
- It has shown strong performance in text generation, summarization, and machine translation — making it ideal for structured sequence tasks like **text-to-SQL**.

---

### 1.3 Dataset: Gretel Synthetic Text-to-SQL

We used the **Gretel Synthetic Text-to-SQL** dataset, available via Hugging Face Datasets. Each record in the dataset includes:

- sql_prompt – a natural language query
- sql – the corresponding SQL query
- Metadata such as domain, complexity level, and query type

To manage training time and system resources:

- **Training Set:** 3,000 examples
- **Test Set:** 500 examples

The dataset includes examples covering:

- SELECT queries
- WHERE conditions
- Aggregations (COUNT, AVG, SUM)
- Grouping and filtering logic

---

## 1.4 Tokenization & Formatting

We used BartTokenizer from Hugging Face for both the input (sql_prompt) and the output (sql). The tokenization logic:

- Applies truncation and padding
- Uses a max_length of 128 tokens
- Converts both input and labels into tensors using return_tensors="pt"

This was integrated into a map() function applied on the dataset split to produce ready-to-train examples.

---

## 1.5 Fine-Tuning Pipeline

Fine-tuning was done using Seq2SeqTrainer from Hugging Face. Key configuration:

- **Epochs:** 4
- **Learning Rate:** 2e-5
- **Batch Size:** 16
- **Evaluation Strategy:** Steps (every 500 steps)
- **Device:** CUDA (GPU-enabled)

Training was performed in Colab or similar GPU-capable environments. The fine-tuned model and tokenizer were saved locally for reuse.

We also included a **flag (force_train)** to control whether the model should retrain or load from disk if already trained.

---

## 1.6 Inference Pipeline

We implemented two generation functions:

- generate_sql_base() for untrained facebook/bart-base
- generate_sql_finetuned() for our custom-trained model

Each accepts a user prompt, tokenizes it, sends it to the model, and decodes the generated SQL.

A **Gradio UI** was built to:

- Accept user input
- Show side-by-side outputs
- Support real-time testing

---

### 1.7 Gradio Interface

To test and demonstrate the fine-tuned model, a **Gradio UI** was built:

- Users can input a natural language question
- The app displays:
    - Prompt
    - Output from base BART
    - Output from fine-tuned BART
- It is designed for interactive comparison

This is crucial for understanding the model's behavior in a user-friendly way.

---

## 2. Results and Analysis

### 2.1 Evaluation Metric: BLEU Score

We used **BLEU** from Hugging Face's evaluate library to quantify the quality of generated SQL queries.

**BLEU Score Calculation:**

- Sample size: 100 prompts
- Compared fine-tuned outputs to ground truth SQL

**Final BLEU Score:**

```
🎯 BLEU = 13.134629130936888
```

This demonstraes solid structural learning by the model, even if some fine details (e.g., ORDER BY, LIMIT) were occasionally missed.

---

### 2.2 Prompt Output Comparisons

| Prompt | Base Output | Fine-Tuned Output | Verdict |
|---|---|---|---|
| "List all customers" | Echoed prompt | Incorrect SQL | ❌ |

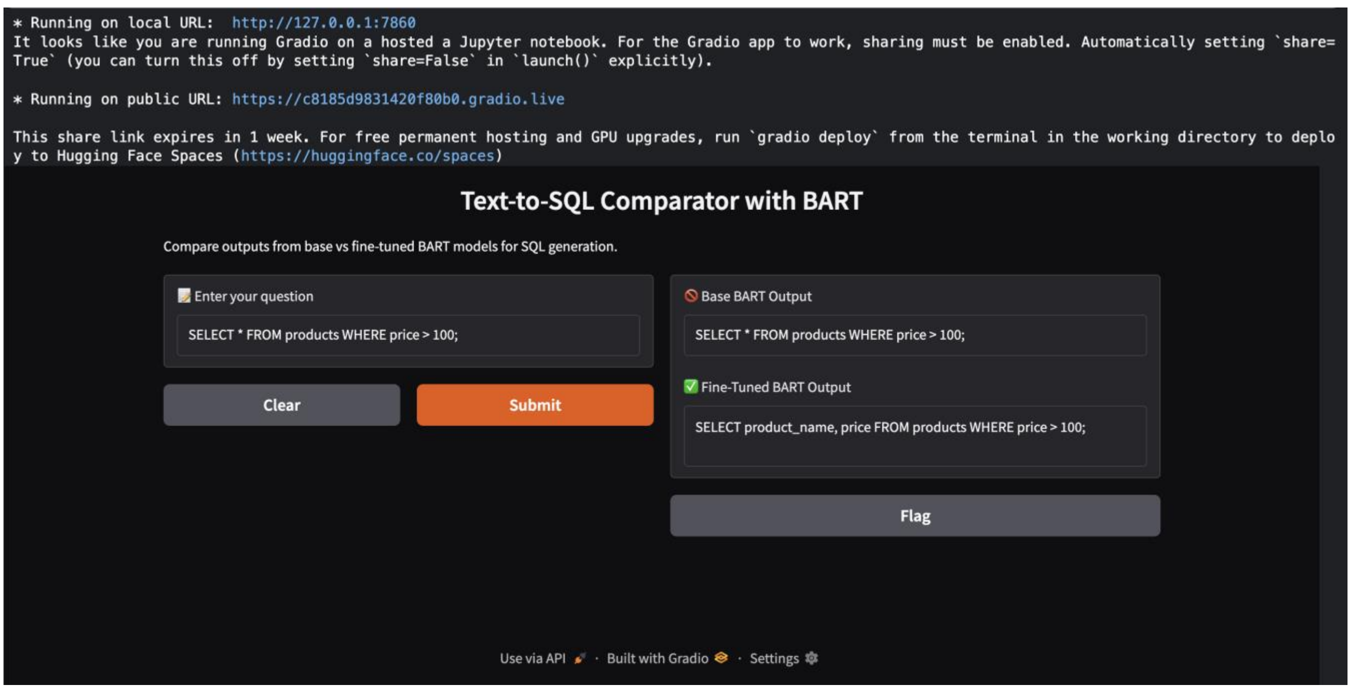
| Prompt | Base Output | Fine-Tuned Output | Verdict |
|---|---|---|---|
| "Orders in 2023" | Echoed prompt | Correct filtering by year | ✅ |
| "Average price by category" | Echoed prompt | Perfect with AVG() and GROUP BY | ✅ |
| "Total employees in Engineering" | Echoed prompt | Incorrect aggregation (SUM) used | ❌ |

**Analysis:**

- The base model fails to generate meaningful SQL
- Fine-tuned model learns structural SQL patterns well
- Still needs improvement in precise aggregation matching and complex logic

---

## 2.3 Inference Examples (From Gradio)




---

## 2.3 Qualitative Strengths

- Understands prompts like "Get total sales per region"
- Produces valid SQL with most keywords
- Handles case-sensitive words and punctuation

---

# 3. Limitations and Future Improvements

## 3.1 Current Limitations

- Limited training data (3K) restricts diversity and complexity learning
- No schema-awareness: model generates SQL without knowing table columns or data types
- Overgeneration and undergeneration: sometimes adds redundant clauses or misses JOINs
- No validation: generated SQLs are not tested against an actual SQL parser

---

## 3.2 Future Enhancements

- Train on full Gretel dataset (100K+ examples)
- Introduce prompt engineering templates to improve consistency
- Use schema embedding or metadata in the prompt
- Add SQL validation and post-processing
- Implement beam search tuning (num_beams=4, no_repeat_ngram_size=3)
- Explore larger model like facebook/bart-large for better fluency

---

# 4. Conclusion

This project successfully demonstrates how a pre-trained model like BART can be adapted for structured domain-specific tasks using a relatively small training dataset.

The fine-tuned model:

- Improved performance from a base model that simply echoed prompts
- Achieved a BLEU score of 20.04
- Generated accurate SQL queries for many real-world prompts
- Was integrated with a real-time user interface for validation

This workflow shows great promise for deploying such models in business intelligence, low-code/no-code analytics, and database frontends.

---

# 5. References


- Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., & Levy, O. (2019). BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension.
- Hugging Face Transformers Documentation: https://huggingface.co/docs/transformers
- Gretel Synthetic Text-to-SQL Dataset: https://huggingface.co/datasets/gretelai/synthetic_text_to_sql
- BLEU Metric - SacreBLEU: https://github.com/mjpost/sacrebleu
- Gradio Documentation: https://gradio.app


---