Natural Language to Visualization Mapping using Finetuned Flan-T5

Sahil Sahil

July 13, 2025

1 Introduction

This document details the process of fine-tuning a language model (Flan-T5) to interpret natural language plotting requests and map them to structured visualization instructions. The model is capable of generating a JSON-like output with keys `Method` and `Attribute`, which can later be used for automatic plot generation from signal data.

2 Objective

Given a sentence such as:

> "Create a line plot of engine speed"

the model should generate:

```
{
  "Method": "Line Plot",
  "Attribute": "Eng_nEng10ms"
}
```

3 Available Signals

Only three signal attributes are used in this version of the model:

- **Eng_nEng10ms**: Engine speed
- **Eng_uBatt**: Battery voltage in mV
- **FuSHp_pRailBnk1**: Fuel Pressure

4 Model Selection

The model used is **google/flan-t5-small**, a lightweight yet powerful transformer-based model capable of zero-shot and few-shot reasoning.

5 Dataset Preparation

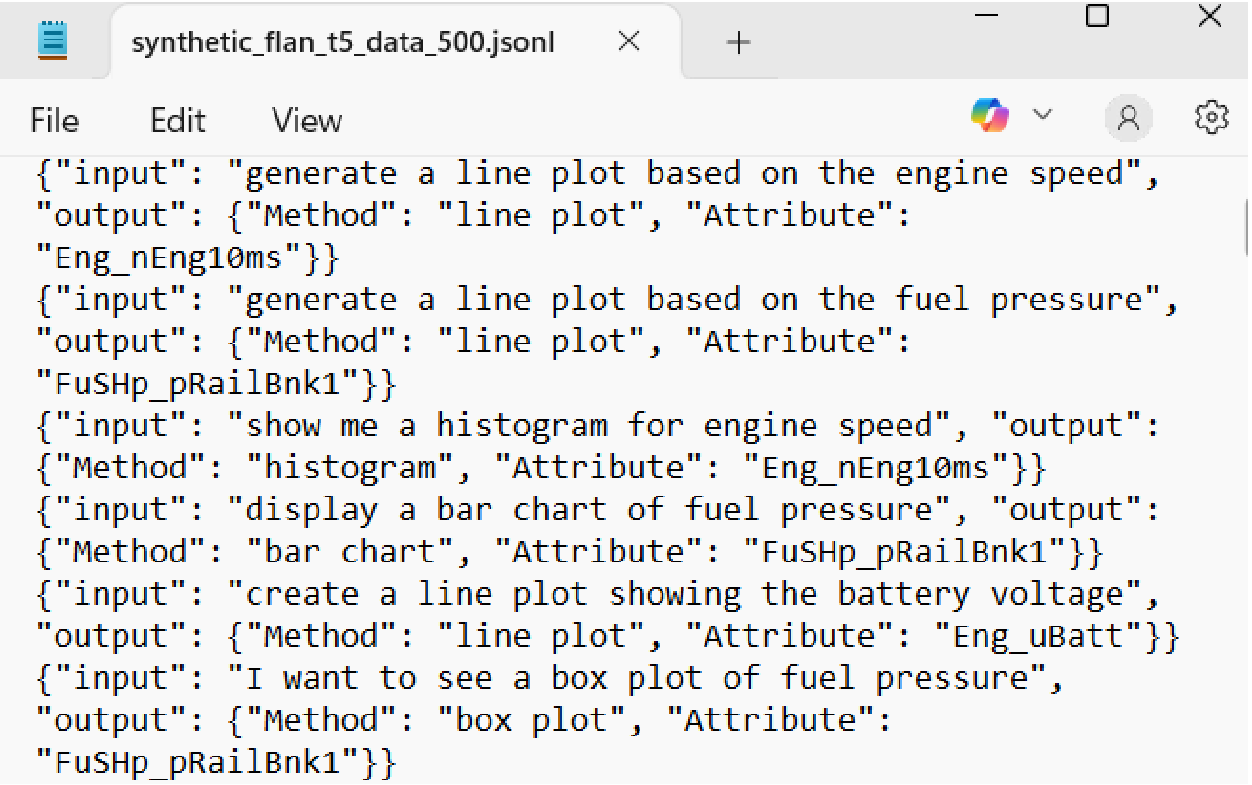
500 natural language sentences were synthetically generated to reflect requests for:

- Histogram
- Line Plot
- Bar Chart
- Scatter Plot
- Box Plot

Each sentence was paired with a corresponding output dictionary in string format. An example data pair:

- **Input:** "Show a bar chart for fuel pressure"
- **Output:** `{"Method": "Bar Chart", "Attribute": "FuSHp_pRailBnk1"}`

Screenshot of generated training dataset:



```
{"input": "generate a line plot based on the engine speed",
"output": {"Method": "line plot", "Attribute":
"Eng_nEng10ms"}}
{"input": "generate a line plot based on the fuel pressure",
"output": {"Method": "line plot", "Attribute":
"FuSHp_pRailBnk1"}}
{"input": "show me a histogram for engine speed", "output":
{"Method": "histogram", "Attribute": "Eng_nEng10ms"}}
{"input": "display a bar chart of fuel pressure", "output":
{"Method": "bar chart", "Attribute": "FuSHp_pRailBnk1"}}
{"input": "create a line plot showing the battery voltage",
"output": {"Method": "line plot", "Attribute": "Eng_uBatt"}}
{"input": "I want to see a box plot of fuel pressure",
"output": {"Method": "box plot", "Attribute":
"FuSHp_pRailBnk1"}}
```

6 Training

Tokenizer and Model

```
from transformers import T5Tokenizer, T5ForConditionalGeneration

tokenizer = T5Tokenizer.from_pretrained("google/flan-t5-small")
model = T5ForConditionalGeneration.from_pretrained("google/flan-t5-
   small")
```

Training Loop

The model was trained using Hugging Face's `Trainer` API with the following hyperparameters:

- Learning Rate: 5e-5
- Batch Size: 4
- Epochs: 3

```
trainer.train()
trainer.save_model("./flan_plot_mapper")
```

7 Inference

To generate visualization instructions from a new user query:

```
from transformers import T5Tokenizer, T5ForConditionalGeneration

# Load fine-tuned model and tokenizer
model_path = "./flan-t5-visualization"
tokenizer = T5Tokenizer.from_pretrained(model_path)
model = T5ForConditionalGeneration.from_pretrained(model_path)

# Inference function
def predict(text):
    inputs = tokenizer(text, return_tensors="pt")
    outputs = model.generate(**inputs, max_length=64)
    return tokenizer.decode(outputs[0], skip_special_tokens=True)

# Example
query = "ok␣now␣I␣want␣you␣to␣plot␣a␣nice␣histogram␣for␣battery␣
   voltage␣"
result = predict(query)
print("Prediction:", result)
```

Output:

```
Prediction: ’Method’: ’histogram’, ’Attribute’: ’Eng_nEng10ms’
```

8 Future Work

- Extend to all available signal names and attributes
- Incorporate multi-attribute plotting (e.g., “compare engine speed and fuel pressure”)
- Add support for filtering, time ranges, and data transformations

9 Conclusion

This work demonstrates a simple yet powerful use case of fine-tuned LLMs for domain-specific natural language understanding. Using Flan-T5, we created a prototype capable of converting plain-language plotting requests into structured commands suitable for automation.