Fine-tuning GPT-2 for Short Query Intent Classification

Jin Young Lee

June 9, 2025

Motivation and Problem Statement

Problem & Importance

- Users input short, ambiguous queries
- Predicting intent is key to user experience
- Applicable to voice assistants, chatbots, search engines
- Challenge: minimal context increases ambiguity

Example Queries

- "Weather?" $\rightarrow$ `weather_query`
- "Pizza nearby" $\rightarrow$ `find_restaurant`

Proposed Approach and Methodology

Model Architecture

- GPT-2 base model (768d hidden)
- Custom classification head
- Two fine-tuning modes:
  - Last-linear-layer only
  - Full-model fine-tuning
- Dropout (0.3) for regularization

Training Setup

- Optimizer: AdamW
- Learning rate: 1e-3
- Batch size: 8
- Cross-entropy loss
- Early stopping on dev accuracy

Dataset and Preprocessing

Amazon MASSIVE Dataset

- EN-US subset
- Multiple domains
- Short/long queries
- Labeled intent classes

Preprocessing Pipeline

- GPT-2 tokenizer
- Dynamic padding
- Truncation to max length
- EOS token as padding
- Unique IDs for tracking

Implementation Details

Model Components

- `GPT2IntentClassifier` class:
  - GPT-2 backbone with frozen/fine-tuned params
  - Linear classification head (768d $\rightarrow$ num_labels)
  - Dropout layer (0.1) before classification
- Custom dataset classes:
  - `IntentClassificationDataset` for train/dev
  - `IntentClassificationTestDataset` for test
- Evaluation metrics:
  - Accuracy and Macro F1 score
  - Per-class performance analysis

Training Progress

Loss and Metrics Tracking

- Training loss decreases steadily
- Validation metrics show convergence
- No significant overfitting observed
- Full-model fine-tuning shows better convergence

Figure: Training metrics over epochs

Experiments and Results

Quantitative Results

- Best performance:
  - Accuracy: 89.4%
  - Macro F1: 88.2%
- Fine-tuning comparison:
  - Full-model ¿ Last-layer (+3.5 F1)
  - Better generalization
- Performance patterns:
  - Strong on multi-word queries
  - Challenging for single-word inputs

Challenges and Learnings

Technical Challenges

- Model architecture:
  - Balancing model capacity vs. overfitting
  - Optimal dropout rates (0.3 for GPT, 0.1 for head)
- Training dynamics:
  - Learning rate sensitivity
  - Batch size limitations (GPU memory)
- Data processing:
  - Tokenization edge cases
  - Padding strategy impact

Future Work

Technical Improvements

- Model efficiency:
  - Implement LoRA for efficient fine-tuning
  - Explore model quantization
- Architecture enhancements:
  - Add attention visualization
  - Implement confidence scoring
- Training improvements:
  - Learning rate scheduling
  - Gradient accumulation

Q&A Preparation

Technical Questions

- Why GPT-2 over BERT?
- Handling of out-of-vocab tokens?
- Memory requirements?

Implementation Questions

- Training time and resources?
- Deployment considerations?
- Error analysis methodology?