Fine-tuning GPT-2 for Short Query Intent Classification

Jin Young Lee

June 11, 2025

Understanding Search Intent

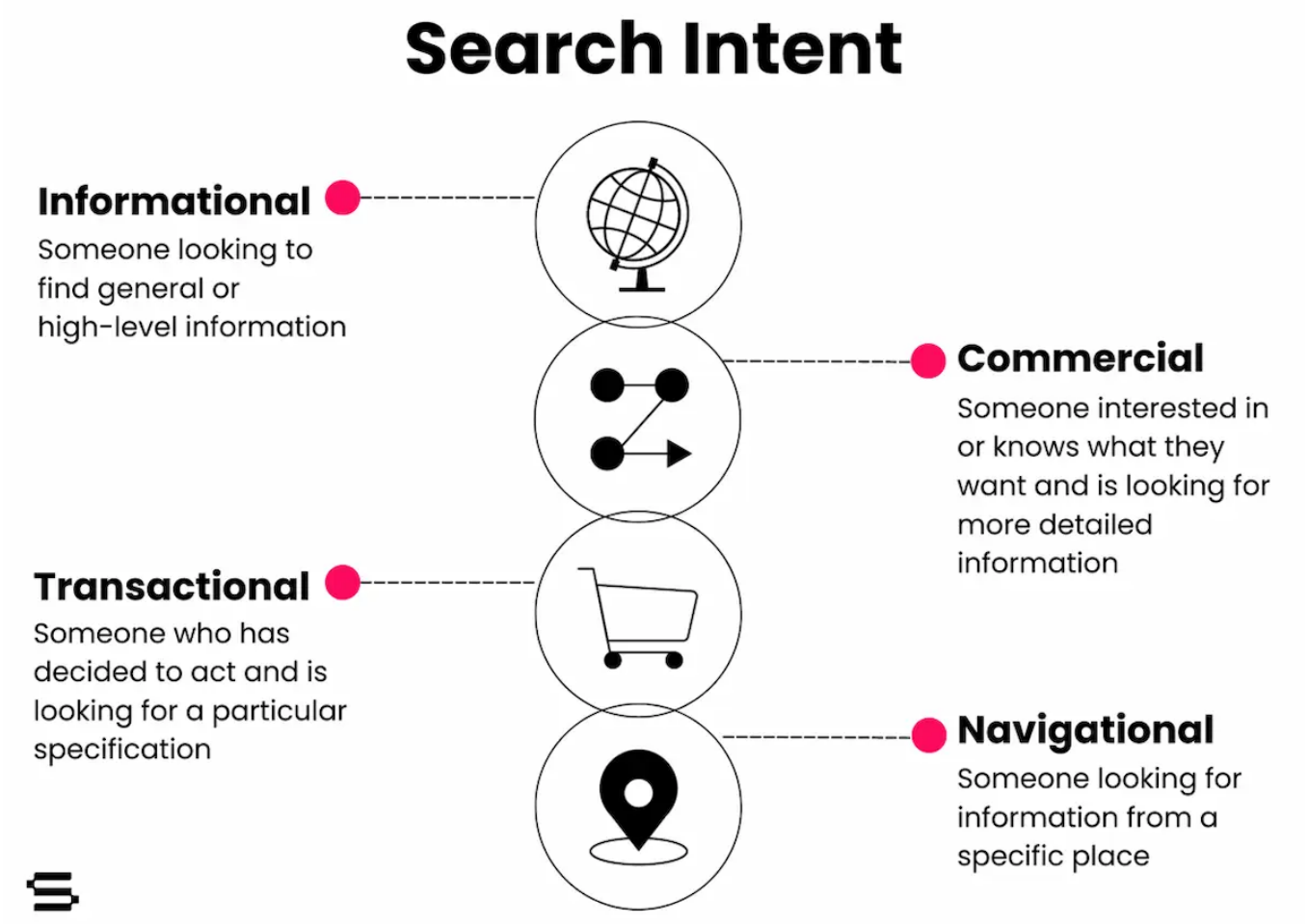


Figure: Different types of Search Intent

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

Category	Detail
Model	GPT-2 base model (768d hidden)
Classification Head	Custom linear layer
Fine-tuning Modes	Last-linear-layer, Full-model
Regularization	Dropout (0.3)
Optimizer	AdamW
Learning Rate	1e-3
Batch Size	8
Loss Function	Cross-entropy
Early Stopping	Based on dev accuracy

Amazon MASSIVE Dataset (EN-US Subset)

[View on Hugging Face]

Feature	Description
Total Languages	51 (Multilingual)
Subset Used	`en-US` only
Utterance Count	$\sim$60,000 utterances (EN-US)
Intent Classes	60 distinct intent types
Domains	Music, Weather, Alarms, Smart Home, etc.
Utterance Length	Mix of short and long queries
Label Quality	Human-annotated, high quality
Source	Amazon Alexa / MASSIVE Dataset

MASSIVE Dataset: EN-US Subset Examples

ID	Utterance	Intent Label
1	wake me up at nine am on friday	alarm_set
2	set an alarm for two hours from now	alarm_set
5	stop	audio_volume_mute
9	make the lighting bit more warm here	iot_hue_lightchange
15	turn off the light in the bathroom	iot_hue_lightoff
22	dim the lights in the kitchen	iot_hue_lightdim
25	olly clean the flat	iot_cleaning
33	check when the show starts	calendar_query
34	i want to listen arijit singh song once again	play_music

Intent types include alarms, smart lighting, music playback, and calendar access.

Data Preprocessing Pipeline - Input Example

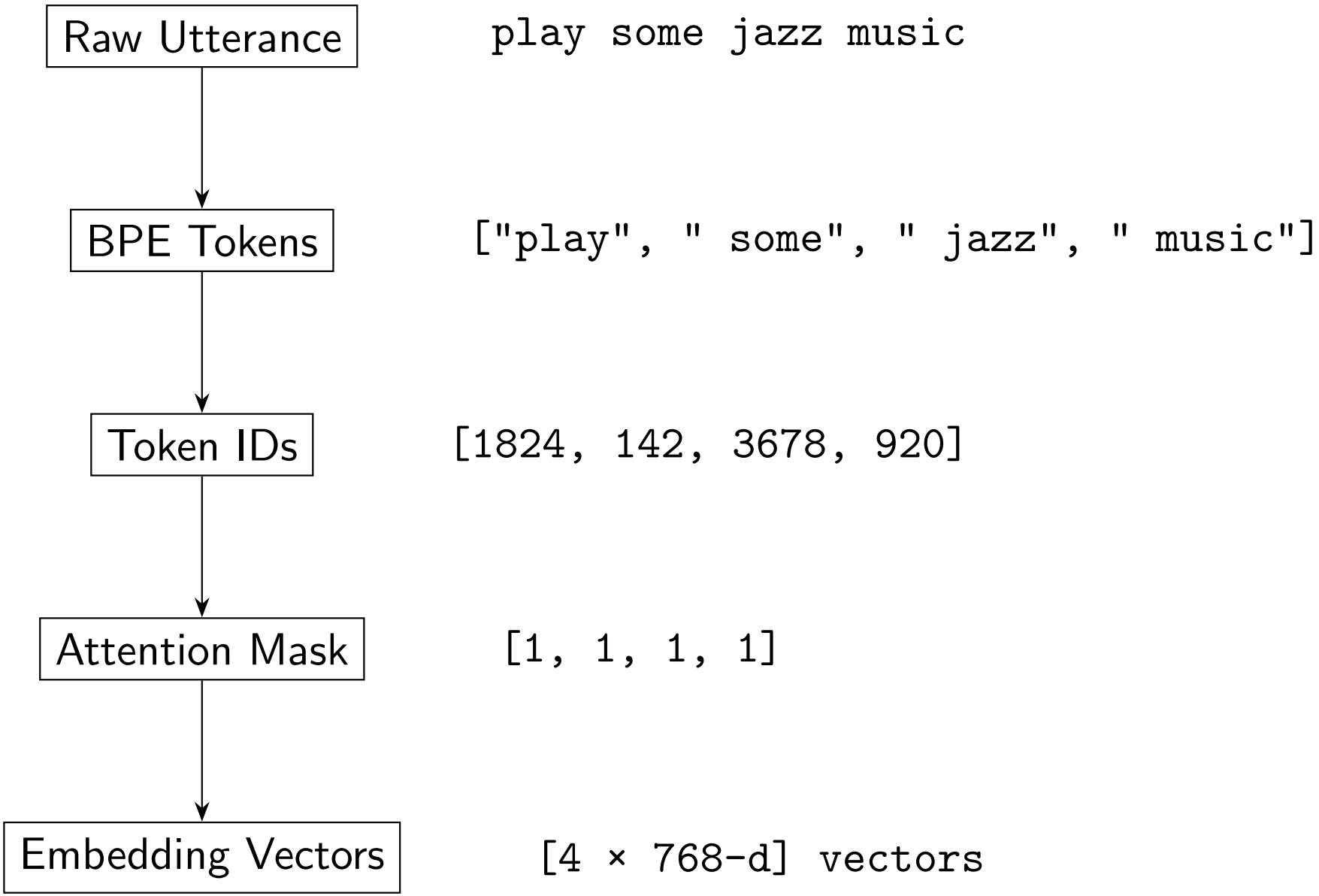
Preprocessing Steps for GPT-2:

- Tokenize input using GPT-2 tokenizer (BPE-based)
- Apply dynamic padding within batch
- Truncate to max model input length (e.g., 128)
- Use <|endoftext|> as padding token
- Track unique utterance ID for evaluation alignment

Example Utterance:

- **Text:** "play some jazz music"
- **Intent:** `music.play_song`

Tokenization Flow: From Text to Embeddings



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

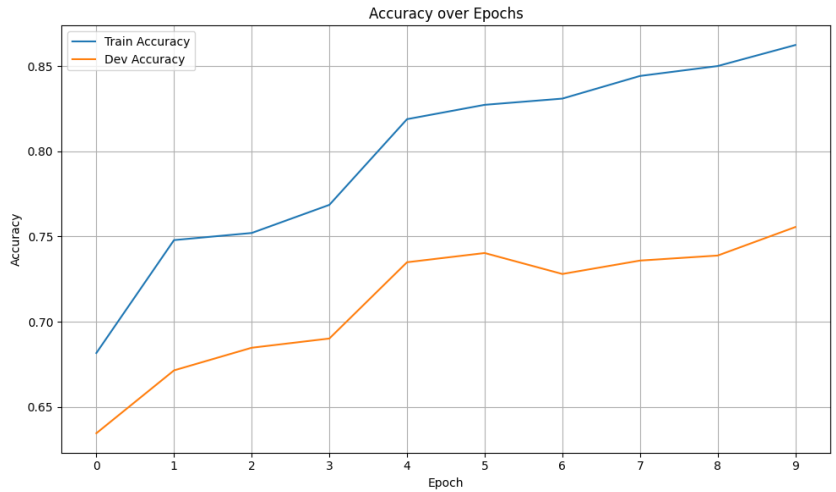


Figure: Example: Accuracy over Epochs

Training and Development Loss

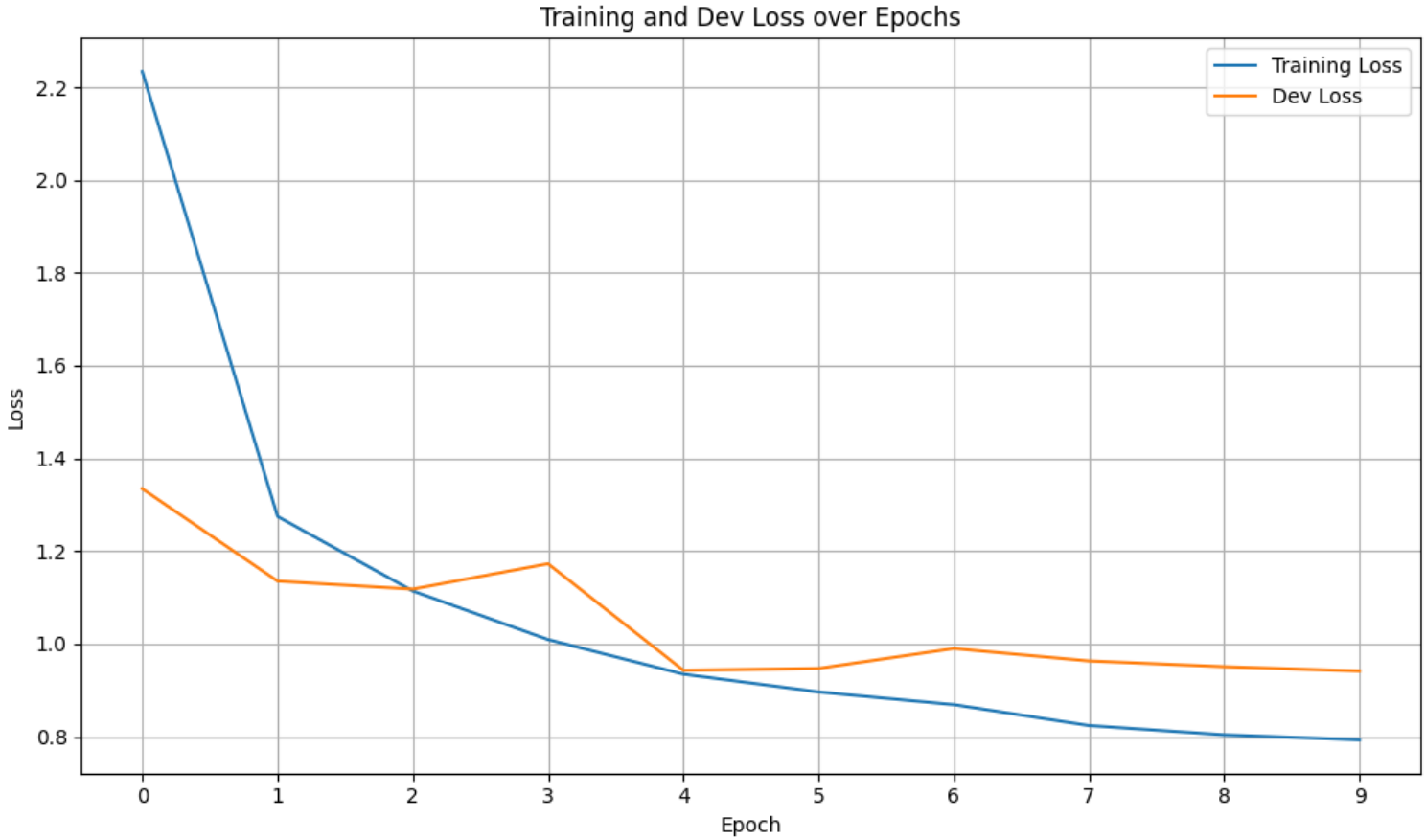


Figure: Training and Development Loss over Epochs

Accuracy over Epochs

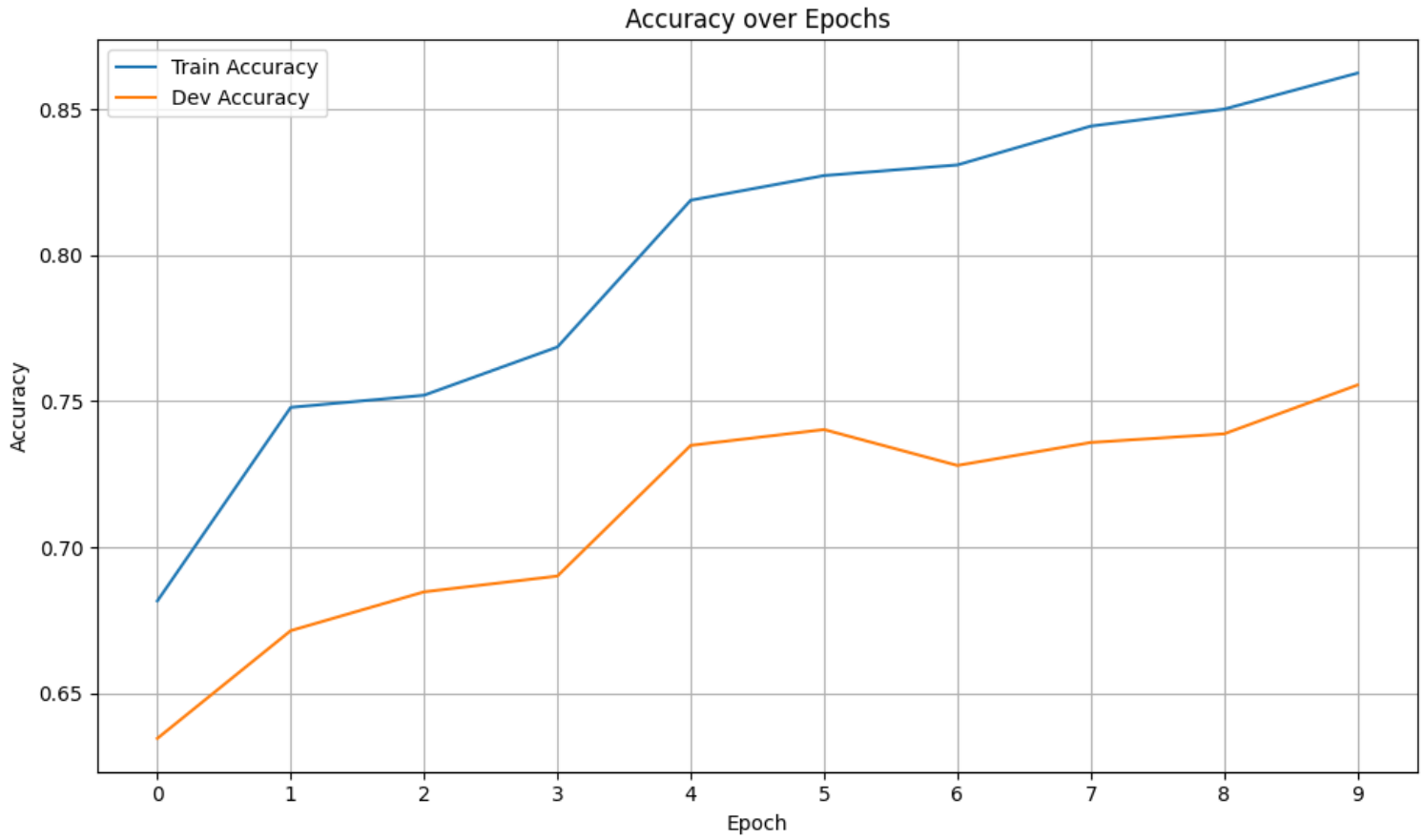


Figure: Accuracy Performance over Epochs (Last-Linear-Layer)

Accuracy Comparison: Last-Linear-Layer vs. Full-Model

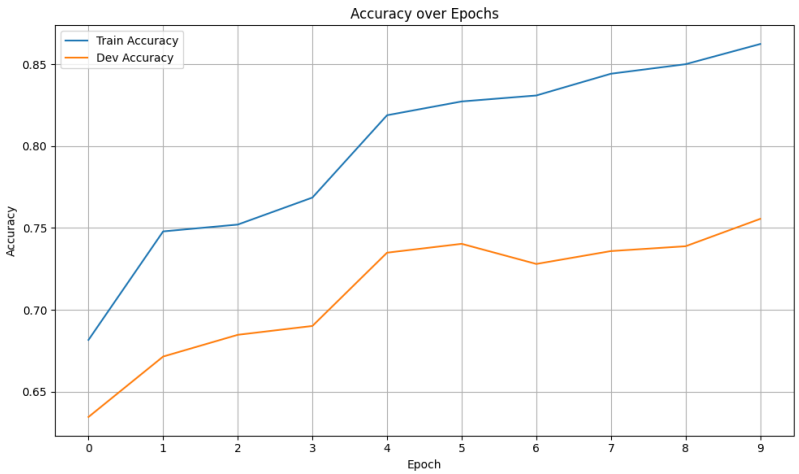


Figure: Last-Linear-Layer Accuracy

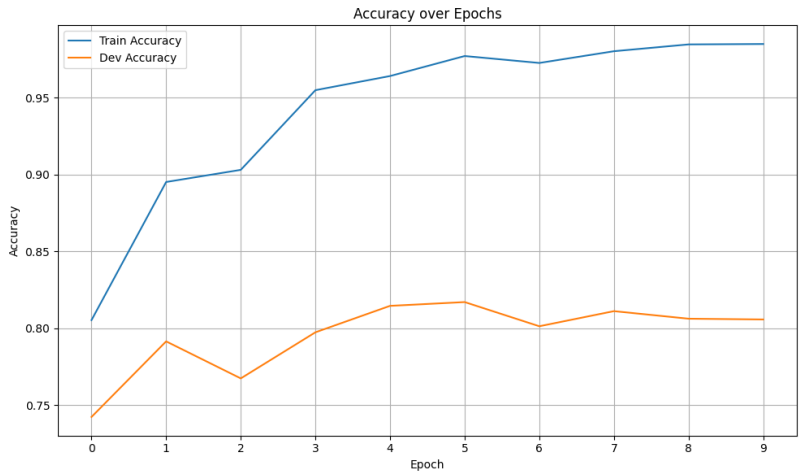


Figure: Full-Model Accuracy

F1 Scores over Epochs

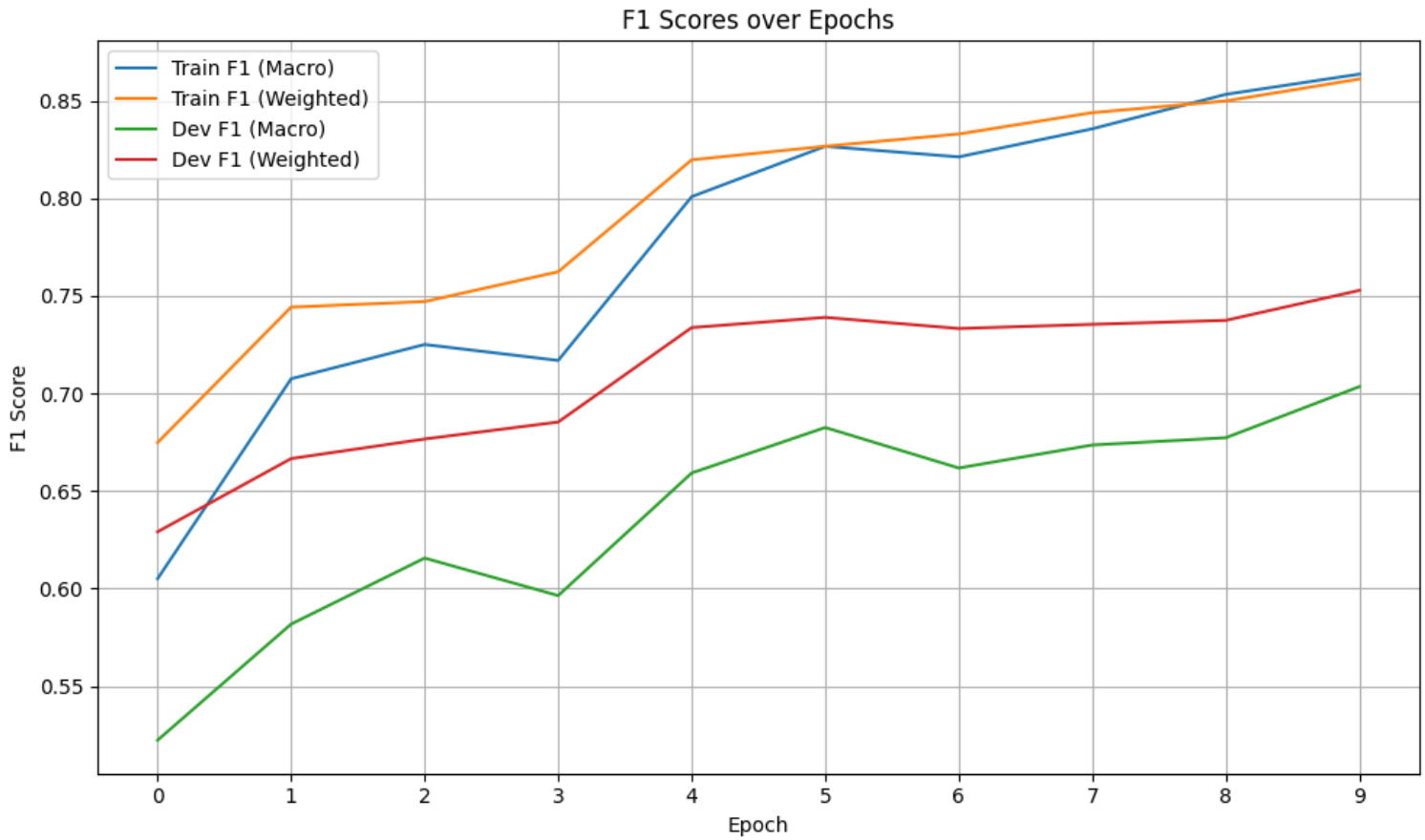


Figure: F1 Score Performance over Epochs

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