Saphalya Peta sqp6215 939607581

Project_P2: Finetuning models for Domain-Specific Tasks

Dataset Creation Process:

Dataset Overview:

- Total Training Examples: 440 voice command pairs, carefully curated examples
- Data Format: JSONL with structured conversation format

Data Collection Methodology:

1. Comprehensive Command Coverage:
   - Navigation commands: "go home", "go back", "scroll up/down"
   - Application control: "open calculator", "launch messages", "start camera"
   - Interaction commands: "tap 1", "select item", "click button"
   - System commands: "show numbers", "hide numbers", "start dictation"
2. Natural Language Variations:
   - Included polite requests: "please", "could you", "would you mind"
   - Added filler words: "um", "ah", "like", "you know"
   - Covered different phrasing styles and regional variations
   - Incorporated common speech patterns and colloquialisms
3. Quality Assurance Process:
   - Manual verification of each command mapping
   - Consistency checks across similar commands
   - Validation of VUI command syntax and structure
   - Multiple rounds of testing and refinement

Data Distribution:

1. Navigation: 26.6%
2. App Control: 33.3%
3. Interaction: 24.4%
4. System Commands: 15.7%

Model Used and Fine-Tuning Parameters:

1. Base Model Specifications:
   - Model Architecture: Gemma 3 270M
   - Framework: Unsloth (Optimized Training)
   - Context Length: 2048 tokens
   - Model Type: Transformer-based Causal Language Model
   - Training Approach: Supervised Fine-Tuning (SFT)
2. Training Environment Details:
   - Effective Batch Size: 8 (2 × 4 gradient accumulation)
   - Total Training Time: ~15-20 minutes
   - Hardware: Google Colab Tesla T4 GPU
   - Memory Optimization: Carefully tuned for 15GB VRAM constraints
   - Convergence: Stable training with consistent loss decrease
3. Model Output Specifications:
   - Output Format: Structured VUI commands
   - Command Structure: [ACTION] [target]
   - Valid Actions: CLICK, OPEN, SELECT, GO, SCROLL, TAP, SHOW, HIDE
   - Target Types: App names, numbers, directions, system functions

Example Inputs and Outputs:

1. Basic Navigation Commands:

Input: "please go home"

Output: "GO home"

Input: "I need to go back"

Output: "GO back"

Input: "scroll down a bit"

Output: "SCROLL down"

Input: "can you scroll up please"

Output: "SCROLL up"

2. Application Control Commands:

Input: "open calculator app"

Output: "OPEN calculator"

Input: "launch camera application"

Output: "OPEN camera"

Input: "start messages please"

Output: "OPEN messages"

Input: "open photos application"

Output: "OPEN photos"

Interaction Commands

Input: "tap number 3"

Output: "TAP 3"

Input: "click on the first item"

Output: "CLICK 1"

Input: "select the second option"

Output: "SELECT 2"

Input: "press button five"

Output: "CLICK 5"

3. System Commands:

Input: "show numbers on screen"

Output: "SHOW NUMBERS"

Input: "hide the numbers please"

Output: "HIDE NUMBERS"

Input: "start dictation mode"

Output: "START DICTATION"

Input: "stop voice typing"

Output: "STOP DICTATION"

4. Complex Natural Language Handling:

Input: "I was wondering if you could possibly open the calculator for me"

Output: "OPEN calculator"

Input: "um, could you like, go back to the previous screen please"

Output: "GO back"

Input: "hey, I need to tap on the third item in the list"

Output: "TAP 3"

Input: "please would you be able to show the numbers on the display"

Output: "SHOW NUMBERS"

Issues Encountered and Potential Improvements:

1. Model Performance Issues:

a. Issue: Spelling inconsistencies in output commands

Example: "calculator" → "CALCULARY"

Example: "messages" → "MESSAGE" or "PLAY"

Root Cause: Limited training data for specific app names

Solution Implemented: Added targeted correction examples and post-processing

b. Issue: Missing action words in commands

Example: "show numbers" → "NUMBERS" (missing "SHOW")

Example: "hide numbers" → "NUMBERS" (missing "HIDE")

Root Cause: Model sometimes omits action verbs

Solution Implemented: Enhanced training with explicit action reinforcement

2. Training Technical Challenges:

a. Issue: Quantization limitations for retraining

Problem: Cannot fine-tune 4-bit quantized models

Solution: Used base model with full training pipeline

Impact: Required complete retraining but ensured model quality

b. Issue: Memory constraints on T4 GPU

Problem: Limited VRAM for large batch sizes

Solution: Optimized batch size (2) with gradient accumulation (4)

Result: Stable training within hardware limitations

3. System Integration Issues:

a. Issue: Async/Sync context conflicts in Streamlit

Problem: Streamlit's synchronous nature vs async MCP calls

Solution: Wrapped async operations in synchronous execution

Benefit: Smooth user experience without blocking

b. Issue: Model format compatibility

Problem: GGUF vs safetensors deployment confusion

Solution: Standardized on GGUF format for Ollama integration

Result: Consistent model deployment and inference

Performance Limitations: Current Accuracy Assessment

Command Type	**Success Rate**	**Major Issues**
Basic Navigation	95%	Minimal issues
App Control	90%	Occasional spelling variations
Number Interaction	85%	Consistent but room for improvement
Complex Language	80%	Struggles with very long phrases
Unseen Commands	65%	Limited generalization

Specific Performance Gaps:

- Vocabulary Limitations: Model occasionally invents non-standard terms
- Context Understanding: Limited ability to handle multi-step commands
- Error Recovery: No graceful handling of completely unrecognized commands
- Confidence Scoring: Cannot identify low-confidence predictions

Potential Improvements:

1. Immediate Short-term Improvements (1-2 weeks):
   a. Enhanced Training Data:
      - Expand to 1000+ examples with more diversity
      - Add regional and dialect variations
      - Include more edge cases and error scenarios
      - Incorporate user feedback from initial testing

   b. Model Architecture Optimizations:
      - Experiment with different model sizes (1B, 2B parameters)
      - Implement ensemble methods for better accuracy
      - Add confidence scoring for predictions

- Incorporate fallback mechanisms

c. System Enhancements:
- Add voice input capability
- Implement command history and favorites
- Create user customization options
- Add multi-language support foundation

2. Medium-term Enhancements (1-3 months):

a. Advanced Features:
- Context-aware command processing
- Multi-turn conversation support
- Personalized command learning
- Cross-platform compatibility expansion

b. Performance Optimizations:
- Reduce inference latency to <1 second
- Implement model quantization for faster deployment
- Add caching for frequent commands
- Optimize TTS for better voice quality

c. User Experience Improvements:
- Visual command confirmation
- Progressive disclosure of advanced features
- Better error messages and suggestions
- Tutorial and onboarding flow

3. Long-term Vision (3-6 months+):

a. Intelligence Enhancements:
- Adaptive learning from user corrections
- Predictive command suggestions
- Emotional tone recognition
- Multi-modal input support (voice + gesture)

b. Platform Expansion:
- Support for additional VUI systems (Alexa, Google Assistant)
- Cross-device synchronization
- Enterprise features and customization
- API for third-party integrations

c. Advanced Capabilities:
- Natural language understanding for complex tasks

- Proactive assistance and suggestions
- Integration with smart home ecosystems
- Accessibility features for diverse user needs

Conclusion:

The current system demonstrates strong foundational capabilities with approximately 85-90% accuracy on core voice command tasks. The main issues revolve around vocabulary consistency and handling of complex natural language constructs. The proposed improvement roadmap provides a clear path from immediate fixes to long-term advanced features, ensuring continuous enhancement of the system's capabilities and user experience.

The architecture is designed to be extensible, allowing for incremental improvements without major refactoring. The focus on data quality and model optimization in the short term will provide the most significant immediate benefits to users.