Military Behavior Tree Generation System: AI-Driven Tactical Decision Making

Project Overview

An innovative system that leverages Large Language Models (LLMs) with LoRA adaptors to generate military behavior trees, combining doctrinal knowledge with AI to create sophisticated tactical decision-making structures.

Core Architecture

1. **AI Foundation**
    a. Base Model: CodeLLaMa (7B parameters) optimized for code generation
    b. LoRA Adapters: Fine-tuned for military-specific behavior tree generation
    c. Specialized node handlers for formations, movement, combat, and tactical operations
2. **Knowledge Integration**
    a. RAG (Retrieval-Augmented Generation) pipeline incorporating military doctrine
    b. Pinecone vector database for efficient context retrieval
    c. Dynamic context injection before LLM generation
    d. Comprehensive military document categorization including:
        i. Formation management
        ii. Movement protocols
        iii. Combat engagement rules
        iv. Tactical evaluations
        v. Unit coordination
        vi. Battlefield conditions
3. **Technical Implementation**
    a. FastAPI backend server for handling generation requests
    b. Vite-powered frontend for behavior tree visualization
    c. GPU acceleration support for model inference
    d. Containerized deployment with conda environment management

BTGenBot Description

The AIRLab-POLIMI's BTGenBot project provides insights and capabilities that can enhance our military behavior tree generation system. Their work demonstrates the effectiveness of using lightweight LLMs (7B parameters) - the same scale as our CodeLLama implementation - for generating behavior trees, particularly when fine-tuned. Additionally, their BT Client system, which can directly execute generated behavior trees on robots, suggests a pathway for implementing our military behavior trees in autonomous simulations in Unreal Engine. Their successful results using LoRA adapters for fine-tuning (available on HuggingFace) validate our approach of using LoRA for military-specific adaptations. We could potentially incorporate their validation framework and client execution system while maintaining our military-focused RAG pipeline and specialized node categories for tactical operations.

Link: [https://github.com/AIRLab-POLIMI/BTGenBot]

Pinecone Description

We're using Pinecone in this project as our vector database for implementing the Retrieval-Augmented Generation (RAG) pipeline, which is crucial for incorporating military doctrine into our behavior tree generation. Pinecone excels at efficiently storing and retrieving high-dimensional vector embeddings of military doctrinal texts, allowing us to perform semantic similarity searches at scale. Additionally, Pinecone's hybrid search capabilities allow us to combine semantic similarity with metadata filtering, enabling us to narrow searches to specific types of military operations or doctrinal categories, making the context retrieval more precise and relevant to the specific tactical scenario being modeled.

Technical Stack

1. **Backend**: Python, FastAPI, Torch
2. **AI**: CodeLLaMa, LoRA, Transformers
3. **Database**: Pinecone
4. **Frontend**: Vite, React
5. **DevOps**: Conda, Docker

Initial Setup on Newton

1. The CodeLLaMa model needs to be downloaded from Hugging Face and stored on Newton's file system
2. Our custom LoRA adapters (fine-tuned for military behavior trees) must be uploaded to Newton

3. Both model and adapters are placed in designated directories accessible by the compute nodes
4. **Process Flow**

[Local Machine] -> [Newton Login Node] -> [Compute Node] -> [Results] -> [Local Machine]

Execution Process

1. SSH connection is established to Newton from local machine
2. SLURM job submission script allocates resources:
3. Demo file (`demo_ssh.py`) is executed on the allocated compute node
4. *Slurm Parameters*

   ```
   #SBATCH --nodes=1

   #SBATCH --gpus=1

   #SBATCH --time=02:00:00
   ```

Demo File Operation Flow

1. **Model Initialization**
   a. Loads CodeLLaMa base model from Newton's storage
   b. Applies LoRA adapter for military behavior tree specialization
   c. Configures GPU settings (uses CUDA if available)
2. **Scenario Processing**
   a. Reads scenarios from JSON input file
   b. For each scenario:
      i. Retrieves relevant military doctrine from Pinecone
      ii. Constructs enhanced prompt with context
      iii. Generates behavior tree using the model
      iv. Extracts and formats XML output
      v. Saves behavior tree and metadata
3. **Output Management**
   a. Creates organized directory structure for outputs
   b. Saves XML behavior trees
   c. Stores metadata about generation process

Future Enhancements

1. Additional specialized node types
2. Enhanced doctrine integration
3. Real-time adaptation capabilities
4. Multi-scenario optimization
5. Advanced frontend visualization features