Indian Legal Corpus LLM Fine-tuning Project

A comprehensive solution for fine-tuning Large Language Models (specifically Qwen 1.5B/2.5B) on Indian legal questions and answers using LoRA (Low-Rank Adaptation) for efficient training.

🎯 Project Overview

This project provides a complete pipeline for:

- Loading and preprocessing Indian legal corpus data
- Fine-tuning Qwen foundation models using LoRA
- Running inference on legal questions
- Evaluating model performance
- Interactive web interface using Streamlit

🏗️ Architecture

```
indian-legal-llm/
├── config.py              # Configuration settings
├── data_loader.py         # Data loading and preprocessing
├── fine_tuner.py          # Model fine-tuning with LoRA
├── inference.py           # Inference engine
├── evaluator.py           # Model evaluation
├── streamlit_app.py       # Web UI (Streamlit)
├── main.py                # Main execution script
├── requirements.txt       # Dependencies
├── data/                  # Data directory
├── models/                # Trained models
└── logs/                  # Training and evaluation logs
```

🚀 Quick Start

1. Installation

```bash
```

```
# Clone the repository
git clone <your-repo-url>
cd indian-legal-llm

# Install dependencies
pip install -r requirements.txt

# Or using conda
conda create -n legal-llm python=3.9
conda activate legal-llm
pip install -r requirements.txt
```

2. Run the Complete Pipeline

```
bash

# Run everything: data preparation, training, evaluation, and interactive chat
python main.py --mode full

# Or launch the web interface
python main.py --mode ui
# Then open your browser to http://localhost:8501
```

📚 Detailed Usage

Data Preparation

```
bash

# Prepare data from HuggingFace dataset
python main.py --mode data --data_source huggingface

# Or use your own CSV file (place in data/ directory)
python main.py --mode data --data_source csv
```

Model Training

```
bash
```

```
# Train model with default settings
python main.py --mode train --data_source huggingface

# Training will use LoRA for efficient fine-tuning
# Configurations can be modified in config.py
```

Inference

bash

```
# Interactive chat mode
python main.py --mode inference --interactive

# Single question
python main.py --mode inference --question "What is Section 302 of IPC?"

# Use specific model
python main.py --mode inference --model_path models/legal_qwen_merged --interactive
```

Evaluation

bash

```
# Evaluate model performance
python main.py --mode evaluate --eval_samples 20

# Evaluate specific model
python main.py --mode evaluate --model_path models/legal_qwen_merged --eval_samples 10
```

Web Interface (Streamlit)

bash

```
# Launch the web interface
streamlit run streamlit_app.py
```

Features of the web interface:

- **Chat Interface**: Interactive legal Q&A
- **Training Dashboard**: Configure and monitor training
- **Evaluation Tools**: Test model performance

- **Model Management**: Load different trained models

⚙️ Configuration

Edit `config.py` to customize:

```python
# Model settings
MODEL_CONFIG.base_model_name = "Qwen/Qwen2.5-1.5B-Instruct"
MODEL_CONFIG.max_length = 512
MODEL_CONFIG.temperature = 0.7

# Training settings
TRAINING_CONFIG.num_train_epochs = 3
TRAINING_CONFIG.learning_rate = 2e-4
TRAINING_CONFIG.per_device_train_batch_size = 4

# LoRA settings
LORA_CONFIG.r = 16
LORA_CONFIG.lora_alpha = 32
LORA_CONFIG.lora_dropout = 0.1
```

📊 Model Evaluation

The evaluation module provides:

- **Answer Length Analysis**: Average words per response
- **Legal Terms Coverage**: Percentage of responses containing legal terminology
- **Relevance Score**: Measure of response relevance to questions
- **Success Rate**: Percentage of successful generations
- **Sample Outputs**: Manual review of generated responses

🎯 Features

Core Features

- ✅ **Data Loading**: HuggingFace dataset integration + CSV support
- ✅ **Efficient Training**: LoRA fine-tuning for resource optimization
- ✅ **Model Inference**: Fast response generation for legal questions
- ✅ **Comprehensive Evaluation**: Automated model performance assessment

- ✅ **Model Management**: Save, load, and merge trained models

Bonus Features

- ✅ **Streamlit Web UI**: User-friendly interface
- ✅ **Toast Notifications**: Real-time feedback
- ✅ **Loading States**: Progress indicators
- ✅ **Chat History**: Conversation management
- ✅ **Export Options**: Download results and chat logs

📁 Data Format

Input Data (CSV)

```csv
question,answer
"What is Section 302 of IPC?","Section 302 deals with punishment for murder..."
"Define bail","Bail is the temporary release of an accused person..."
```

Training Data (JSONL)

```json
{
  "prompt": "<|im_start|>system\nYou are a helpful assistant...",
  "completion": "Section 302 of the Indian Penal Code...<|im_end|>",
  "text": "Complete formatted text for training"
}
```

🔧 Technical Details

Model Architecture

- **Base Model**: Qwen 1.5B/2.5B Instruct
- **Fine-tuning Method**: LoRA (Low-Rank Adaptation)
- **Target Modules**: Query, Key, Value projection layers
- **Precision**: FP16 for efficient training

Training Process

1. **Data Preprocessing**: Clean and format legal Q&A pairs

2. **Tokenization**: Convert text to model-compatible format
3. **LoRA Setup**: Apply efficient fine-tuning adapters
4. **Training**: Supervised fine-tuning on legal corpus
5. **Merging**: Combine LoRA weights with base model
6. **Evaluation**: Test on held-out samples

Performance Optimizations

- **Gradient Accumulation**: Handle larger effective batch sizes
- **Mixed Precision**: FP16 training for speed and memory
- **LoRA**: Reduce trainable parameters by ~99%
- **Efficient Data Loading**: Parallel processing and caching

📈 Expected Results

After training, you should see:

- **Legal Knowledge**: Accurate responses to IPC, CrPC, and constitutional questions
- **Context Understanding**: Proper interpretation of legal scenarios
- **Terminology Usage**: Appropriate legal vocabulary and concepts
- **Response Quality**: Well-structured and informative answers

🛠️ Troubleshooting

Common Issues

1. **Out of Memory**
   - Reduce `per_device_train_batch_size` in config.py
   - Enable gradient checkpointing
   - Use smaller model variant
2. **Slow Training**
   - Ensure CUDA is available
   - Check GPU utilization
   - Reduce sequence length if needed
3. **Poor Model Performance**
   - Increase training epochs
   - Adjust learning rate

- Verify data quality and formatting

4. **Module Import Errors**
   - Check all dependencies are installed
   - Verify Python environment
   - Ensure all files are in the same directory

📝 Development Notes

Extending the Project

1. **Add New Data Sources**
   - Implement new loaders in `data_loader.py`
   - Support additional legal domains
2. **Model Improvements**
   - Experiment with different LoRA configurations
   - Try other base models (Llama, Mistral)
   - Implement more sophisticated evaluation metrics
3. **UI Enhancements**
   - Add more visualization charts
   - Implement user authentication
   - Add model comparison features

📋 Deliverables Checklist

- ☑ **GitHub Repository**: Complete modular codebase
- ☑ **Data Processing**: Load from HuggingFace + CSV support
- ☑ **Model Training**: LoRA fine-tuning implementation
- ☑ **Inference Engine**: Question answering system
- ☑ **Evaluation Suite**: Comprehensive performance testing
- ☑ **Web Interface**: Streamlit UI with notifications
- ☑ **Documentation**: Complete setup and usage guide
- ☑ **Sample Results**: Evaluation reports and examples

🔗 Useful Links

- HuggingFace Dataset
- Qwen Model Documentation

- LoRA Paper
- Streamlit Documentation

📄 License

This project is licensed under the MIT License - see the LICENSE file for details.

🤝 Contributing

1. Fork the repository
2. Create your feature branch (`git checkout -b feature/AmazingFeature`)
3. Commit your changes (`git commit -m 'Add some AmazingFeature'`)
4. Push to the branch (`git push origin feature/AmazingFeature`)
5. Open a Pull Request

📞 Support

For questions or issues:

- Create an issue in this repository
- Check the troubleshooting section above
- Review the configuration options in `config.py`

Happy Legal AI Building! ⚖️🤖