Team Members

Abdelrahman Tarek Zaki

Mazen Gaber Mahmoud

English to French Translation Project Report

Project Overview

This project implements an English to French translation service using deep learning models. The application offers two different translation models: a Seq2Seq LSTM model and a T5 Transformer model. The service is exposed through a FastAPI backend with both REST API endpoints and a user-friendly web interface.

Architecture

The project follows a modular architecture with clear separation of concerns:

1. **API Layer**: FastAPI application with routes for handling translation requests
2. **Model Layer**: Two translation models (Seq2Seq LSTM and T5 Transformer)

3. **Configuration**: Environment-based configuration system
4. **Web Interface**: Simple HTML/CSS/JS frontend for end users

Project Structure

```
├── main.py              # FastAPI application entry point
├── src/
│   ├── __init__.py      # Package initialization
│   ├── config.py        # Configuration handling
│   ├── inference.py     # Model inference functionality
│   └── request.py       # API request handling
├── templates/           # HTML templates for web interface
├── artifacts/           # Trained models and tokenizers
│   ├── seq2seq/         # Seq2Seq model artifacts
│   └── t5/              # T5 model artifacts
├── eng_to_french.ipynb  # Notebook for Seq2Seq model training
├── transformer-t5.ipynb # Notebook for T5 model training
```

```
├── requirements.txt        # Project dependencies
├── .pre-commit-config.yaml  # Pre-commit hooks configuration
├── .env                    # Environment variables (not shared)
└── .env.example            # Example environment variables
```

Models

The project implements two different translation models:

1. Seq2Seq LSTM Model

- **Architecture**: Encoder-decoder architecture with LSTM layers
- **Training**: Trained on an English-French dataset from Kaggle
- **Features**:
  - Embedding layer (256 dimensions)
  - LSTM encoder and decoder (512 units)
  - Custom inference for handling sequence generation

2. T5 Transformer Model

- **Architecture**: Based on Google's T5 Transformer model

- **Training**: Fine-tuned on the same English-French dataset
- **Features**:
    - Pretrained transformer architecture
    - Beam search decoding
    - More advanced context understanding than the Seq2Seq model

The training process for both models is documented in their respective Jupyter notebooks.

API Design

The application uses FastAPI to provide a robust API:

Endpoints

- **POST /translate**: Main endpoint for translation
    - Takes text input and model choice
    - Returns translated text with metadata
- **GET /health**: Health check endpoint
    - Returns service status information

Request/Response Models

- **TranslationRequest**:
    - text: English text to translate (1-1000 chars)

    - model_type: Model selection (seq2seq or t5)
- **TranslationResponse**:
    - original_text: Input English text
    - translated_text: Output French translation
    - model_used: Which model was used
    - processing_time: Time taken to process

Middleware

The application includes middleware for:

- Logging all requests with timing information
- Consistent error handling

Web Interface

A clean, user-friendly web interface is provided for end users:

- **Features**:
    - Text input area for English text
    - Model selection dropdown (T5 or Seq2Seq)
    - Translation results display
    - Performance metrics (model used, processing time)

- **Design**:
    - Responsive layout
    - Clean, modern styling
    - Loading indicators during translation

Development Environment

The project uses several tools to maintain code quality:

Pre-commit Hooks

- **Black**: Code formatting
- **Ruff**: Linting with automatic fixes
- **MyPy**: Static type checking
- **Other hooks**: Trailing whitespace removal, YAML validation, etc.

Dependencies

Key dependencies include:

- **FastAPI + Uvicorn**: For the API server
- **TensorFlow**: For the Seq2Seq model
- **PyTorch + Transformers**: For the T5 model
- **Pandas + NumPy**: For data processing
- **Pydantic**: For data validation

- **Jinja2**: For templating

Configuration

The project uses a flexible configuration system:

- **Environment Variables**: Via .env file
- **Typed Configuration**: With validation using Python's type system
- **Default Values**: Sensible defaults for all settings

Key configuration options include:

- Model selection (T5 or Seq2Seq)
- Model paths and parameters
- API host and port
- Debug mode toggle

Deployment

The application can be deployed in several ways:

1. **Local Development**:
    - Clone repository
    - Create virtual environment
    - Install dependencies via pip install -r requirements.txt

    - Configure .env file based on .env.example
    - Run with python main.py
2. **Production Deployment**:
    - Can be containerized with Docker
    - Deploy behind a reverse proxy like Nginx
    - Consider using Gunicorn as a production WSGI server

Model Training

The project includes complete training pipelines:

Seq2Seq Training

The eng_to_french.ipynb notebook demonstrates:

- Loading the English-French dataset
- Tokenization and preprocessing
- Model architecture definition
- Training loop implementation
- Model evaluation
- Artifact saving for inference

T5 Training

The transformer-t5.ipynb notebook covers:

- Dataset preparation
- T5 tokenizer and model initialization
- Fine-tuning process
- Evaluation metrics
- Example translations
- Model saving for inference

Performance and Limitations

Performance

- **T5 Model**: Higher accuracy and better handling of complex sentences
- **Seq2Seq Model**: Faster inference but less accurate on complex text

Limitations

- Maximum text length of 128 tokens
- Limited vocabulary based on training dataset
- No support for specialized terminology (technical, medical, etc.)

Future Improvements

Potential enhancements for the project:

1. **Model Improvements**:

- Train on larger, more diverse datasets
- Implement more advanced models (e.g., T5-large)
- Add support for specialized domains

2. **Feature Additions**:
   - Support for additional language pairs
   - Batch translation capabilities
   - Translation memory for frequently translated text
3. **Infrastructure**:
   - Implement caching for common translations
   - Scaling for higher throughput
   - API key authentication for production use
4. **User Experience**:
   - Enhanced web interface with more features
   - Feedback mechanism for improving translations
   - Mobile-optimized interface

Conclusion

This English to French translation project demonstrates a complete machine learning application with both research (Jupyter notebooks) and production components (FastAPI service). The dual-model approach provides flexibility for

different use cases, balancing accuracy and performance needs. The clean architecture and comprehensive development setup make it maintainable and extendable for future enhancements.