Code Repository: All scripts for data preprocessing, model training, optimization, and deployment.

1. Code Repository:

The code repository for this project can be found at the following link: Invoice Data Extraction Using Machine Learning

The repository contains the following scripts:

- `data_preprocessing.py`: This script contains the code for data acquisition and preprocessing. It extracts text from PDF invoices using pdfplumber and pytesseract libraries.
- `model_training.py`: This script contains the code for model architecture, training process, and evaluation metrics. It utilizes a pre-trained BERT model and fine-tunes it on the annotated dataset.
- `model_optimization.py`: This script contains the code for optimization techniques. It converts the trained model to ONNX format and optimizes it using ONNX Runtime.
- `model_deployment.py`: This script contains the code for deployment. It creates a Python script to load the optimized ONNX model and run it for inference.
- `README.md`: This file contains detailed instructions on how to run the code, including dependencies, installation, and usage.

Prerequisites

The following libraries are required to run the code:

- pdfplumber
- pytesseract

- torch
- torchvision
- transformers
- numpy
- pandas
- onnxruntime
- joblib

Usage

To use the invoice data extraction system, follow these steps:

1. Prepare the invoice dataset by converting the invoices to text using the `data_preprocessing.py` script.
2. Train the model using the `model_training.py` script.
3. Optimize the model using the `model_optimization.py` script.
4. Deploy the model using the `model_deployment.py` script.

Project Structure

The project structure is as follows:

- `data_preprocessing.py`: This script contains the code for data acquisition and preprocessing. It extracts text from PDF invoices using pdfplumber and pytesseract libraries.
- `model_training.py`: This script contains the code for model architecture, training process, and evaluation metrics. It utilizes a pre-trained BERT model and fine-tunes it on the annotated dataset.
- `model_optimization.py`: This script contains the code for optimization techniques. It converts the trained model to ONNX format and optimizes it using ONNX Runtime.
- `model_deployment.py`: This script contains the code for deployment. It creates a Python script to load the optimized ONNX model and run it for inference.
- `README.md`: This file contains detailed instructions on how to run the code, including dependencies, installation, and usage.

Evaluation Metrics

The evaluation metrics used to assess the model's performance are:

- Accuracy: The percentage of correctly extracted information.
- Precision: The ratio of correctly predicted instances to all instances predicted.
- Recall: The ability to identify all relevant instances.
- F1-score: A combined measure of precision and recall.

Optimization Techniques

The optimization techniques used to improve the model's performance are:

- Model Conversion: Convert the trained model to ONNX format for efficient deployment.
- Quantization: Optimize the ONNX model using ONNX Runtime to reduce its size and improve performance.

Deployment

The deployment steps involve the following:

1. Install the necessary libraries: onnxruntime, joblib, etc.
2. Create a Python script to load the optimized ONNX model.
3. Use the onnxruntime library to run the model for inference.
4. Provide input invoice text to the model and obtain the extracted information.

Future Improvements

The following improvements can be made to enhance the model's accuracy, performance, and usability:

- Larger Dataset: Training on a more extensive and diverse dataset could improve generalization.
- Advanced NLP Techniques: Exploring more sophisticated NLP methods like Named Entity Recognition (NER) could enhance accuracy.
- Improved OCR: Investigating more robust OCR solutions could improve extraction from scanned documents.
- Web Application: Creating a web-based interface for easy user interaction.
-