Document Automation Architecture

External Email Server

2. Email Processor

Input: Email Text, Attachments

Output: List of Documents
Doc#1: Email text
Doc#2 - #n: Attachements
- Document Details

1.
Document Processor

Input: list of documents
Output:
- Document Type
- Document Details
- Document Status, Exceptions

4.
Auto DataEntry

Input:
- Document Type
- Document Details
- Txs ID (autogen)
Output: Ref Txs ID 1, Ref Txs ID 2

3.
Email Responder

Input: Document Status, Exceptions

Output: An email text for human to review and sent

3rd Party System 1
- Document Type
- Document Details
- Txs ID
Output: Ref Txs ID 1

3rd Party System 2
- Document Type
- Document Details
- Txs ID
Output: Ref Txs ID 2

miro

Document processor

GCP: [Document AI](#) (Classification, Text extraction)

Pro: Pre-trained models, Trainable processor

Con:Manual work

AWS: Textract (Text extraction)

Pro:Ready to use, AWS

Con:Untrainable

Document Processor

AWS: [Rekognition](Classification)

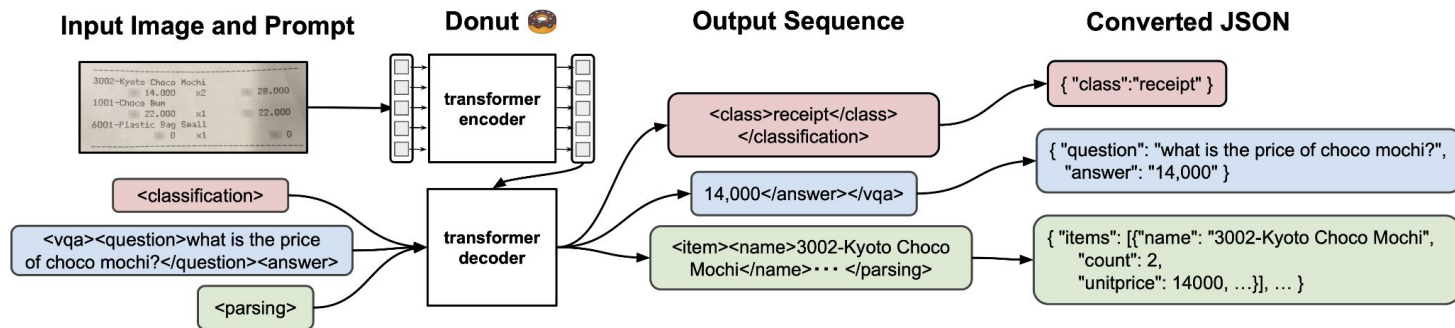Pro: AWS

Con: relatively expensive hosting fee

On-going:

AWS: Comprehend(Classification)

Azure: Document Intelligence(Classification, Text Extraction)

Document Processor

Alternative:

Document understanding transformer(DONUT)

Pro: Open source

Con: Time-consuming

Caution

1. Lack of data in amount and variety
2. Pay attention to pricing

Email processor/responder

Large Language Model

Framework: [Langchain](#)

Details:

1. Prompt Engineering
2. Deployment
   a. Might be slow on local
3. Retrieve Augmented Generation
   a. How to generate content based on a specific customer

Email Processor/Responder

Vector database

   Milvus, Weaviate, Pinecone, Elastic search…