Attendance

Code: https://code.bizgaze.com/SadhulaSaiKumar/AI/src/branch/master/Attendence

To run the code
1) create an environment and run below command
   - pip install -r requirements.txt
   - python attendence.py

For test the app using postman
Url is http://115.248.56.9:5003/register & http://115.248.56.9:5003/detect
In body select raw and type is json then past he json text present in input_attendence_data.json

Aim:Develop an attendance app to streamline tracking and management, offering a user-friendly interface for efficient attendance recording and real-time monitoring

Technologies:
- Programming Languages: Python
- Web Development: HTML, CSS
- Version Control: Git
- Frameworks: Django, Flask
- Libraries: OpenCv,Tensorflow, Keras, Pytorch,face-recognotion,Dlib,cmake.
- Deployment: UWSGI and NGINX server.
- Operating Systems: Linux, Windows

Impact:Attendance app enhances organizational efficiency by automating attendance tracking, reducing errors, and providing instant insights, fostering better resource utilization and productivity. Additionally, it promotes accountability and punctuality, creating a culture of responsibility among individuals.

Events

Code: https://code.bizgaze.com/SadhulaSaiKumar/AI/src/branch/master/Events/src

To run the code
1) create an environment and run below command
   - pip install -r requirements.txt

- `python src/myproject2.py`

For test the app using postman
Url is http://115.248.56.9:8081/predict

Body : `{"People": "/opt/bizgaze/events.bizgaze.app/wwwroot/_files/People/", "Gallery": "/opt/bizgaze/events.bizgaze.app/wwwroot/_files/1/Gallery/"}`

Aim:Implemented an Automatic Face Detection system, streamlining image distribution based on facial features

- Installing dlib for CPU and dlib with cuda for gpu support   (Depends on the type of server being used)
- Used haar cascade model of opencv for separating only face structure from whole dataset of Images
- Clustering code in python to cluster similar faces and taking unique(single image) all in one folder (People folder)
- People folder (Test images) compared using face recognition library (dlib) the main (myproject.py) file, for final output
- Above steps will separate the face according to the unique folders for separate unique images
- Finally the code is deployed in Flask framework
- The api calls tested in postman and forwarded to the .net team to get the output as json file
- The accuracy was improved by setting up the tolerance ratio in the code mostly ranging from 0.4 to 0.6

- The idea here is to encode the dataset from people's folder and then convert them to numpy arrays
- These arrays are then compared with the images in the Dataset folder
- The compared images are then separated according to their respective arrays and the array near to the original image is clustered in the same folder
- The library has two main pretrained models like cnn can be used if running the application with gpu support
- For using with gpu CUDA has to be installed along with dlib cuda version and NVIDIA drivers
- The library outperforms all the other available libraries on the internet and gives a very good accuracy of about 85-90 percent
- Following is the link for the library for reference

Technologies:

- Programming Languages: Python

- Web Development: HTML, CSS
- Version Control: Git
- Frameworks: Django, Flask
- Libraries: OpenCv,Tensorflow, Keras, Pytorch,face-recognotion,Dlib,cmake.
- Deployment: UWSGI and NGINX server.
- Operating Systems: Linux, Windows

Impact: Implementing an Automatic Face Detection system expedites image distribution by automatically identifying and categorizing faces, improving content organization and accessibility. This technology-driven solution enhances efficiency in image management, showcasing the practical application of facial recognition in optimizing workflows.

Demand Forecasting

Code: https://code.bizgaze.com/SadhulaSaiKumar/AI/src/branch/master/Demand_forcasting
To run the code
1) create an environment and run below command

- pip install -r requirements.txt
- python forcasting2.py

For test the app using postman
Url is http://115.248.56.9:8082/sales_forcast
Body:

```
{
  "wise": "day/month",
  "future_dates": 5,
  "get_url": "http",
  "get_url_token": "stat 558154141",
  "post_url": "http",
  "post_url_token": "stat 525963525",
}
```

Aim:To prevent stockouts and overstock situations, improving overall operational efficiency and customer satisfaction.

Technologies:

- Programming Languages: Python
- Web Development: HTML, CSS
- Version Control: Git
- Frameworks: Django, Flask
- Libraries: Scikit-learn,Pandas,Numpy,Machine Learning,Deep Learning
- Deployment: UWSGI and NGINX server.
- Operating Systems: Linux, Windows

Impact:A demand forecasting app enhances supply chain efficiency by accurately predicting product demand, minimizing overstock and stockouts. This application optimizes inventory management, reducing costs and improving overall business resilience.

Business Card

Code: https://code.bizgaze.com/SadhulaSaiKumar/AI/src/branch/master/Business_cards
To run the code
1) create an environment and run below command

- pip install -r requirement.txt
- python Business_cards.py

For test the app using postman
Url is http://192.168.88.69:1112/upload_BusinessCards
In body select raw and type is json then past he json text present in input_business card_data.json

Aim: Automatic parsing/filling of Business card details into csv file using image or Pdfs
Technologies:

- Programming Languages: Python
- Web Development: HTML, CSS
- Version Control: Git
- Frameworks: Django, Flask
- Libraries: OpenCv, Spacy, NLTK, Fitz, Pdfminer, Plumber, Paddleocr, Tesseract-ocr,Tensorflow, Keras, Pytorch
- Deployment: UWSGI and NGINX server.
- Operating Systems: Linux, Windows

Impact:

1. Time Savings: Business card parsers automate data extraction, significantly reducing the time spent manually entering contact information.

2. Error Reduction: By minimizing manual entry, these tools enhance data accuracy, mitigating the risk of typos and inaccuracies in business contact details.
3. Efficient Networking: Business card parsers streamline the digitization of contacts from networking events, facilitating prompt and organized integration into databases for improved relationship management.

Invoice Parsing

Code: https://code.bizgaze.com/SadhulaSaiKumar/AI/src/branch/master/Invoice_parser

To run the code
1) create an environment and run below command

- pip install -r requirementsinvoice.txt
- python invoice.parser.py

For test the app using postman
Url is http://192.168.88.69:1112/upload_invoice
In body select raw and type is json then past he json text present in input_invoice_data.json

Aim:Automatic parsing/filling of invoice details into csv file using image,Pdf,Doc files.

Step1: Data Collection

Collected around 1000 INVOICES combining pdf and IMAGE files both from internet and other sources.

Step2: Data Preparation

- Extracted text from all the collected documents and created a text file of all
- Python Libraries used here are pdftotext, pdfminer, doctotext, etc.
- Manual Annotation for all the Keywords such as Name, Address, Education, Experience, DOB, etc. using Spacy NER annotation tool. The Keyword should be the entity and the value should be the answer of that entity
- The Team spent more time for manual annotations combining around 1000 invoices after cleaning the data
- Converted the text file into .pkl format using python as Spacy uses the data in .pkl to be trained
- Data augmentation was done on 200 files and hence the dataset was increased to more than 1000 files

Step3: Regex and Table extraction

- Regular expressions and python were used to extract Addresses in an invoice.
- The challenge here was for dynamic addresses and similar format of text extraction from fitz and other libraries, hence the solution for Regex was used here
- Table extraction was another challenge here so we implemented multiple libraries like excalibur, opencv, camelot, etc.

Step4: Training and Testing

- Trained an NLP basic english model using spacy with 350 epochs on the above collected data.
- The trained model is around 5-10 mb and totally built from scratch using google collab for a better speed with good amount of GPU
- The model is then saved and loaded again and tested with new unseen resumes
- For testing mostly, we used libraries like Fitz and pdfminer

Step5: Evaluation and Deployment

- Deployed the above models into Flask application using Flask API.
- Here we can upload a doc/pdf file and directly download the csv file of parsed resume
- The Flask application is deployed using UWSGI and/or NGINX server.

Technologies:

- Programming Languages: Python
- Web Development: HTML, CSS
- Version Control: Git
- Frameworks: Django, Flask
- Libraries: OpenCv, Spacy, NLTK, Fitz, Pdfminer, Plumber, Paddleocr, Tesseract-ocr,Tensorflow, Keras, Pytorch
- Deployment: UWSGI and NGINX server.
- Operating Systems: Linux, Windows

Impact: Invoice Parsing app accelerates data extraction from invoices, minimizing human intervention and enhancing accuracy, thereby optimizing financial data processing and bolstering data-driven decision-making. It showcases the practical application of advanced data science techniques for efficient automation in business processes.

Resume Parser

Code: https://code.bizgaze.com/SadhulaSaiKumar/AI/src/branch/master/Resume_parser
To run the code
1) create an environment and run below command

- pip install -r requirementsresume.txt
- python resume.parser.py

For test the app using postman
Url is http://192.168.88.69:1112/upload_resume
In body select raw and type is json then past he json text present in input_resume_data.json

Aim:Automatic parsing/filling of Resume details into csv file using PDF/Doc files.

Step1: Data Collection

Collected around 600 resumes combining of pdf and word files both from internet, HR, and other sources

Step2: Data Preparation

- Extracted text from all the collected documents and created a text file of all
- Python Libraries used here are pdftotext, pdfminer, doctotext, etc.
- Manual Annotation for all the Keywords such as Name, Address, Education, Experience, DOB, etc. using Spacy NER annotation tool. The Keyword should be the entity and the value should be the answer of that entity
- The Team spent more time for manual annotations combining around 555 Resumes after cleaning the data
- Converted the text file into .pkl format using python as Spacy uses the data in .pkl to be trained

Step3: Training and Testing

- Trained an NLP basic english model using spacy with 225 epochs on the above collected data.
- The trained model is around 5-7 mb and totally built from scratch using google collab for a better speed with good amount of GPU
- The model is then saved and loaded again and tested with new unseen resumes
- For testing mostly, we used libraries like Fitz and pdfminer

Step4: Evaluation and Deployment

- Deployed the above models into Flask application using Flask API.
- Here we can upload a doc/pdf file and directly download the csv file of parsed resume
- The Flask application is deployed using UWSGI and/or NGINX server.

SupportGPT

code:https://code.bizgaze.com/SadhulaSaiKumar/AI/src/branch/master/Supportgpt
To run the code
1) create an environment and run below command

- pip install -r requirements.txt
- python ingest.py
- python supportgpt_mistral.py

Aim:To provide a responsive and automated communication channel that can efficiently interact with users, answer queries, and perform tasks, thereby enhancing user engagement and satisfaction while streamlining customer support processes.

Technologies:

- Programming Languages: Python
- Web Development: HTML, CSS
- Version Control: Git
- Frameworks: Django, Flask,Langchain
- Libraries: Scikit-learn,Pandas,Numpy,Machine Learning,Deep Learning,langchain
- Deployment: UWSGI and NGINX server.
- Operating Systems: Linux, Windows

Impact:It improves customer service efficiency by offering instant responses to queries, reducing response times, and enhancing user experience. It also enables businesses to scale customer interactions, freeing up human resources for more complex tasks, ultimately leading to increased operational efficiency.
Reference : https://www.youtube.com/watch?v=AjQPRomyd-k&ab_channel=codebasics

To run the code on VM

Open winscp and login with below credentials

Ip	: 115.248.56.9
Username	: root
Pwd	: Qazplm@4#2
Port	**:** 220

Navigate to /home/ubuntu/AI

For running attendance code
Open putty run below line

- cd /home/ubuntu/AI/ATTENDANCE

To attendance app run the below command

- python runapp.py

To run demand forecasting app run below line

- cd /home/ubuntu/AI/forcasting
- python forecasting.py

To run Event app run below line

- cd /home/ubuntu/AI/Events
- python myproject.py

For running Resume,Invoice , Business card from ubuntu system
Go to project folder in desktop and run finalapp.py

For running supportgpt from ubuntu system
Go to supportgpt folder in desktop and run supportgpt.py

To run any application in background in vm run below command
nohup python filename.py &