

Intelligent Travel Solutions: Merging User Preferences With Real – Time Contextual Awareness

MSc Research Project
Master of Science in AI

Shalini Priya
23214333

School of Computing
National College of Ireland

Supervisor: SM Raza Abidi

National College of Ireland
MSc Project Submission Sheet
School of Computing



Student

Name: Shalini
Priya

Student ID: 23214333

Programme:
AI

Programme:

Module:

Supervisor:

SM Raza

Abidi

Submission

Due Date:

Project Title: Intelligent Travel Solutions: Merging User Preferences with Real –
Time Contextual Awareness

Student

Shalini Priya
Name:

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Shalini Priya

Signature:

Date:29/01/2025.....

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	<input type="checkbox"/>
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only

Signature:	Shalini Priya
Date:	29/01/2025
Penalty Applied (if applicable):	

Intelligent Travel Solutions: Merging User Preferences With Real – Time Contextual Awareness

Shalini Priya

x23214333@student.ncirl.ie

1. Introduction

This configuration manual offers a comprehensive guide for setting up and implementing the research project titled "Intelligent Travel Solutions: Merging User Preferences with Real-Time Contextual Awareness." The project aims to recommend the most reliable itineraries by integrating user preferences, past experiences, and current trends with real-time contextual factors, including climatic conditions. The system adapts travel recommendations based on efficient algorithms and data-driven insights, increasing customer satisfaction and reliability. This manual describes the processes required to configure the system, assuring smooth execution and excellent performance in order to meet the project's objectives efficiently.

2. System Specification

To ensure the successful execution of the code, your system should meet the following minimum specifications:

- **Operating System:** Windows 10 or higher, macOS 10.15 or higher, or a Linux distribution (e.g., Ubuntu 20.04).
- **Processor:** Intel Core i7 or equivalent AMD processor (Quad-Core or higher).
- **Memory:** 8 GB RAM/16 GB RAM
- **Storage:** At least 100 GB free SSD space.
- **GPU:** NVIDIA GPU with CUDA support (e.g., NVIDIA GeForce GTX 1060 or higher) with at least 6 GB of VRAM.
- **Python Version:** Python 3.7 or higher.

3. Softwares Used

The following software and libraries are used in this project:

3.1 Programming Language

- **Python:** The primary programming language used for implementing the model and processing data.

3.2 Python Libraries

- **NumPy:** For numerical operations.
- **Pandas:** For data manipulation and analysis.
- **Matplotlib & Seaborn:** For creating static and interactive visualizations of the data.
- **Scikit-learn:** For machine learning models and evaluation metrics.
- **Citipy:** For determining city coordinates based on latitude and longitude.
- **Requests:** For making HTTP requests to external APIs (e.g., weather, Google Maps).
- **Datetime:** For handling date and time functionalities.
- **Time:** For managing time delays in API requests.

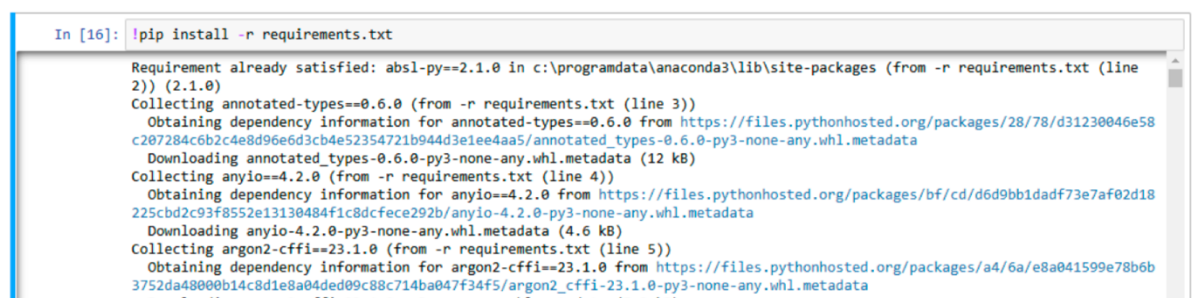
- SciPy: For statistical operations such as linear regression.
- Gmaps: To integrate Google Maps for visualizing locations.
- Scikit-learn: For model evaluation metrics like confusion matrices.
- PIL (Pillow): For image enhancement and manipulation.
- Geopy: For calculating distances using geographical coordinates (e.g., `great_circle` for distance calculation).
- Shapely: For geometric operations and manipulating geographic data (e.g., `MultiPoint`).
- DBSCAN: For clustering locations based on spatial proximity.
- `train_test_split`: For splitting data into training and testing sets.
- Cosine similarity: For measuring similarity between vectors, especially for user preferences.
- pairwise distances: For calculating pairwise distance metrics.
- `mean_squared_error`: For evaluating the performance of models.
- minmax scale: For normalizing data into a specific range.
- MultiLabelBinarizer: For converting multi-label data into binary format.
- NMF (Non-Negative Matrix Factorization): For dimensionality reduction and extracting features.
- Random: For generating random values (e.g., selecting random cities or locations).
- `surprise.SVD`: Singular Value Decomposition algorithm for collaborative filtering in recommendation systems.
- `surprise.Reader`: For loading datasets in a format compatible with the Surprise library.
- `surprise.Dataset`: For managing and pre-processing recommendation datasets.
- `surprise.model_selection.cross_validate`: For evaluating models using cross-validation techniques.

3.3 Development Environment

- Jupyter Notebook or Visual Studio Code: For running and editing the Python scripts.
- Jupyter Notebook 6 or above (Notebook Environment package details added in the Code Artifacts)

• Library installation:

A `requirements.txt` file has been shared with the Code Artifacts. We can use `!pip install -r requirements.txt` command within jupyter notebook. The output of installation looks similar to the following image. Ref 1



```

In [16]: !pip install -r requirements.txt

Requirement already satisfied: absl-py==2.1.0 in c:\programdata\anaconda3\lib\site-packages (from -r requirements.txt (line 2)) (2.1.0)
Collecting annotated-types==0.6.0 (from -r requirements.txt (line 3))
  Obtaining dependency information for annotated-types==0.6.0 from https://files.pythonhosted.org/packages/28/78/d31230046e58c207284c6b2c4e8d96e6d3cb4e52354721b944d3e1ee4aa5/annotated_types-0.6.0-py3-none-any.whl.metadata
  Downloading annotated_types-0.6.0-py3-none-any.whl.metadata (12 kB)
Collecting anyio==4.2.0 (from -r requirements.txt (line 4))
  Obtaining dependency information for anyio==4.2.0 from https://files.pythonhosted.org/packages/bf/cd/d6d9bb1dadf73e7af02d18225cbd2c93f8552e13130484f1c8dcfece292b/anyio-4.2.0-py3-none-any.whl.metadata
  Downloading anyio-4.2.0-py3-none-any.whl.metadata (4.6 kB)
Collecting argon2-cffi==23.1.0 (from -r requirements.txt (line 5))
  Obtaining dependency information for argon2-cffi==23.1.0 from https://files.pythonhosted.org/packages/a4/6a/e8a041599e78b6b3752da4800b14c8d1e8a04ded09c88c714ba047f34f5/argon2_cffi-23.1.0-py3-none-any.whl.metadata
  Downloading argon2_cffi-23.1.0-py3-none-any.whl.metadata (5.7 kB)

```

Figure 1: Installing requirements.txt into conda/jupyter/pycharm environment

Config file :-

A custom configuration file, **config1**, has been created for this project, containing essential API keys such as **g_key** and **weather_api_key**.

These keys are utilized for accessing required services.

The configuration file is seamlessly imported into the project using the command:

```
import config1
```

3.4 Optional Tools

- Anaconda: For managing Python environments and dependencies.
- CUDA Toolkit: If using an NVIDIA GPU, ensure that CUDA and cuDNN are installed to leverage GPU acceleration.

4. Dataset Source

The DataSet used for this study

- Yelp Academic Dataset – Kaggle
Link :- <https://www.kaggle.com/datasets/abidmeeraaj/yelp-labelled-dataset>
- Flickr data - Kaggle
- Weather data – Openweather API data.

5. Environment Setup/Mandatory steps to Project setup

This project integrates three datasets, each interconnected to enhance the overall analysis and modeling. To ensure efficiency and streamline execution, I employed a combination of **Bash scripting** and **Jupyter Notebook** workflows:

- 1. Data Integration and Preprocessing:**
 - Bash scripts were used to automate the integration and preprocessing tasks, enabling seamless handling of the datasets and reducing manual overhead.
- 2. Data Visualization and Exploration:**
 - Jupyter Notebooks were utilized for interactive data visualization and exploratory data analysis (EDA). The notebook environment provided flexibility to experiment with various plots and better understand the relationships within the data.
- 3. Model Development and Deployment:**
 - The final machine learning model was implemented in a Python script (.py file) to facilitate efficient execution in a production-like environment (e.g., Coda or other runtime environments). This separation ensures reproducibility and scalability for final deployment.
 - For Final weather Integration the final file is running on jupyter.

By structuring the workflow in this way, I achieved both efficiency in execution and clarity in analysis and modelling.

6. Execution of the Code Implementation

Step 1: Setting Up the Environment

A. Install Anaconda (optional): If you prefer managing environments with Anaconda, install it from [Anaconda's official website] (<https://www.anaconda.com/>).

B. Run the file in Google colab for better Experience

C. Install Required Libraries:

```
! pip install numpy pandas tensorflow keras opencv-python matplotlib seaborn plotly scikit-learn pillow
```

Python

Run The flickr data

- Since we Only London data so we are keeping for London while Yelp data is processed on the UK And US data.
- and then run DBSCAN cluster method.

Then Process the Yelp data

To Run the Yelp data

```
cd code && python train.py
```

Folder structure of this data set

```
.
├── Data Processing and Analysis
│   ├── DA_Cleaning_Business.ipynb
│   ├── DA_Review.ipynb
│   ├── example_plot
│   ├── loss.png
│   ├── mkyelp2018.py
│   ├── ndcg-recall.png
│   ├── nohup.out
│   ├── plot.ipynb
│   └── yelp-json-EDA.ipynb
├── Procedure.py
├── __pycache__
│   ├── Procedure.cpython-311.pyc
│   ├── dataloader.cpython-311.pyc
│   ├── filter.cpython-311.pyc
│   ├── idIndex.cpython-311.pyc
│   ├── model.cpython-311.pyc
│   ├── parse.cpython-311.pyc
│   ├── rcmdweb.cpython-311.pyc
│   ├── recommendation.cpython-311.pyc
│   ├── register.cpython-311.pyc
│   ├── utils.cpython-311.pyc
│   └── world.cpython-311.pyc
├── checkpoints
└── lgn-yelp2018-3-64.pth.tar
```

```
├── dataloader.py
├── filter.py
├── idIndex.py
├── model.py
├── parse.py
├── recommendation.py
├── register.py
├── train.py
├── utils.py
└── world.py
```

To integrate Yelp and Flickr data with weather data, execute the following:

1. Run the Final file.
 2. Execute the Combine file.
 3. Process the After_weather Integration notebook.
- After each step one CSV will be created and saved for next process.

Step 3: Running the Code

A. Open Jupyter Notebook or VS Code in the project directory.

B. Run the Code:

For Yelp data :-

- We recommend not using the matrix multiplication code for splitting the user-item matrix, as it can significantly slow down the training process.
- If you encounter slow test processing, try increasing the testbatch value and enabling the multicore option (note that Windows systems may experience issues with this option enabled).
- We suggest utilizing the tensorboard option, which is a useful tool for monitoring the training process.
- To run lightGCN on your own dataset, modify the dataloader.py file by implementing a custom dataset class inherited from BasicDataset. Then, register your dataset in register.py.
- To run your own models on the provided datasets, modify the model.py file by implementing a custom model class inherited from BasicModel. Then, register your model in register.py.
- To implement custom sampling methods on the provided datasets and models, modify the Procedure.py file by implementing a new function. Finally, update the corresponding code in main.py.

Additional Tip

- Since we set fixed seeds for numpy and torch using --seed=2020, running the command with this option will result in identical output logs, regardless of the execution time. Specifically, you can verify this by comparing the output at epochs 5 and 116.


```

18 Recmodel = Recmodel.to(world.device)
19 bpr = utils.BPRLoss(Recmodel, world.config)
20
21 weight_file = utils.getFileName()
22 print(f"load and save to {weight_file}")
23 if world.LOAD:
24     try:
25         Recmodel.load_state_dict(torch.load(weight_file, map_location=torch.device('cpu')))
26         world.cprint(f"loaded model weights from {weight_file}")
27     except FileNotFoundError:
28         print(f"{weight_file} not exists, start from beginning")
29 Neg_k = 1
30 world.TRAIN_epochs=100
31
32 # init tensorboard
33 if world.tensorboard:
34     w : SummaryWriter = SummaryWriter(
35         join(world.BOARD_PATH, time.strftime("%m-%d-%Hh%Mm%Ss-") + "-" + world.comment)
36     )
37 else:
38     w = None
39     world.cprint("not enable tensorflowboard")
40
41 try:
42     for epoch in range(world.TRAIN_epochs):
43         start = time.time()
44         if epoch % 10 == 0:
45             cprint("[TEST]")
46             Procedure.Test(dataset, Recmodel, epoch, w, world.config['multicore'])

```

```

36 •[0;30;43m[TEST]•[0m
37 {'precision': array([1.06040178e-05]), 'recall': array([0.047507]), 'ndcg': array([5.67807516e-04])}
38 EPOCH[1/100] loss0.543-[Sample:8.30]
39 EPOCH[2/100] loss0.290-[Sample:8.13]
40 EPOCH[3/100] loss0.219-[Sample:8.13]
41 EPOCH[4/100] loss0.181-[Sample:7.97]
42 EPOCH[5/100] loss0.156-[Sample:8.26]
43 EPOCH[6/100] loss0.135-[Sample:8.20]
44 EPOCH[7/100] loss0.118-[Sample:8.11]
45 EPOCH[8/100] loss0.106-[Sample:8.06]
46 EPOCH[9/100] loss0.095-[Sample:7.96]
47 EPOCH[10/100] loss0.086-[Sample:7.96]
48 •[0;30;43m[TEST]•[0m
49 {'precision': array([0.004217]), 'recall': array([0.0596639]), 'ndcg': array([0.00335421])}
50 EPOCH[11/100] loss0.078-[Sample:7.80]
51 EPOCH[12/100] loss0.071-[Sample:8.13]
52 EPOCH[13/100] loss0.065-[Sample:8.05]
53 EPOCH[14/100] loss0.060-[Sample:7.88]
54 EPOCH[15/100] loss0.056-[Sample:8.06]
55 EPOCH[16/100] loss0.051-[Sample:7.93]
56 EPOCH[17/100] loss0.048-[Sample:7.91]
57 EPOCH[18/100] loss0.045-[Sample:7.87]
58 EPOCH[19/100] loss0.042-[Sample:8.09]
59 EPOCH[20/100] loss0.040-[Sample:8.04]
60 •[0;30;43m[TEST]•[0m
61 {'precision': array([0.0047706]), 'recall': array([0.0613784]), 'ndcg': array([0.0373768])}
62 EPOCH[21/100] loss0.037-[Sample:7.86]
63 EPOCH[22/100] loss0.035-[Sample:8.08]
64 EPOCH[23/100] loss0.034-[Sample:7.95]
65 EPOCH[24/100] loss0.032-[Sample:7.89]
66 EPOCH[25/100] loss0.030-[Sample:8.10]
67 EPOCH[26/100] loss0.029-[Sample:8.00]
68 EPOCH[27/100] loss0.027-[Sample:7.94]
69 EPOCH[28/100] loss0.026-[Sample:8.01]

```

- **Integration of weather data**

Need to import config1

Get the dummy API_KEY for gmaps and weather api key for this project I generated my API keys.

```

In [195]: from config1 import weather_api_key, g_key
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from citipy import citipy
import requests
from datetime import datetime
import time
from scipy.stats import linregress
import gmaps

In [196]: import config1
print(config1.__file__)

/Users/shalini/Flickr/config1.py

In [197]: gmmaps.configure(api_key="YOUR_API_KEY")

In [198]: weather_api_key = "Your_API_KEY"

In [199]: lats = np.random.uniform(-90, 90, size=1500)
lngs = np.random.uniform(-180, 180, size=1500)

coordinates = zip(lats, lngs)
cities = []

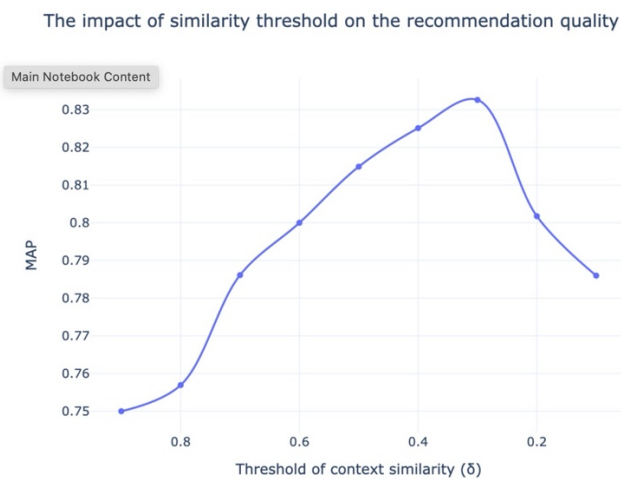
```

Later integrated with both datas and
Get the Final Recommendation from After Weather Integration notebook

Step 4: Evaluating the Model

- After training, assess the model's accuracy using the validation and test datasets.
- we evaluate the proposed method based on common evaluation metrics in recommendation systems, MAP and RMSE.

Final Evaluation graph



Step 5: Making Predictions

- Make the predictions for each users w.r.t their likeliness of location
- Here for one user data

	true	pred
location_id		
140662	0.000000	2.935065
165202	0.000000	0.000000
185565	1.444444	1.198561
220206	0.000000	2.282828
299823	0.000000	1.556474
330684	0.000000	1.956710
335960	0.000000	1.369697
422774	0.000000	0.000000
505088	5.000000	4.583333
519537	0.000000	2.092593
529367	0.000000	0.684848
594032	0.000000	0.000000

References

Python: <https://www.python.org>

For RefLightGCN :- <https://github.com/kuandeng/LightGCN>)