Configuration Manual

Akanksh Reddy Muddam

Student ID: X23228482

1. Introduction

The increasing demand for proper nutrition recommended to individuals with unique requirements has spurred the establishment of systems that honour particular nutrition. This project is centred on the development of a Personalized Meal Recommendation System responsible for recommending the most appropriate meals to a specific user given his/her; restrictions; and the machine learning and artificial intelligence involved.

The system blends both content and Collaborative Filters to enhance recommendations and observe user satisfaction. The real-time user interaction and feedback are integrated to enhance the quality of the recommendations. In this manual, it is described which settings need to be set up in order to reproduce the experiments, and information about hardware, SW, data sets and procedures is given.

2. System Requirements

2.1 Hardware

- **Processor:** Minimum of quad-core 3.0 GHz or better if data processing and models are to be developed at a renowned pace.
- **RAM:** Minimum of 16 GB to support large data sets or calculations and operations.
- **GPU:** For the faster deep learning computations NVIDIA CUDA-enabled GPU such as RTX 3060 and above is recommended.
- **Storage:** That is 50 GB of free space for datasets and model artefacts at the very least.

2.2 Software

- **Operating System:** Works with Linux (Ubuntu 20.04 for instance), Windows 10, or the Macintosh OS system.
- **Development Environment:** Visual Studio Code or PyCharm for the code written part. Jupyter Notebook for the testing and prototyping of the work.
- **Programming Language:** Python 3.9 or later because of its best compatibility with most ML/AI-related libraries.
- **Python Libraries:** Key libraries include:
  - TensorFlow for deep learning model training and evaluation.
  - Scikit-learn for machine learning algorithms and preprocessing.
  - **Additional libraries:** numpy, pandas, flask, pymongo, SQLalchemy, BeautifulSoup, selenium.
- **Database Tools:** For the structured data, we use PostgreSQL. MongoDB for unstructured and semi-structured data.
- **Deployment Tools:** The programming for the containerized deployment: Docker for containers. AWS EC2 for scalable hosting.

3. Dataset Configuration

This project involves parsing user-related information and nutrition facts in order to build artificial intelligence recommendation systems for meals. Much caution should be exercised in order to manage and clean these datasets for the purpose of achieving accurate outcomes.

3.1 Sources

Structured Data: The data extracted is nutritional and most of them are obtained from standard databases like the USDA FoodData Central. The data is in CSV files (e.g., nutrition_data.csv) that contain the names of the ingredients, the energy value, macronutrients, and dietary labels (e.g., vegan, gluten-free). **User**

Interaction Data: Data generated by users from synthetic or actual Usage logs such as historical meal menus, ratings or feedback. These are stored in JSON format in MongoDB collections; the format used by MongoDB for data storage.

3.2 Preprocessing

Handling Anomalies: Poor or missing values such as "-9999" in fields that pertain to caloric or macronutrient input should be replaced with the median value in that particular column.

Normalization: Make all quantitative parameters like Energy, Protein, fat, carbohydrate and so on at a similar range for all the variables between 0 and 1.

Class Balancing: For structured data, use the SMOTE for handling the issues related to the dietary tag category imbalances For example, there could be fewer low-carb meals in your training dataset. For user interaction data, avoid skewed distribution of meal type categories to mean that you over-sample rare meal preferences.

Feature Engineering: Derived features should include nutrient density scores and meal type and user's restrictions on the type of foods they can eat.

4. Code Configuration

4.1 Setup

1. **Environment Setup**:

```
python -m venv mealrec_env
source mealrec_env/bin/activate
pip install -r requirements.txt
```

2. **Database Configuration**:

   - **PostgreSQL**: Create a database named meal_recommendation and import the schema:

     ```
     psql -U postgres -d meal_recommendation -f schema.sql
     ```

   - **MongoDB**: Import user interaction data:

```
mongoimport --db user_logs --collection interactions --file interactions.json
```

4.2 Preprocessing

- **Structured Data**:
  - Replace outliers using the Interquartile Range (IQR) method.
  - Encode categorical variables with LabelEncoder.
- **User Profiles**:
  - Add demographic and health data to initialize user profiles for cold-start scenarios.

---

5. Model Training and Evaluation

5.1 Training

1. **Content-Based Filtering**:
   - Use cosine similarity to calculate meal recommendations based on nutrient composition.
   - Train a KNN model to rank meals.
2. **Collaborative Filtering**:
   - Train a matrix factorization model using Singular Value Decomposition (SVD).
   - Use TensorFlow to implement a neural collaborative filtering model if additional accuracy is needed.

5.2 Deployment

1. **Backend API**:
   - Host using Flask:

```
flask run --host=0.0.0.0 --port=5000
```

2. **Frontend Interface**:
   - Implement a responsive UI using React.js.
   - Connect the backend API to display recommendations dynamically.
3. **Cloud Hosting**:
   - Deploy the system on AWS EC2 and enable auto-scaling for traffic management.

6. Outputs, Artifacts, Troubleshooting and Recommendations

6.1 Models

- Content-Based Model: Saved as content_model.pkl.
- Collaborative Filtering Model: These are saved as collaborative_model.h5 files.
- Hybrid Model: Improves the_index_of_the_above_two_and_saves_it_as_hybrid_model.pkl.

6.2 Deployment Files

- app.py: Flask API script.
- frontend/: Directory that contains the codebase of the React.js frontend of the elastic inferred search application.
- Dockerfile: Setting up for the application to be containerised.

6.3 Common Issues Environment Errors:

- The dependence should be installed in the right environment of the python. To check the installed packages pip freeze > requirements.txt.
- Database Connection Errors: Make sure PostgreSQL and MongoDB services are running. Make certain that .env includes valid connection strings.

6.4 Recommendations Model Explainability:

- For model reports, use SHAP or L Lime to get explanations of predictions made by the model.
- Scalability: Organize data using Kafka or AWS Kinesis for the change data capture mechanism of the Real-time Data Pipeline.
- Future Enhancements: The enhancements include the ability to blend wearable device data into its overall health assessment. Rewear deep reinforcement learning for an adaptive recommendation of meals.

7. References

1. Burke, R. (2007). Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction.*
2. Ricci, F., Rokach, L., & Shapira, B. (2015). Recommender Systems Handbook. *Springer.*
3. Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. *IEEE Computer.*
4. Zhang, Z., et al. (2017). Data Cleaning and Preprocessing for Machine Learning. *Journal of Big Data.*
5. Nguyen, P., et al. (2019). Nutritional Recommendation Systems: Trends and Challenges. *Journal of Nutrition and Dietetics.*
6. Armbrust, M., et al. (2010). A View of Cloud Computing. *Communications of the ACM.*
7. He, X., et al. (2017). Neural Collaborative Filtering. *Proceedings of the 26th International Conference on World Wide Web.*
8. Lops, P., et al. (2011). Content-Based Recommender Systems: State of the Art and Trends. *Recommender Systems Handbook.*
9. Brooke, J. (1996). SUS: A Quick and Dirty Usability Scale. *Usability Evaluation in Industry.*
10. Agrawal, R., et al. (2020). Applications of Machine Learning in Personalized Recommendations. *International Journal of Data Science and Analysis.*