**Data map:**

We can use different sources of information to prepare a high quality dataset for this tourism place recommender system. The information can be acquired from the following sources:

* • Demographic features (Age, gender, nationality etc.)
* • Social information (liked news videos, subscribed channels, comments, ratings)
* • Internet of things information (GPS location etc.)
* • Content information (Categories, features)

Content can be stored in a distributed storage device, google cloud storage buckets or Amazon’s S3 storage while content related information i.e. tags can be stored in a database in the form of tables. Demographic, Social and IoT data can also be stored in the form of tables in a database.

**Algorithm details:**

State-of-the-art recommender system algorithms try to balance factors like accuracy, novelty, dispersity and stability in the recommendations. Our algorithms would be a combination of 4 major sub-categories of algorithms:

**1. Collaborative Filtering:** Collaborative filtering algorithms tend to filter items that a user might like on the basis of preferences of similar users. K-nearest neighbour is the most popular collaborative filtering algorithm that has been widely used to build recommendation systems. In the first step k similar users are selected by using a similarity measure e.g. cosine similarity, jaccard similarity, mean squared error, pearson similarity etc. This step is followed by an aggregation function e.g. mean, median e.t.c. which ranks similar items and suggests them to users. There have been several advances in CF algorithms over the past years that improve computation time and accuracy of the system using learning methods, margin based similarity measures, temporal dynamics and several factorization techniques.

**2. Content based filtering:** Machine learning has played a great role to understand and categorize unstructured complex content e.g. speech, images, videos etc. This information is used to develop content similarity and set item preferences for users.

**3. Demographic filtering:** Humans live in the form of groups, cities, states and countries. Each group has similar social norms, behaviours and preferences. This principle holds with other personal attributes too (sex, age etc.). Demographic filtering utilizes this principle to set preferences for user groups.

**4. Model-based filtering:**Model based methods try to create a model of underlying information to suggest items. There has been a lot of work done on model based methods in previous years. These improvements include several types of loss function, margin-based approaches, attention based learning and problem formulation in different ways.

**Testing :**

Evaluation of recommender systems is a really important topic ever since the research started in this field. Mean absolute error, root mean squared error, precision, recall, coverage and ROC are commonly used evaluation metrics on the set of known ratings of users on items. Stability, novelty, reliability and diversity measures are also important to test prior the system is launched.

**Post-launch improvements:**

Once the system is trained and launched it can be improved in several ways. Oftentimes, the tendency to collect more data and retraining can lead the system towards providing better recommendations. Incremental learning and online learning methods can also be used to continuously improve the quality of the system.

**Technical details:**

Python is an effective language to implement, train and cross validate recommender systems prior to their launch. But once the system is found after experimentation it can be optimized using multi-threaded c++ (partial or complete). It can be dockerized to take advantage of containerization and cluster orchestration. Code can be organized over a gitlab or bitbucket repository. For effective task management jira or asana can be used.

**Non-Functional description**

Additional functionalities that will be responsible for the sustainability and better user experience of this system can be listed as:

1. Crash handling & errors logging.

2. Low latency of the system.

3. Minimum hardware usage for cost-effectiveness.

4. Test cases to ensure reliability of the system.

5. Security to authorise users.

**6.** Scalability of the system

**Hardware Requirements**

Hardware required for model inference in deployment and its corresponding cost is:

1. VM (only CPU) on GCP - $0.16/hour

**Time and Cost Estimates**

|  |  |
| --- | --- |
| A tentative time and cost estimate of each of the above milestones is given in the table below. **Milestone** | **Time (business days)** |
| Data Preparation Pipeline |  |
| Recommender System |  |
| Web App |  |
| Non-Functional Requirements |  |
| API development |  |
| Deployment and Testing |  |