Tourist Information - Recommender System Office

You are a tourist information office that wants to suggest their registered users the next destination to visit. A lot of users have registered on your platform and they check-in places they have visited. These places are of any sorts and do not contain only historical landmarks. They can be anything from restaurants to interesting locations in a city. Instead of asking directly at your offices, visitors want to see where their friends have been electronically. Hence, they might want to go and visit these places confirming their common interests with their peers. Your goal is to design and test a Recommender System in the ***tourism*** *domain.*

There will be two different projects for students who interacted and submitted the exercises from those that did not. Therefore, the first project description is reserved to those who have at least **+4.5** points from the laboratory exercises. The other is designed for those that don't reach the previously mentioned threshold.

Delivery:

- Programming language: Feel free to use either Java or Python or both. It would be better for you to use Java as, in this way, you are aligned with what we've done in the laboratory exercises. Nevertheless, it is understandable to use Python for its simplicity and availability of packages.
- You must submit a thorough report **(max 8 pages)** of the choices that you've made when implementing your recommender system and its evaluation as described at the end. **The evaluation will be weighted on the clarity and technical soundness of your report.** *Do not occupy an entire page presenting the project title and your student ID. Your student ID at the top left of the page is sufficient. Please do not describe the dataset features, Stanford (wikiMID) has already done that. There's no need for you to fill pages.*
- You must submit the code/notebook with the code without any errors **(especially missed exceptions)**. The methods in the code should be commented (Javadoc, Pydoc) **(+1 pts)**.
- First project: You are encouraged to find or build your own dataset **(+2pts)**. In the end, you need data about people and places where people have traveled or would like to travel. Plus, you can integrate your dataset with other types of information, for example social information (who is friend of whom) semantic information (if you manage to connect places to categories of a POI taxonomy). A taxonomy (manual) is here. If you decide to do so, please provide a section where you describe the data. Check out the following links:
    - https://data.world/datasets/travel
    - https://data.world/crawdad/mobility-traces-of-taxi-cabs
    - https://www.europeandataportal.eu/en/highlights/open-transport-data-european-data-portal)

First project:

You are given two datasets, Gowalla and Brightkite, that contain data on people movements expressed in latitudinal and longitudinal coordinates. Each place $P(l_i, l_j)$ situated in the geolocation (l_i, l_j) is denominated as a Point of Interest (POI). You don't actually have the explicit locations where people travel, but only the coordinates, however you can use google maps (or other tools) to associate a POI to the given coordinates. A certain visitor $v_i$has a set of POIs $\{P(l_i, l_j) \mid visited(v_i, P(l_i, l_j))\}$. Notice that the previous notations doesn't address the number of

times that $v_i$has visited $P(l_i, l_j)$. Assume that it does[1], and we decide to abuse the set notation for brevity purposes. Besides the POI data, the datasets provide an undirected graph $G(V, E)$ that models friendship relationships between visitors $v_1, \ldots, v_{|V|}$where $V$is the set of vertices in G. Hence, each edge $e = \{v_i, v_j\} \in E$denotes that $v_i$and $v_j$are friends.

Let us describe the dataset files for explainability purposes. Both datasets contain the following files:

- **<dataset-name>_totalCheckins.txt.gz**: It has five columns (user id, timestamp of check in, latitude, longitude, location id). The user id is self explicable. Latitude and longitude constitute each POI as mentioned above. Although it would be interesting to devise time-dependent recommender systems, it is out of scope for this project. Therefore, you can drop the timestamp column. *If the location ids are big numbers, remap them from 0 up to m where m is the maximum location number.*
- **<dataset-name>_edges.txt.gz**: It contains two columns (user id v_i, user id v_j). Each row represents a direct edge $e = (v_i, v_j)$. You can find the incoming edge (v_j, v_i)if you scroll the file. Hence, the graph is undirected.

Tasks:

Choose one of the previously enlisted datasets.

1. Select an abstract representation of each user according to their POIs and friendship graph from the following:
    a. **Those who aim up to 26/30:**
        i. A visitor $v_i$can be represented as an indicator vector $\underline{vis}$ of POIs. Therefore, $\underline{vis}[k] = 1$if $visited(v_i, P(l_i, l_j))$where k is the ID of $P(l_i, l_j)$; 0 otherwise.
    b. **Those who aim up to 30/30:**
        i. A visitor $v_i$can be represented as a weighted vector $\underline{vis}$ of POIs. Therefore, $\underline{vis}[k] = cts$if $visited(v_i, P(l_i, l_j))$where k is the ID of $P(l_i, l_j)$ and cts is the number of visits; 0 otherwise.
    c. **Those who aim at the 30/30 e lode mark:**
        i. Propose your own modelisation strategy providing meaningful insights.
2. Calculate the following:
    a. **Those who aim up to 26/30:**
        i. Jaccard and cosine-similarity between each visitor and report it in a csv file as *visitor1, visitor2, value*.
        ii. Select the top 10 most dense connected components of the graph. The density of a graph is measured as $D = \frac{2|E|}{|V|(|V|-1)}$. Report the components ordered by density in a csv file as "*component (set of visitor ids), density*".
    b. **Those who aim up to 30/30 e lode:**
        i. Find the top 10 brokers in the top 10 most dense connected components.
        ii. Let the top 20 brokers in the most dense connected component be the initial set of infected nodes. Perform a spreading of influence algorithm (Linear Threshold) on the components where the threshold of each node v_i in the component is $\frac{1}{deg(v_i)}$and the weight $w(e)$ of each arch $e = (v_i, v_j)$ is $w(e) = cos(v_i, v_j)$. Limit the number of iterations to 100.

[1] In that case, each visitor v_i would have a set of tuples <$P(l_i,l_j)$, cts> where cts is the number of times v_i has visited $P(l_i,l_j)$.

3. Design a recommender system either by implementing a state-of-the-art method or by inventing your own strategy (**for those aiming at 30/30 e lode**) according to the data[2].
4. **Evaluation**: To evaluate your system, perform a k-fold cross validation (60-20-20% for training, validation and test sets), and averaging performances. The evaluation is as follows:
    a. You remove k interests from the list of interests of each user
    b. You immerse them in a list of 2k interests (the other k must **NOT** belong to the initial set of user's interest and must be selected from the total set of different interests at random) and then
    c. You ask your system to select k' interests from these 2k for recommendation. The number k' is another parameter and you should make it vary from 1 to k. For a single user v_i, the performance is the fraction of k' recommended interests that were actually included among the k removed from v_i's original interests. For the entire system, it is the average over all users. The value k depends on the number of interests per user and the distribution. You are free to fix it **(those who aim at 26/30)** or to make it vary across experiments **(those who aim at 30/30 e lode)**.

Second project:

You are given WikiMID (you can download the datasets here or go to the project homepage for other formats), which doesn't contain touristic information per se, but, for any Twitter user, you are given a set of interests extracted from messages from their list of followers. The dataset contains the following tab-separated files[3]:

- **message-based_dataset.tsv**: It contains four columns (user id, tweet id, interest id, url to the interest). Here you can extract the interests of each user as expressed in the tweets that the authors of the project have extracted. You won't need the interest url for this project so feel free to drop that column.
- **message-based_interest_info.tsv**: It contains three columns (interest id, platform type, wikipedia page - no URL). There's no need for the platform type as we're not interested where the interest came from. Here you need to filter those wikipedia categories that describe tourism. You have to design a crawler (see link for a Pythonic solution) that searches on Wikipedia for the wikipedia page in the file and then extract its categories found at the bottom of the page. If one of the categories contains:
    - tourism, places, archeology, restaurant,
    - world heritage, city(ies), capital(s)
- **friend-based_dataset.tsv**: This is a file that depicts a directed graph modelling the relationship of followings. In other words, the users in the first column follow those in the second column. Notice that Twitter users can also be artificial things such as photography, photoshop, Roma Capitale, Hard Rock Café etc. Therefore, you can build a quasi-bipartite directed graph $G(V_1 \cup V_2, E)$ where $V_1 = \{u_1, \dots, u_n\}$, $V_2 = \{i_1, \dots, i_m\}$, and $E = \{e \mid i \neq j \wedge (e = (u_i, i_j) \vee e = (u_i, u_j))\}$. We use $following(u_i)$to denote all the interests $i_j$that $u_i$follows. G is quasi-bipartite because you might have that a user $u_i$follows i_j that doesn't have a wikipedia page, thus representing another common Twitter user u_j.
- **friend-based_interest_info.tsv**: This file contains information whether $following(u_i)$ have wikipedia pages (no URL). Those that have a wikipedia page denote interests. For instance,

[2] You can check the link for a lot of details on the type of recommender systems.

[3] Notice that you download datasets in two languages, Italian and English. Please use the second one since wikipedia interests are mapped to English pages rather than Italian ones.

Geoffrey Hinton, although a normal user, has a wikipedia page and, hence, classifies as an interest.

Tasks:

Choose one of the previously enlisted datasets.

1. Select a representation of each user according to their touristic interests:
    a. **Those who aim up to 26/30**: A user $u_i$can be represented as an indicator vector $\underline{int}$ of interests. Therefore, $\underline{int}[k] = 1$if $interest(v_i, i_j)$where k is the ID of i_j; 0 otherwise.
    b. **Those who aim at the 30/30**: Propose your own modelisation strategy providing meaningful insights (possibly using a taxonomy for the interests). Report how the performances change w.r.t. the previous indicator vector $\underline{int}$.
2. Calculate the following:
    a. **Those who aim up to 30/30:**
        i. Jaccard, cosine-similarity and Euclidean distance between each user. Report it in a csv file as *user1, user2, value*.
        ii. Create a user-to-user similarity graph where the weight of an edge $e = (u_i, u_j)$is equal to their cosine-similarity. Therefore, the graph formed is a complete dense graph where the degree of each node is n-1 where n is the number of users. Perturbate the graph - eliminate edges with uniform probability 1/n. Finally, find the top 20 brokers on this graph.
3. Design a recommender system either by implementing a state-of-the-art method or by inventing your own strategy (**for those aiming at 30/30**) according to the data[4].
4. **Evaluation**: To evaluate your system, perform a k-fold cross validation (60-20-20% for training, validation and test sets), and averaging performances. The evaluation is as follows:
    a. You remove k interests from the list of interests of each user
    b. You immerse them in a list of 2k interests (the other k must **NOT** belong to the initial set of user's interest and must be selected from the total set of different interests at random) and then
    c. You ask your system to select k' interests from these 2k for recommendation. The number k' is another parameter and you should make it vary from 1 to k. For a single user v_i, the performance is the fraction of k' recommended interests that were actually included among the k removed from v_i's original interests. For the entire system, it is the average over all users. The value k depends on the number of interests per user and the distribution. You are free to fix it **(those who aim at 26/30)** or to make it vary across experiments **(those who aim at 30/30 e lode)**.

[4] You can check the link for a lot of details on the type of recommender systems.