Recommending Accommodations using Hybrid Recommender Systems

Final Proposal for Galvanize Data Science Immersive Capstone

Introduction

When searching and booking a hotel room for our next vacation is only one click away on personal electronic devices, we are asking for more personalized search results than just the nearest or most affordable hotels. While search engines continue to provide us with almost instant access to reservations, we still struggle with millions of online reviews to manually choose our favorites. It is time for machine to automate this process of selecting not only the hotels that have the highest ratings, but also those with positive reviews from people that have the same taste with us. This project exploits hotel reviews provided by hotel visitors' explicit feedbacks (ratings, rating breakdown by services and written reviews in English), scraped and processed from TripAdvisor.com (original dataset link: http://sifaka.cs.uiuc.edu/~wang296/Data/index.html) The goal of this project is to recommend the most personalized hotels based on the user preferences, similarity between this user and other users with the same taste in hotels, and the other users' reviews of hotels. This goal is achieved by applying a hybrid recommender system that is the result of merging latent factor models and the neighborhood models. Recommender performance is evaluated through a top-K recommendation task, which suggests the 'top K recommended hotels' to a given user.

Data description

Raw data contains hotel information and user reviews for each hotel and is in .json format. After processing using Pandas, the user-item matrix including: hotel ID, hotel location, hotel overall rating and rating breakdowns for difference amenities and services, user names (on TripAdvisor), etc, can be found in the following format (.csv format converted to-from Pandas DataFrame):

```
   business service  cleanliness  front desk  hotel id  location  ratings  \
0               NaN          4.0         NaN     99774       4.0        4
1               NaN          5.0         NaN     99774       4.0        4
2               NaN          5.0         NaN     99774       5.0        5
3               NaN          3.0         NaN     99774       4.0        3
4               NaN          5.0         NaN     99774       5.0        4

   rooms  service            user                        user location  value
\
0    4.0      3.0  Stepballchange                           Strasbourg    4.0
1    3.0      5.0  strawberryshtc                           chicago,IL    5.0
2    4.0      5.0    travelingTch  Washington DC, District of Columbia    5.0
3    3.0      3.0        Dorit147                     Tel Aviv, Israel    3.0
4    5.0      5.0        ashley r                       Ottawa, Canada    3.0

    review date
0   March 29, 2012
1   April 02, 2011
2   June 23, 2009
3   August 11, 2012
4   September 05, 2010
```

Each hotel file contains a dictionary of information pertaining hotel address, hotel ID, hotel URL, hotel name and hotel price range. For instance:

```
{u'Address': u'<address class="addressReset"> <span rel="v:address"> <span
dir="ltr"><span class="street-address" property="v:street-address">77 Yesler
Way</span>, <span class="locality"><span property="v:locality">Seattle</
span>, <span property="v:region">WA</span> <span property="v:postal-
code">98104-2530</span></span> </span> </span> </address>',
 u'HotelID': u'72572',
 u'HotelURL': u'/ShowUserReviews-g60878-d72572-Reviews-
BEST_WESTERN_PLUS_Pioneer_Square_Hotel-Seattle_Washington.html',
 u'ImgURL': u'http://media-cdn.tripadvisor.com/media/ProviderThumbnails/dirs/
51/f5/51f5d5761c9d693626e59f8178be15442large.jpg',
 u'Name': u'BEST WESTERN PLUS Pioneer Square Hotel',
 u'Price': u'$117 - $189*'}
```

Each hotel contains multiple reviews. An example of the user review is in dictionary format as such:

```
'Content': u"We stayed here in late August. This hotel is a decent stay for a
decent price for this time of year. The service is awesome. They would clean
our room in the morning and again in the late afternoon with candy on the
pillow. It is in such a historic area and right across the street from Pier,
underground tour and waterfront walk. The area is in a positive transistion
for such an older area of town. Lots to see and do and easy to catch the bus
to other areas of town. Hotel is totally redone and refurbished with great
service. If you don't care about having a room with a view, much cheaper
rates. It is however in the older area of town and the panhandlers are
everywhere. Panhandlers are quite polite, but aggresive."
```

The user reviews will be processed using Scikit learn and SpaCy tools with Natural Language Process models, such as Tf-Idf, and will serve as a vectorized additional user features for the recommender systems.

This dataset contains 1.05 million reviews from 546k users (anonymous users have been treated as one and will be processed separately). There are over 7k hotels. Visitors to these hotels reside in 73k difference cities and towns worldwide.

Next data steps

1. Perform EDA on visitor information matrix:

   - How many anonymous visitors have left reviews?
   - Are their impact negligible? If not, how to incorporate in the recommender system?
   - Who are the most frequent users? Are there any indicators of fake reviews?
   - When were the reviews written? Last 5 year? 10 year? Any hotel that has the most recent review longer than 3 years before the ending of dataset might be out of business, thus cannot present on the result of recommendation.
   - Do visitors tend to cluster in some countries?

2. Perform EDA on hotel information matrix:

- Do hotels tend to cluster in some areas and countries?
- Are hotel reviews well represented by countries and area?
- For each hotel, get the earliest and latest review dates. Detect any new hotel and/or inactive (out of business or remodeling, etc.).

Next modeling steps

1. A preliminary recommender using low-rank matrix factorization method:

    1.1) Set up raw ratings data: getting ratings and user names into DataFrame, normalize by each user's mean rating, convert to numpy array.

    1.2) Singular value decomposition using Scipy tool, 'svds', to get the 'sigma' matrix.

    1.3) Making preliminary predictions from the decomposed matrix:

        1.3)1) Create a training set and a validation set.

        1.3)2) Optimize the number of latent features by minimizing RMSE.

        1.3)3) Generalize to unseen data.

    1.4) Return hotels with the highest predicted overall rating to a specified user.

2. Extending model 1. to NSVD model, the "improved regularized singular value decomposition" (Paterek, A., Improving regularized singular value decomposition for collaborative filtering. KDDCup.07 August 12, 2007, San Jose, California, USA).

3. Build a neighborhood model:

    3.1) Using traditional Pearson correlation to measure similarity between hotels.

    3.2) Using a new neighborhood model with a set of global weights, instead of user-specific weights in the previous model to measure similarity between hotels. Apply a customized simple gradient descent solver to compensate the time it takes a regular least square solver on the cost function. (Koren, Y., Factorization meets the neighborhood: a multifaceted collaborative filtering model. KDD'08 August 24-27, 2008, Las Vegas, Nevada, USA)

    3.3) Plot RMSE's over different numbers of parameters (k ranging from 250 to infinity, which is numerically equal to 7000, the number of hotels in our training data). Compare the two models' RMSE.

4. Perform natural language processing for the review corpus (word2vec):

    - For each hotel, extract the latent features from the reviews provided by visitors.

5. Integrate the improved NSVD model with the new neighborhood model with global weights. Add on feature vectors from step 4. Compare runtime for 50, 100, 200 factors and RMSE for each model.

6. Evaluate through a top-K recommender. Pondering on the question of lowering RMSE blindly. Question to consider instead: what effect on user experience should we expect by lowering the RMSE by a 10% or some other number? Select all 5-star ratings from our validation set as a proxy for hotels that interest users. The goal is translated to finding the relative place of these interesting hotels within the total order of hotels sorted by predicted rating for a specific visitor. Compare performances of average ratings model, Pearson's correlation neighborhood model, weighted neighborhood model, NSVD model and the integrated model on one plot.

Potential problems

1. Runtime might be long during the Pearson's correlation similarity calculations. Can break down to sub-samples, then test the algorithm on, say, 100 hotels.

2. Not sure how to integrate step 4, word2vec, to the integrated model. Might serve as EDA procedure, if too complicated.

3. Step 5, the global weights that are independent of any specific user, may be hard to implement. If too complicated, need to switch back to traditional measurements of similarity, with a relatively short runtime, for testing purpose.