**A REPORT ON**

**HOTEL RECOMMENDATION**

**By**

Ayush Jain 2011A7PS066P

Aradhya Saxena 2011C6PS599P

**Prepared for the course**

**Advanced Data Mining**

**Submitted To**

**Dr Poonam Goyal,**

**Instructor In-charge.**

C:\Users\aniket lohia\Desktop\bitslogo-3-copy-copy.gif

Birla Institute of Technology & Science, Pilani

APRIL 2014.

**ACKNOWLEDGEMENTS**

We would like to express our gratitude towards the management at BITS Pilani, including Prof B.N. Jain and staff from Computer Science Department at BITS Pilani for ensuring that we have the opportunity to study the course.

We would also like to thank the various publishers, researchers and authors whose material has been the guidance for our report.   
We are especially indebted to Dr Poonam Goyal, the course instructor who has been guiding us throughout the project and through the course structure been the motivation behind it. We thank her for the project ideas that were responsible for our enthusiasm in the project.

# ABSTRACT

We implemented a hotel recommender system which consists of content based filtering as well as collaborative based filtering of user reviews to provide n top recommendations (n as specified by user).

**In content based filtering**,test user chooses a particular city in which he wants to travel and get recommendations of hotels located in that city. Our recommender system gives recommendations based on similarity of content(review) of already visited hotels around the world and most popular reviews of hotels in the city test user wants to travel.

This is achieved by creating a test user profile using ensemble classifiers. Since data is unlabeled , we do topic modeling using NMF(Non-negative Matrix Factorization) that creates certain broad topics in which each of the test user review belongs. Thus, we have target classes for training data (user reviews) and ensemble classifier is applied on testing data ( most popular reviews of hotels in the city in which test user wants to travel ).

Each topic has a priority based on training data\_target classes as it tells us that test user talks more about certain features in comparison to others. Based on this weight-age and sentiment score of each testing review , a overall score of each test hotel is calculated by summation of (sentiment\_score\* topic\_classified\_priority) of each review of test hotel.

The top n recommendations are given based on this score of different hotels in that city.

In collaborative based filtering, we take two different approaches:

a) User Based Filtering:

In user based approach, similarity of users with the test user is calculated based on the number of same hotels visited by both of them till now using pearson coefficient and eucledian distance. That similarity score is multiplied by rating of hotel visited by the similar user but not test user. Since a hotel can be visited by multiple similar users ,summation of this product is taken to get the final score. As a result, all such hotels are sorted based on this score and top n recommendations are provided.

b) Item Based Filtering:

In item based approach, each item is a vector representing ratings of different users of that particular item (hotel in our case).This is obtained by taking inverse of the previous matrix in user based approach. Similarity of items is calculated similar to similarity of users as mentioned previously. All the items of test user are iterated and similar items of each item of test user are found so that their similarity \* rating can be obtained. Since a particular similar item can come through multiple items of test user, summation is taken of this product and top n recommendations are given based on this score.

Item based filtering is very helpful when number of users are much larger than the number of items. Thus,calculating the similarity of user is much more expensive than calculating the similarity of items.

**PROPOSED TEXT MINING APPROACH AND METHODLOGY**

**Phase 1: Preprocessing**

* **DATA COLLECTION (Customer Reviews):** We took the dataset for the hotels from www.tripadvisor .com . The dataset available is the collection of reviews for different hotels of different cities worldwide.
* **DATA PREPROCESSED:** Removing unneeded word is a basic operation when we mine unstructured data . We need to clean the data through basic operations. Therefore collected reviews are preprocessed by using the NLTK(natural language toolkit) platform in python
  + **REMOVING STOP WORDS:** We have used the corpus stopword from ntlk that removes the stopwords. It removes words like a,an the etc from our reviews
  + **STEMMING:** We have used the corpus wordnet from nltk in python which lemmatizes the word using wordnet’s built in morphy function from nltk.stem.wordnet
  + **PUNCTUATION REMOVAL:** For getting text free of any punctuation like ! or ‘ , : etc we use the RE(regular expression) operation module in python.
* **DATA PROCESSING:**
  + **STEP 1: PART-OF-SPEECH TAGGING (POS)** :POS tagging is the part of speech tagging from natural language processing. The process of classifying words into their parts of speech and labeling them accordingly is known as part-of-speech tagging.   
    We use the nltk.tag package in python for this. Following are list of POS tag generated.

| **Tag** | **Meaning** | **Examples** |
| --- | --- | --- |
| ADJ | adjective | *new, good, high, special, big, local* |
| ADV | adverb | *really, already, still, early, now* |
| CNJ | conjunction | *and, or, but, if, while, although* |
| DET | determiner | *the, a, some, most, every, no* |
| EX | existential | *there, there's* |
| FW | foreign word | *dolce, ersatz, esprit, quo, maitre* |
| MOD | modal verb | *will, can, would, may, must, should* |
| N | noun | *year, home, costs, time, education* |
| NP | proper noun | *Alison, Africa, April, Washington* |
| NUM | number | *twenty-four, fourth, 1991, 14:24* |
| PRO | pronoun | *he, their, her, its, my, I, us* |
| P | preposition | *on, of, at, with, by, into, under* |
| TO | the word *to* | *to* |
| UH | interjection | *ah, bang, ha, oops* |
| V | verb | *is, has, get, do, make, see, run* |
| VD | past tense | *said, took, told, made, asked* |
| VG | present participle | *making, going, playing, working* |
| VN | past participle | *given, taken, begun, sung* |
| WH | *wh* determiner | *who, which, when, what, where, how* |

**COLLABORATIVE FILTERING**

Users having similar profiles may share similar interests. For a user, information can be filtered in/out regarding to the behaviors of his or her similar users. For simplicity we have explicitly provided the rating of each user to some of the hotels he has visited.

**PART1: User based filtering technique**:

Now we need somehow a metric to determine how similar the users are compared to the test user’s tastes. To measure this, we have to compare each user with other using a similar measure distance. To evaluate this metric we use the [**pearson correlation**](http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient). The basic idea behind this measure is that the more the users have similar tastes the more they are next to each other in the preferences search space. We use the pearson correlation because generally it gets slightly better results compared to Euclidian distance since it shows how much the variables change together.

Since we have the measure distances to compare two users, we now can define other function to classify all users compared to a specified user and find the one that is most similar. In this particular case, the goal is to find users that rated and have the similar taste so that one can know who he can ask for advice when i want to choose a hotel. The function *topMatches* returns a sorted list of *n* users with similar preferences to a specified user. Now, with the list, we can see the ratings done by other users that have similar preferences as the test user. So the idea he should see the hotels that he rated, then choose new hotels.

Generally what we really want is to make recommendation of hotels not users. We could simply look to the user profile and seek for hotels the user likes and the test user hasn't been yet, but this it's not so correct. This approach could eventually result in a user that hasn't done an evaluation on hotel that the test user could like. It could also return a user that liked a hotel that was badly reviewed (low rates) by all other users returned by the *topMatches*. To solve those problems, we give rates to items using a weighted average that can properly classify the evaluations.

The function *getRecommendations* looks at each user except the one passed as parameter which is the test user. It calculates how similar the users are to the specified user and after looks at each item rated by those users. As result you now have a classified hotels list and also a estimated rate that the test user would give for each book in it.

Finally the hotel will be recommendation according to the ranks.

**PART2: Item based filtering technique:**

The way how the recommendation system works, using user based [collaborative filtering](http://en.wikipedia.org/wiki/Collaborative_filtering), it requires all recommendations of each user to build a data set. Obviously, it will work well with a small amount of items and users, but in the real world it would be inefficient and web stores like imdb or amazon where there are millions of users and products, it would become impracticable to compare each user against other users, and also each item that the user has evaluated - it would be extremely slow.

Hence item based filtering technique is used. The procedure to do the filtering based on items is similar that we have discussed earlier with user-based collaborative filtering. The basic idea is to pre-process the most similar items against each other. So when the user asks for a recommendation, the system examines the items with the greatest scores and it builds a ranked list calculating the weighted average of the items similar to the items in the user profile. The main difference between the two approaches is that in the item-based one the comparison between the items will not change so frequently as the comparison between users in the user-based technique. It means it will not be necessary to evaluate every time the similarity between the items.

We implemented the function called calculateSimilarItems(), it builds a dictionary of items showing for each one the items that they are most similar.

Now we are ready to make recommendations using the data set of similarities between items without needing to examine all data set. The function getRecommendedItens() does it, the basic idea behind it is to compare the items the test user hasn't rated yet against items that he already has evaluated. This comparison uses some calculations and as result we receive a prediction of the rate the test user would give for the item

**RESULTS**

**CONCLUSION**