**Corporate Gifting Recommender**

**Abstract**

The objective of this assignment is to outline how to provide recommendations on items to users given a database of user profile status and interests and a database of product descriptions and tags. In order to accomplish that the optimistic goal of inferring a user’s preferences for items based on their interests and profile status is rather necessary and fitting to the context. It’s a challenging task to establish the relationship between the user’s interest and the item descriptions. Why? Because they both are two separate entities and therefore have to pick a suitable model to establish that relationship. This assignment suggests using Latent Dirichlet Allocation to group similar items under latent topics for every user’s interests. Then finding the similarity score (cosine similarity for example) between the user interests and the latent topics of grouped items will establish the relationship between the two. These item details in turn can be linked to unique item\_id. The LDA model also characterizes each user’s interests, thereby indirectly the user, by a vector of weights that indicate how strongly a user identifies with each topic of items. These vectors can be used to summarize a user’s preferences for items. The users with latent topics are represented in a vector space via Locality Sensitive Hashing, where topic vector of items that are closer to each other, based on a distance metric, are mapped to similar hashes or in the same bucket with high probability. Then comes the recommendation part. It is possible to rank the items under each topic according to descending probability value and pick the top ‘k’ ones in each related topic and recommend to similar users in the same bucket.

**My thought process**

Why topic modeling and LDA?

I had text data to deal with and so my default choice was some kind of topic modeling. But which distribution to choose to represent the data became a question. The LDA model, which is a generative probabilistic model, allows documents to exhibit multiple topics to different degrees. This overcomes the limitations of the mixture of unigrams model, which only allows a single topic per document. Strictly pertaining to the nature of the data at the context, it is possible to have one product appear in more topics and a user liking more products across various topics. It is also notable that the LDA model does not attempt to model the order of words within a document. It is precisely this “bag-of-words” concept that is central to the efficiency of the LDA model. So I choose LDA.

**Execution**

Step 1: Does topic modeling on product description and finds the hidden latent topics by grouping similar items together.

But the challenge is to find similar items according to each user’s interest/profile status. It’s significant to customize the recommendation according to user’s profile/interests. Otherwise it’s not possible to justify the recommendations to user’s likings. Once the topics are found the similarity (similarity can be established using cosine similarity) between a user’s interest and every latent topic is calculated. The similarity is thus calculated for every user’s interest.

For example:

The dataframe or table looks like:

|  |  |  |  |
| --- | --- | --- | --- |
| User\_id | User\_interest | Product\_id | Product\_description |
| 45237 | “opensource evangelist &  design thinker,  startups, Google,  Ride sharing” | prod\_1 | "It's the small iPod  with a big idea: Video, cult products" |
| 32000 | “Artist, ballet dancer, expressionist, gossiper with colorful vision” | prod\_2 | A great light offering flexibilty and functionality around the home and garden. |

Product details with name and unique product\_id can be stored in another table for reference.

|  |  |  |
| --- | --- | --- |
| Product\_id | Product\_name | Product\_description |
| prod\_1 | iPod | "It's the small iPod  with a big idea: Video, cult products" |
| prod\_2 | Bellhop Battery | A great light offering flexibility and functionality around the home and garden. |

The hidden topics with terms, probabilities, and their corresponding product\_id in the same order are displayed by LDA as follows (example with three latent topics):

Topic #0: ipod, 0.85, prod\_1; earphone, 0.62 , prod\_15; headset, 0.90, prod\_120

Topic #1: wine, 0.92, prod\_300; glass, 0.89, prod\_35; cheese, 0.75, prod\_98

Topic #2: camera, 0.74, prod\_111; lens, 0.93, prod\_267; light, 0.96, prod\_198

Step 2: Finds similarity between user’s interest and the latent topics from Step 1.

Getting tf-idf of the user’s interest and tf-idf of terms in each latent topic will allow us to calculate cosine similarity between the two entities. The method is repeated for every latent topic and for every user. Suppose if there are 20 latent topics, the procedure is repeated for every latent topic per user. Every latent topic has terms or words associated with different products. The idea is to represent a utility matrix with products and users. Here we are trying to choose the products according to user’s interest and which is accomplished through the similarity measure. It’s possible to list the products in descending order, in every latent topic, based on the probability values produced by LDA model. We shall pick, say, on average top 5 products in every latent topic based on their high probability values for every user and represent a dataframe with user\_id and prod\_id. There could a lot of common products and more uncommon. As can be imagined it could be a very big sparse matrix. And the dataframe has the probability values (from the LDA model) filled in for appropriate fields. Techniques like SVD can be used for matrix completion. Now what we have is a utility matrix of user\_id and different prod\_id or items.

Step 3: Why Locality Sensitive Hashing (LSH) and how does it work?

Instead of finding cosine similarity between every pair of users, an idea to leverage the technique called Locality Sensitive Hashing popped up. Locality Sensitive Hashing helps to identify the approximate nearest neighbors in high dimension. The idea is to take the topic vectors of each user and hash them so that similar tastes will end up in the same bucket. The locality is preserved with respect to a distance function. It helps with reducing the high dimensional features to smaller dimensions while preserving the differentiability and grouping similar objects (similar products) into same buckets with high probability. Following are the 3 latent topics that resulted from the LDA model with topic vectors that are associated with their corresponding probabilities and respective unique product\_id, and user\_id.

Topic #0:

coffeetable, 0.92, prod\_3324, 54678

mug, 0.68, prod\_6574, 89076

handcraft, 0.87, prod\_7865, 10087

filter, 0.76, prod\_3874, 21322

Topic #1:

iphone, 0.50, prod\_1134, 10078

hardware, 0.63, prod\_3000, 89666

watch, 0.43, prod\_8765, 30087

apple, 0.59, prod\_9111, 77320

Topic #2:

light, 0.92, prod\_1024, 94076

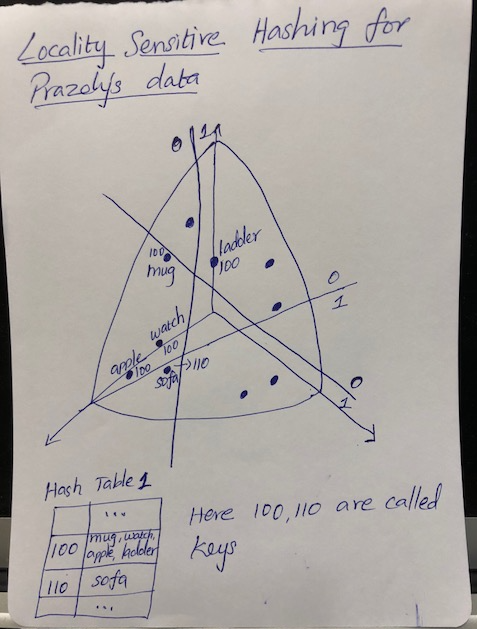
sofa, 0.68, prod\_4074, 11010

wood, 0.87, prod\_2065, 66632

ladder, 0.76, prod\_5564, 82223

These topic vectors of users are represented in space where it is cut into various zones by passing hyperplanes through the angle between two points that are represented in the space. LSH operates on an assumption that randomly partitioning the space will allow the pair points that are closer to collide than to move farther. The hash table is a data structure that is composed of buckets, each of which is indexed by a hash code, which is called as key. There are two basic strategies for using hash codes to perform nearest (near) neighbor search: hash table lookup and fast distance approximation.

**An illustration of how the topic vectors get represented using LSH**



As can be seen from the figure in the bucket with key 100 we can find the topic vectors: apple, watch, mug, and ladder. Although ladder and mug have same key value they are not allowed in the same bucket with apple and watch because they both are far away. Both ladder and mug will be removed from the bucket as false positives. The algorithm is repeated by generating random hyperplanes for a fixed number of hash tables. LSH provides the query time of O(log N) if K number of hyperplanes are set to log N and that is impressive when compared to doing things brute force and getting O(N) time complexity.

Step 4: How’s recommendation done?

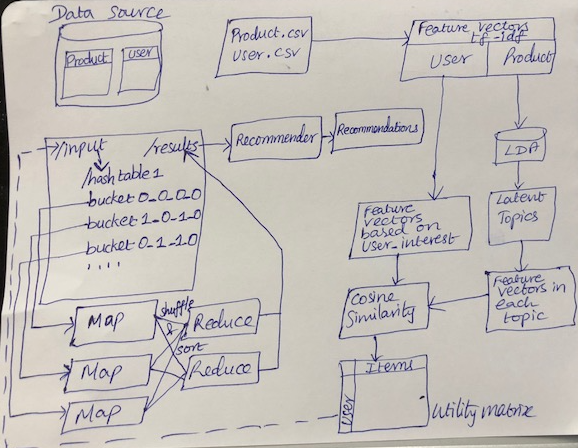
Scenario 1: **To scale to billions of users**

After the step of hashing algorithm, the buckets where the new values got placed are retrieved and cosine similarity is calculated between the new user and every different user in that bucket. If cosine similarity is used then the values are expected to be high or if distance metrics like cosine distance is calculated then it is expected to be low. So instead of finding, say, the cosine similarity between every pair of users in the utility matrix, it is calculated only between the new user’s value and the values of other users in the bucket. A threshold value is observed to decide who will become the nearest neighbors to the new user. For example, let’s say we are calculating the Euclidean distance between the value of the topic vector of the new user and that of the user in the bucket and we have 5 users in the bucket. We observe a threshold of 0.7 and only the users with values greater than 0.7 will be considered as nearest neighbors to the new user. The values of items thus grouped from the similar users can be sorted in descending order. The items with highest values are obviously recommended to the new user. Computing the nearest neighbors can be done in parallel using Map-reduce so that we may scale to billions of users without memory or running time becoming an issue.

Scenario 2: **When dataset can be fitted into RAM**

In the beginning of deploying the recommendation model, the dataset might be small. In that case it will be easier to just calculate the pairwise cosine similarity between the new user and all other users in the utility matrix. The items of the users with high similarity score are grouped, ranked according to magnitude, and recommended to the new user.

**Model Prototype**



**Cold Start Problem**

The utility matrix in this project will be built by finding the similarity between user’s interest and latent topics of products that are closely associated to the user’s interest semantically. Instead of inferring the user’s favorite products from external sources of data or from the explicit feedbacks from the user, which would be unavailable at the very beginning of deploying the recommendation engine, the recommendation model in this assignment infers user’s, likely, favorite products/items by associating terms in the latent topics, that’s comprised through topic modeling the product description, with the user’s interest/social profile through cosine similarity metric. Let’s consider the following use cases:

Scenario: When it’s not possible to divide the dataset into training and test sets

The model will perform the topic modeling part and find out the closely associated products according to the user’s interests. LDA does well with small documents and therefore with small vocabularies, but the latent topics provided by LDA might be different than it normally does with large vocabularies. So not sure whether it would provide meaningful latent topics for later associating words, representing the products, with those topics. Nevertheless, it will be a good starting point to provide recommendations as the terms will be closely associated with user’s interests and profile status and wait for more user interactions, explicit feedback, and inferred insights about user behavior to enhance the model’s performance.

**Evaluating the recommender’s performance:**

It’s rather hard, as currently we do not have rating data from the users to evaluate the accuracy of the model using metrics like root mean square error and absolute mean error. But we shall try to control the factors that could be verified to check whether we are in the right direction like the following:

* Determine the optimum number of topics for topic modeling through LDA model using Gibbs sampling.
* It’s possible to tune the parameters alpha and beta in LDA to minimize divergence between the variational and posterior distributions thereby maximizing the likelihood of the observed data.
* The affinity categories in the user’s profile status: high, medium, and low can shed light on deeper insights re user behavior. So could build a multi-class classification model with three categories and have to incorporate external data to establish ground truth. We shall start with probability values for each category and also include other useful features to build the classification model. It’s possible to cross-validate and evaluate the performance of the classification model. Then have to think about a way to involve the results in building the recommendations.

**Future Research**

* Considering only the items that are well regarded as corporate gifts (for example: avoiding clothing except for scarves or ties, ignoring jewelry, etc.,) will refine the recommendations.
* Ways to incorporate feedback from users re whether they liked the recommended gift and also devising a threshold to recommend an item or similar one after some time will improve the model’s accuracy in providing the recommendations.
* I strongly lean towards finding the conditional probability of whether a user will like an item given their affinity towards a specific interest. Let’s consider the use case of a user having high affinity towards a specific interest. According to Bayes’ rule the conditional probability can be calculated as follows:

**P (liking an item| affinity=High)=P (High affinity| liking an item) \* P (liking an item) / P (High affinity)**

But doing this gets computationally expensive because finding the value for the term P (High affinity| liking an item) is complicated. It’s necessary to associate these two terms in order to find the probability. There are a lot of terms in each of the three affinity categories (High, Medium, and Low). Google follows similar Bayesian approach to predict user’s genuine news interest based on users past click behavior. It will be good information filtering step before collaborative filtering. But in the context it is much difficult to compute that conditional probability value for every affinity category of user’s interests and every latent topic of the products. Hidden Markov Models can prove useful and I have experience building it. But didn’t have time to think through an idea involving it.

* User’s interest and their profile status change over time. So need to incorporate the change to reflect in providing recommendations.
* Have to start considering many useful signals such as location, gender, age (could infer if not available), and especially location in case of considering location specific in-person gifts to users.
* The challenge in this project is to infer the user’s favorite products and likings from their interests and profile status. We can use external data source like historical data for example to infer about a person who likes startups might also like wearable technology products.

**Reference**

<https://link.springer.com/article/10.1023/A:1011419012209>

<https://people.mpi-sws.org/~gummadi/papers/recsys14-userinterests.pdf>

<https://www.research.ibm.com/haifa/dept/imt/papers/GuyWWW13.pdf>

<https://arxiv.org/pdf/1804.04095.pdf>

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.33.9040&rep=rep1&type=pdf>

<https://mimno.infosci.cornell.edu/nips2013ws/nips2013tm_submission_23.pdf>

<https://www.microsoft.com/en-us/research/wp-content/uploads/2014/05/ICWSM2014.pdf>

<http://maroo.cs.umass.edu/getpdf.php?id=1101>

<https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/35599.pdf>

<http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf>

<https://www.ics.uci.edu/~asuncion/pubs/KDD_08.pdf>

<https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/35599.pdf>