Recommender Systems

June-July 2017

Saumya Rawat

ABSTRACT

In today's world of growing online retailers, any consumer at an online marketing website is flooded by product choices. In such a scenario, personalized recommendations play an important role in helping one discover new products or sorting through large choice sets and in turn help companies generate more revenue while increasing customer retention and overall customer satisfaction. Enter recommender systems, they provide a scalable way of personalising content for users in scenarios with many items. Through experiments on an ecommerce dataset: Retail Rocket, I attempt to overcome various challenges associated with building a robust recommender system for a generalized implicit ecommerce dataset like producing accurate recommendations, handling many recommendations efficiently and coping with the vast growth of number of participants in the system.

PROBLEM STATEMENT

- Given a set of users and any kind of interactions with products, including but not limited to, views, add to cart and transaction, recommend top 'k' items to the current active user based on his/her implicit interactions with the relevant and irrelevant items in the dataset and their transitive relation.
- Experiment with different kinds of recommender systems and evaluate their performance on the Retail Rocket dataset using a relevant evaluation metric.

OVERVIEW

To look for an item for a user, search queries are appropriate when users have explicit intent, but they perform poorly when the intent is difficult to express or if the user is simply looking to be inspired. A recommendation, the nucleus of a recommendation system, is a suggestion or proposal as to the best course of action for a particular user. The role of a recommendation system is predicting user responses to options and suggesting those the user would be most likely to select.

Recommendations in any domain help users retrieve high quality and relevant information. For example, in health related websites, any kind of shopping portals, video and music streaming services, even news articles, as long as there is data captured about the interactions of users with an item and some notion of preference between users and items, the use of a recommender system becomes a theoretical possibility. For companies such as Amazon, Netflix, and Spotify, recommender systems drive significant engagement and revenue. Take the case of Netflix, according to an article published by them in January, 2016, their recommendation algorithm influences viewers' choices for about 80% of hours streamed and the combined effect of personalization and recommendations save them more than $1B per year.

THEORY

Recommender systems apply data mining techniques and prediction algorithms to predict users' interest on information, products and services among the tremendous amount of available items. The two main entities in any recommender system are: USER and the ITEM. The user for which the recommendations are being made is generally referred to as an active user. There are many recommendation paradigms in use today. The usual classification is by the type of data that is used to generate recommendations.

The choice of a paradigm is driven by the kind of interaction that is recorded between the user and the item. Explicit feedback is when the user is asked to give a quantitative rating to the item or like/dislike it. Here, both positive and negative information is recorded.

Typically, recommender systems use ratings as a mechanism to proactively express their interests in items and seamlessly collected clickstream data for inferring users' interests or preferences, the latter is known as implicit feedback.

We can recommend options to users based on the following ratiocination:

1. Tell me what's popular among my peers. This kind of logic is commonly called *Collaborative Filtering*. It is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating).

2. Show me items similar to what I have liked in the past. This method is based on the past history of product purchases focusing on the similarity of each item and is called *Content Based*.

3. A variety of methods use a combination of the two methods given above and are known as hybrid approaches.

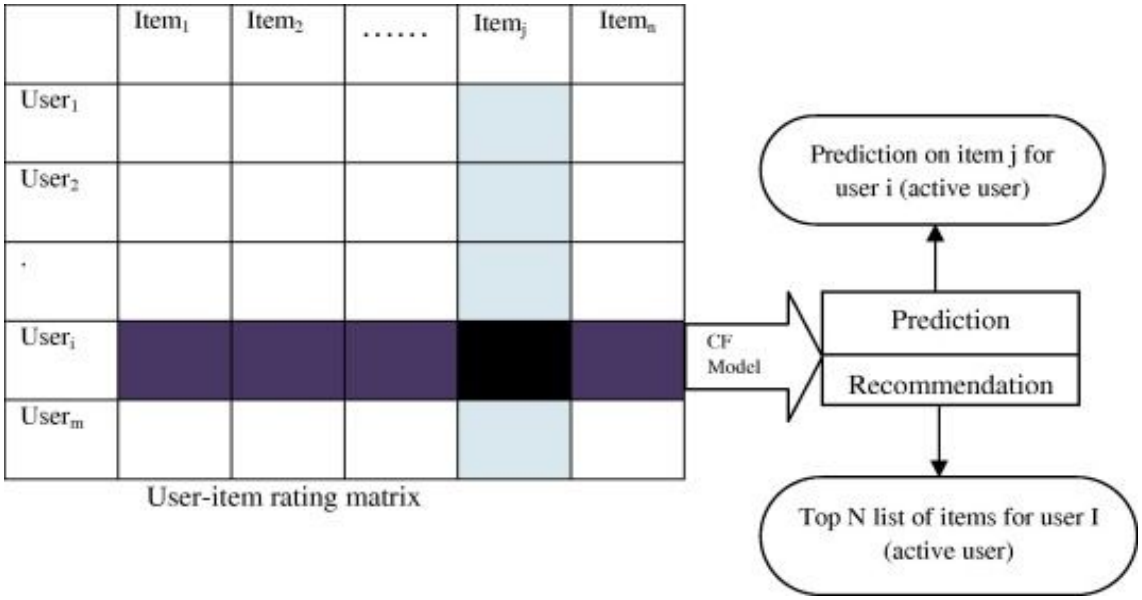
Collaborative Filtering Recommendation System

The most commonly used and simple class of algorithms in recommender systems is Collaborative Filtering. Since the final approach is based on a collaborative filtering approach it has been explained in greater detail than other models. It works under the assumption that an active user will be interested in items that users similar to him have rated highly. This is a suitable method for content that cannot easily and adequately be described by metadata such as movies and music.

There are three main steps to make a prediction as to whether the user will like or dislike an item that he has not seen/purchased/rated as follows:

1. In the first step, users rate some items they have experienced previously.
2. In the second step, an **active user**'s profile is matched with other users' profiles in the system. A set of similar users also called **neighbours** of the active user are found.
3. In the last step, predictions are made for items that the active user has not rated based on the ratings provided by its nearest neighbours. Finally, these items are presented to the active user in a suitable order.

The algorithms uses a MxN matrix of preferences for M users and N items and these are used to predict missing preferences and recommend the items with high predictions as illustrated in the figure below.



There are two kinds of collaborative filtering based methods, Memory and Model based. Memory-based techniques rely heavily on simple similarity measures (Cosine similarity, Pearson correlation, Jaccard coefficient etc) to match similar people or items together. If we have a huge matrix with users on one dimension and items on the other, with the cells containing votes or likes, then memory-based techniques use similarity measures on two vectors (rows or columns) of such a matrix to generate a number representing similarity.

The similarity measures work in the following manner:

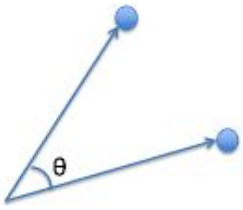
- Pearson correlation tries to measure how much two users vary together from their normal votes - that is, the direction/magnitude of each's vote in comparison to their voting average. If they vary in the same way on the items they have rated in common, they will get a positive correlation; otherwise, they will get a negative correlation. The pearson correlation is given by :

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n(\sum x^2) - (\sum x)^2]\,[n(\sum y^2) - (\sum y)^2]}}$$

Where n is the number of paired scores, and x and y are the two vectors being compared.

- Cosine Similarity measures the difference in the angle of the two vectors. If the two vectors generally point in the same direction, they get a positive similarity; if they point in opposite directions, they get a negative similarity. The cosine similarity is given by :

$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$



Model-based techniques on the other hand try to further fill out the user item matrix. They tackle the task of “guessing” how much a user will like an item that they did not encounter before. For that they utilize several machine learning algorithms (Bayesian, clustering) to train on the vector of items for a specific user, then they can build a model that can predict the user’s rating for a new item that has just been added to the system. These models can also be built using similarities between items rather than users and in fact, sometimes it is more desirable to do so. For example, the Netflix Prize data contains slightly fewer than 5,00,000 users, but only a little over 17,000 movies. This makes it likely that the resulting model over items will be smaller than that for users.

USE CASES

Amazon uses item-item collaborative filtering based matching to improve its recommendations. They match each of the user’s purchased and rated items to similar items, then combine those similar items into a recommendation list.

LIMITATIONS

In a cold-start scenario, where there are no preferences at all, a collaborative filtering based model cannot generate any recommendations. However, cold starts can also occur when there are millions of available preferences, because pure collaborative recommendation doesn't work for items or users with no ratings, and often performs pretty poorly when there are only a few ratings. Further, the underlying collaborative model may yield disappointing results when the preference matrix is sparse as it leads to inability to locate successful neighbors and finally, the generation of weak recommendations. It always requires tweaking, and never simply works out of the box. Synonymy is another prevalent problem for collaborative filtering. It is the tendency of very similar items to have different names or entries. Collaborative Filtering systems usually find no match between the two terms to be able to compute their similarity. Different methods, such as automatic term expansion, the construction of a thesaurus, and Singular Value Decomposition (SVD) can be used to combat this issue.

Content Based Recommendation System

Content based recommendation systems match the inherent properties of items liked by a user to unknown items and thus propose recommendations. It involves creating a profile for each item that describes the characteristics of the items, for eg, in the case of a movie the relevant features are the genre, director, cast. A content based recommendation system uses different types of models to find similarity between documents in order to generate meaningful recommendations. It could use Vector Space Model such as Term Frequency Inverse Document Frequency (TF/IDF) or Probabilistic models such as Naïve Bayes Classifier[1] , Decision Trees[2] or Neural Networks[3] to model the relationship between different documents within a corpus. These techniques make recommendations by

[1] "Content-based Recommendation Systems - fxpal." http://www.fxpal.com/publications/FXPAL-PR-06-383.pdf. Accessed 12 Jul. 2017.
[2] "A Decision Tree Based Recommender System - Journals." https://subs.emis.de/LNI/Proceedings/Proceedings165/170.pdf. Accessed 12 Jul. 2017.
[3] "Ask Me Any Rating: A Content-based Recommender System based on ...." 16 Dec. 2016, https://www.slideshare.net/AlessandroSuglia/ask-me-any-rating-a-contentbased-recommender-system-based-on-recurrent-neural-networks. Accessed 12 Jul. 2017.

learning the underlying model with either statistical analysis or machine learning techniques. Some of the problems associated with content-based filtering techniques are limited content analysis, overspecialization and sparsity of data

Hybrid

Hybrid models are a combination of both content based and collaborative filtering approach. It can be done in multiple ways, like a simple weighted combination of the two, or switching between the two based on a heuristic or using a cascaded mix where the successor's rec is restricted by the predecessor. Hybrid systems add a level of complexity to the solution and hence require an explicit target in mind while designing the system.

DATASET

Retail Rocket

The dataset is an ecommerce dataset hosted on Kaggle[4] and it has been collected form a real life ecommerce website. It consists of three kinds of files: behaviour data, item properties and category tree. It is raw data, i.e. without any content transformations, however, all values are hashed due to confidential issues. The behavior data file was used during the course of the project which contains record of any view, add to cart and transaction event on the website. There are in total there are 2,756,101 events including 2,664,312 views, 69,332 add to carts and 22,457 transactions produced by 1,407,580 unique visitors. Category Tree consists of category of item, category of parent.

Pre Processing

The dataset has a remarkably high sparsity of 99.99% which is much above the generally accepted as a rule of thumb sparsity mark of 97% for most recommender system algorithms that can deal with high sparsity. As a result a denser subset of the data is used. Given only the 'transaction' action items a confidence matrix is constructed that acts as an alternative to the explicit ratings matrix ordinarily used. A binary score of 1 and 0 is assigned for every item w.r.t every user based on whether they purchased the item or not.

[4] "retail rocket-recommendation | Kaggle." https://www.kaggle.com/riverxie/retail-rocket-recommendation. Accessed 12 Jul. 2017.

```
# PRE PROCESSING
# combine item matrix
item_properties = pd.concat(([item_1, item_2]), ignore_index=True)
events_full = events
test, train = pd.DataFrame([]), pd.DataFrame([])
n = events.shape[0]

# use only transaction data for now
events = events[events.event=='transaction']
events = events.sort_values('timestamp')
events = events[events.groupby('visitorid').visitorid.transform(len) > 1]
```

Train and Test set cannot be constructed in the traditional 80-20 ratio as that does not accurately model a real world which forms a time-series of purchases as they occur. Since a timestamp exists in the dataset for every event recorded, this attribute is used to order each user's transaction. 80% of the user's initial purchases are added to the train set while the latter 20% are added to the test set. This simulates a freezing effect on the events data that acts as if we are making predictions for a particular user at a given timestamp. A further improvement in the construction of the confidence matrix can be adding different weights for each action for example out of 10, since a transaction is the most confident indicator of a user's validation for a product it can be awarded 5 while an add to cart can be 3 and view can be 2. However this results in a larger matrix and adds the challenge of time complexity etc. which is not preferred for a real time recommender system.

```
# Iterate over all users with sorted transaction history and put top 80% in train and the latter
transactions in test
for name, group in events.groupby('visitorid'):
        train_size = int(math.ceil(0.7*n))
        test_size = int(math.floor(0.3*n))
        if test_size == 0:
        test_size = 1
        train = train.append(group.head(train_size))
        test = test.append(group.tail(test_size))
        ind +=1
print('Train and test constructed with sizes',train.shape[0],test.shape[0])
```

Also only those users are included who have made greater than 5 transactions.

```
# Construct confidence matrix based on whether the item was purchased by a
particular user (1) or not (0)
for index, row in train.iterrows():
        if row['event'] == 'transaction':
        ui_matrix[row['visitorid'],row['itemid']] = 1
sample = []
row_sums = ui_matrix[users,:].sum(axis=1)
sample = [i for i in range(len(row_sums)) if row_sums[i] >=5]
sampled_train = []
for user_index in sample:
        sampled_train.append(list(ui_matrix[users[user_index],:].A[0]))
print('Sampled_train constructed, contains item purchases (5+ items per row)')
```

INVESTIGATION

First the various kinds of recommender systems were studied from a broader angle, including Knowledge Based, Demographic and Utility based. Since the dataset consisted of specifically implicit feedback Collaborative Filtering methods were considered. An initial approach consisted of directly creating a ratings matrix out of the given events data by giving different weights to different events however on closer scrutiny it's limitations were exposed. Several recommendation algorithms do not account for the variability in human behavior and activity. Often, they are hardwired for explicit ratings rather than implicit ratings hence the absence of negative feedback in our case would not be accounted for in the algorithms and this approach was dropped.

ALTERNATIVE SOLUTIONS

For the evaluation of any recommender system on a particular dataset, unlike other domains we can not rely on numbers like accuracy. Specially in our case since we do not have an explicit dataset. As a result various kinds of approaches were tried before the final approach was fixed. The evaluation metric for these algorithms is discussed in greater detail later. The popular user based and item based correlation algorithms were evaluated but their performance as opposed to the final solution was poor.

The User-based algorithm that was evaluated matches the active user to similar users using the cosine similarity measure described above. Using the row vectors of the confidence matrix constructed an MxM correlation is calculated. Let this be $WC_{MxM.}$

Each product's potential confidence value for a user is then given by

$$WC \bullet A$$

Where A is the MxN confidence matrix.

The Item-based algorithm computes item similarities. Based on the columns of the confidence matrix, an NxN matrix WP_{NxN} is constructed.

Now each product's potential score for the user is given by

$$A \bullet WP$$

FINAL SOLUTION

The Link Analysis algorithm[5] formed the final solution for our dataset. This algorithm treats the entities in a recommender system as nodes in a graph. The users and items form a bipartite graph and a link between two such nodes indicates that the user can be a part of the product's consumer base and the product can be a part of the user's interest. Two aspects to this algorithm that need to be carefully handled are:

Normalize the effect of a very active consumer in our database. In this case active means one who has links to almost all products. And also that the WC_{NxN} and WP_{MxM} matrices do not converge to similar values. So an intermediate matrix CR is defined which maintains high weightage for the active user and custom score updates for individual users.

The steps for the algorithm are as follows:

- Initialize Customer Representative Matrix, CR as an MxM identity matrix
- While convergence on CR is false do,

$$PR(t) = A' \bullet CR(t-1)$$

[5] "A Comparison of Collaborative-Filtering ... - ACM Digital Library." 1 Sep. 2007, http://dl.acm.org/citation.cfm?id=1304521. Accessed 12 Jul. 2017.

$$CR(t) = B \bullet PR(t)$$

Normalize CR such that the row sum is 1

```
# LINK ANALYSIS
print('Commence link analysis')

gamma = 0.9
A = sampled_train
B = []
lol = []


for i,user in enumerate(sample):
        B.append(A[i]/row_sums[user,0]**gamma)

A = np.array(A)
B = np.array(B)
CR_prev=np.identity(len(sample))
CR_0=CR_prev
counter = 0
print('Begin loop')
while(True):
        PR = np.dot(A.T,CR_prev)
        CR_new = np.dot(B,PR)
        r_s = CR_new.sum(axis=1)
        new_matrix = CR_new / r_s[:, np.newaxis]
        CR_new = new_matrix+CR_0
        if(np.linalg.norm(CR_new-CR_prev))<=10:
        break
        CR_prev = CR_new
        counter += 1

print('Link Analysis Done')
PR=PR.T
```

The value of PR now gives the product score for users.

Recommendations are made by sorting the scores and choosing the top k value.

EVALUATION METRICS

Evaluation of a recommender system is a complicated task. Each metric focuses on a particular characteristic of the system. Popular metrics such as Root of the Mean Square Error (RMSE) measure the distance between predicted preferences and true preferences over items, while recall computes whether the good options were included in the recommendations or not. Clearly, it cannot be said that a single metric would be better than all others over all possible approaches. So we need to decide what aspect of the end product of a recommendation we shall be focusing on before deciding on a metric, it can be for instance, the revenue increase. As such, selecting the proper evaluation metric has a crucial influence on the selection of the recommender system algorithm that will be selected for deployment[6]. The following metrics can be used to evaluate a system's overall performance well, however they require feedback from users once the recommender system has been put into place.

Coverage: For how many users can we make recommendations? How many catalog items are ever recommended?

Diversity & Novelty: Avoiding monotone lists, discover new (families of) items

Serendipity: Unexpected and surprising items might be valuable

Familiarity: Give the user the impression of understanding his/her needs

Biases: Does the recommender only recommend popular items and blockbusters?

For the aforementioned system we use redefined values of Precision, Recall and F-Score[7]. Precision is defined as the ratio of recommendations that were present in the future transactions and the total number of recommendations while Recall is the ratio of recommendations that were present in the future transactions and the total number of

[6] "A Survey of Accuracy Evaluation Metrics of Recommendation Tasks." http://jmlr.csail.mit.edu/papers/volume10/gunawardana09a/gunawardana09a.pdf. Accessed 13 Jul. 2017.

[7] "A Link Analysis Approach to Recommendation under Sparse Data." https://pdfs.semanticscholar.org/2962/86ee15e413b39295c78387080e36c822cfd2.pdf. Accessed 13 Jul. 2017.

items in the future purchases. F-score is defined as the harmonic mean of Precision and Recall.

The numbers reported are as follows:

Algorithm	Precision	Recall	F-measure
User Based	0.054	0.073	0.062
Item Based	0.046	0.056	0.050
Link Analysis	0.144	0.075	0.089

It can be observed that overall Link Analysis as expected worked best and showed consistently high performance. Although the numbers are low in the traditional sense of Precision and Recall, in the case of recommender systems these help in choosing which algorithm to choose for our data and nothing more.

REVENUE

Recommender systems are a great way to perform product upsell and cross-sell to your existing customers. It is also a sure fire way to generate more revenue while increasing customer retention and overall customer satisfaction. A study conducted by Wharton School (on movie sales from a top retailer in North America) reports:

“For sales volume, we show that different algorithms have differential impacts, with some widely used algorithms having no impacts. Purchase-based collaborative filtering (“Consumers who bought this item also bought”) causes a 25% lift in views and a 35% lift in the number of items purchased over the control group (no recommender) for those who purchase. For sales diversity, we find that collaborative filtering algorithms cause individuals to discover and purchase a greater variety of products but push each individual to the same set of popular titles,leading to concentration bias at the aggregate level.”

CONCLUSION

We saw that for a sparse e-commerce dataset, building a recommendation system was a complex process that involves carefully studying the dimensions and the sparsity of the dataset. Three kinds of solutions are now at our disposal that are assumably viable for most e-commerce datasets. The evaluation of these systems requires a clear cut definition of a success measure that is hoped to be achieved from the deployment of the recommender system, which is in most cases revenue. For the given RetailRocket dataset, Link Analysis performed significantly better than the two other popular algorithms and on productionalizing can lead to higher customer retention and product upsell and cross-sell for existing customers.

References

Basics

1. Health Recommender Systems: Concepts, Requirements, Technical Basics and Challenges, Martin Wiesner, Daniel Pfeifer ,International journal of environmental research, 2014
2. Towards Health (Aware) Recommender Systems - Christoph Trattner
3. Reliable Medical Recommendation Systems with Patient Privacy
4. Item-based collaborative filtering http://www.cs.carleton.edu/cs_comps/0607/recommend/
5. Recommendation systems: Principles, methods and evaluation, F.O. Isinkaye, Y.O. Folajimi,B.A. Ojokoh
6. https://www.quora.com/Recommendation-Systems-What-is-the-difference-between-item-to-item-collaborative-filtering-and-content-based-filtering
7. Seroussi Yanir, “The wonderful world of recommender systems”, https://yanirseroussi.com/2015/10/02/the-wonderful-world-of-recommender-systems/
8. Jaimeel Shah, Lokesh Sahu, A Survey of Various Hybrid based Recommendation Method, Volume 4, Issue 11, November 2014 https://www.ijarcsse.com/docs/papers/Volume_4/11_November2014/V4I11-049pdf
9. M. Jones, Introduction to approaches and algorithms, Concepts that underlie web recommendation engines, https://www.ibm.com/developerworks/library/os-recommender1/
10. Michael J. Pazzani, Daniel Billsus, Content-based Recommendation Systems, http://www.fxpal.com/publications/FXPAL-PR-06-383.pdf

Case Studies and Sparsity

11. https://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf

12. Solving the Sparsity Problem: Collaborative Filtering via Indirect Similarities,Christian Desrosiers,George Karypis

13. Dunja Mladenić Blaž Fortuna, Marko Grobelnik, Data Sparsity Issues in the Collaborative Filtering Framework, October 2006 https://www.researchgate.net/publication/227044312

How to evaluate RS

14. Evaluating Recommender Systems, http://fastml.com/evaluating-recommender-systems/

15. https://medium.com/@m_n_malaeb/the-easy-guide-for-building-python-collaborative-filtering-recommendation-system-in-2017-d2736d2e92a8

16. http://www.ebaytechblog.com/2010/11/10/**measuring-search-relevance**/

Algorithms used in Collaborative Filtering

17. http://papers.nips.cc/paper/3208-**probabilistic-matrix-factorization**.pdf

18. http://www.cs.rochester.edu/twiki/pub/Main/HarpSeminar/**Factorization_Meets_the_Neighborhood-_a_Multifaceted_Collaborative_Filtering_Model**.pdf

19. Probabilistic Matrix Factorization, https://www.cs.toronto.edu/~hinton/csc2515/notes/pmf_tutorial.pdf

20. Claudio Rojas Bórquez, SVD++ and Probabilistic Matrix Factorization, http://dparra.sitios.ing.uc.cl/classes/recsys-2015-2/student_ppts/CRojas_SVDpp-PMF.pdf

Implicit Feedback

21. Hu Yifan, Koren Yehuda, Volinsky Chris, Collaborative Filtering for Implicit Feedback Datasets, 2008

22. Implicit - Python toolbox for recommender systems, https://github.com/benfred/implicit

23. Steinwood, Jesse, A gentle introduction to recommender systems with implicit feedback, https://jessesw.com/Rec-System/

24. Zan Huang, Daniel Zeng, Hsinchun Chen , A Link Analysis Approach to Recommendation under Sparse Data, https://pdfs.semanticscholar.org/2962/86ee15e413b39295c78387080e36c822cfd2.pdf