Reado - A personalized book recommender system

Case Study

Story

Milton Keynes lives in England and like almost any other 29 year old, he is finally in a place in his life where he has a job that pays the bills, a nice place to stay and finally some time that he can use to focus on his hobbies. Milton likes reading -but just not like any other person, he loves reading so much so that he has a small library in his house where he spends about 4 hours on average every day! He has already read and reviewed about 450 books. Now, that is something you don't expect a typical 29 year old. But Milton is not alone. There are many others like him, who like to have conversations with their favourite books. It wouldn't be an exaggeration to say that they live partly in a different dimension (fiction lovers will like this).

As it so happens, Milton's best friend Jake can't stop thinking about using the data on these ratings to find people similar to Milton and creating a recommendation algorithm that generates a list of books that he should read. This would make a great birthday gift for Milton - for a birthday which is not far! But the problem is much complex than it seems.

Firstly, the data is extremely sparse. The dataset at whole has a sparsity of more than 99.99%.

Secondly, the definition of similarity is very vague. How do we characterize similarity? How do we select similar entities?

Note:In this case study, we shall cover these as well as other pertinent aspects that can inform the solution. We also take a step-by-step approach in explaining our approach to elucidate our work to the diverse set of readers. We welcome any feedback and/or questions from our readers.

Business Perspective

Motivation

Thanks to Gutenberg and now, the digital boom, we now have access to a huge amount of collective intelligence, wisdom and stories. Indeed, humans perish but their voice continues to resonate through humans brains and minds long after they are gone - sometimes provoking us to think, making us parts of revolutions and sometimes confiding in us with their secrets. They have the ability to make us laugh, cry, think - think hard, and most importantly, change our lives the way, perhaps nobody else can. In this sense, books are truly our loyal friends.

Can the importance of books as loyal friends ever be overestimated? We think not. Which is why we think that creating just the 'right' recommendations for readers is a noble objective. Consider it a quieter (Shh.. no noise in this library! :)) Facebook or a classier Tinder for those who like to read and listen, patiently.

Vision

To provide *meaningfully relevant* book recommendations for readers based on their readership data.

Mission

We aim to achieve our vision through a systematic process which is given below:

1. Meaningful: The recommendations have to be something that the user is expected to find useful. Usefulness in this context can have several definitions in terms of genre, publisher, author etc. We begin with an assumption that similar people read and rate similar books and hence this similarity becomes a good discriminant for selection.
2. Relevant: There has to be a measurable performance that can represent the relevance of a recommendation. On a ten-point scale, we would not want the estimated ratings to be "off" by more than one point on the scale. For this our target MAE is less than 1.00 (for a ten point scale) which translates to less than 0.5 for a 5 point scale.
3. Exciting: We would want out recommendations to have an element of serendipity - indeed, a recommendation is more meaningfully relevant if it excites the user (which is the ultimate objective). This approach would require more data and we

intend to enrich our data and attempt to factor in this requirement in Phase II of our project.

Objective

We are trying to optimize on several parameters:

1. **RMSE/MAE**: It is important that the estimated ratings do not have a large variance to the extent that their relative rankings are affected. For this purpose, an ideal RMSE threshold has been fixed as mentioned above. We are also looking at MAE.
2. **Serendipity**: Although this is more subjective , it is a very important component in making a recommendation meaningfully relevant. We shall attempt to characterize, determine and measure this component in Phase II.

In our pursuit to provide meaningfully relevant book recommendation to readers, we are willing to compromise on the following:

1. **Absolute rating estimates**: Below a certain threshold, our approach would not aim to improve absolute rating values. As mentioned before, for the algorithm, the relative rankings are more important and we aim to keep a control on this based on the defined RMSE and MAE threshold.
2. **Right vs. correct**: We would be ready to compromise correct recommendations for right recommendations. Correct recommendations would be the ones calculated without factoring in serendipity. Right recommendations might not include all the correct recommendations but would have a higher relevance. We aim to explore this factor more in Phase II.

Dataset

We are using Book - Crossing Dataset. The book crossing is a medium to share your books with strangers via control release and track the book where it goes using journal entries. This dataset contains 278,858 users providing 1,149,780 ratings (explicit / implicit) about 271,379 books. Rating scale was 1 through 10.

Cleaning:

- We remove all implicit ratings marked with zero to avoid confusion.
- There are some books which have same title but different ISBN numbers. So we removed multiple ISBNs with one for a book .

- Number of unique books in the ratings file is more than number of unique books in the books csv describing each book. As a result we take a inner join on both the dataframe to include only those books which are in the books.csv catalog.

These tasks are executed in the Cleaning section of the CF-DATA Notebook.

Sampling

As specified in the instructions document, we have to select a sample where users are less than 10000 and items <= 100. This was a challenging task owing to the sparsity and peculiar nature of the data.

The average number of books rated by the user was 5.63 and each book was rated by 2.56 users on average. Our task was to find a dense subset of data to work on. We selected top 100 rated items took intersection with records having top 20000 users. This resulted in a dataset with 100 items, 2517 users and 8242 data-points with sparsity of 96.72%.

To test the scalability of our models we sampled bigger datasets with 150, 200, 300 and 500 top rated items.

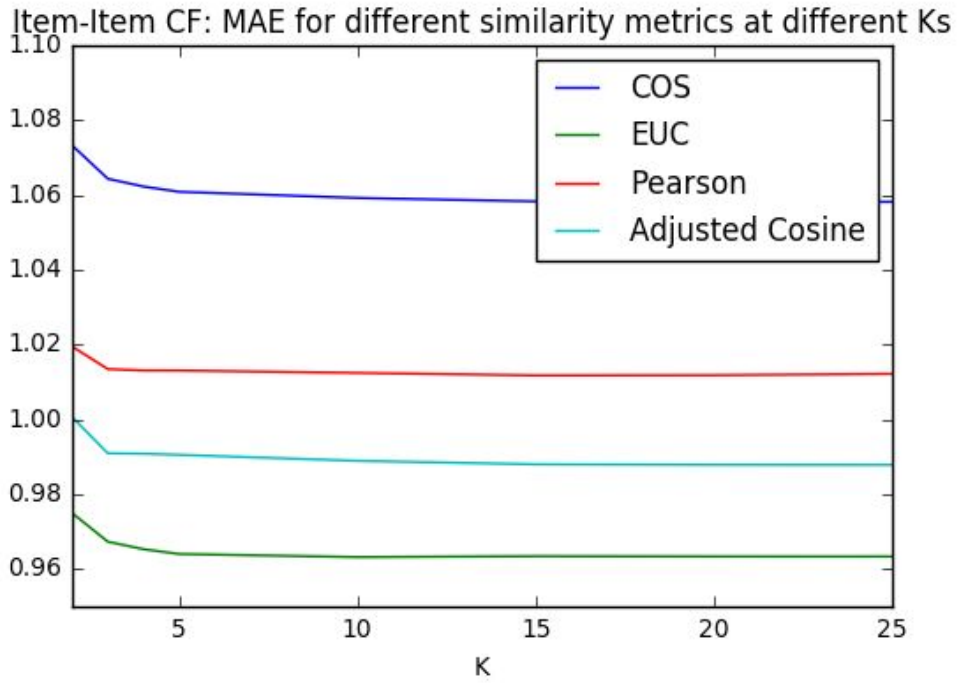
Memory Based Algorithm

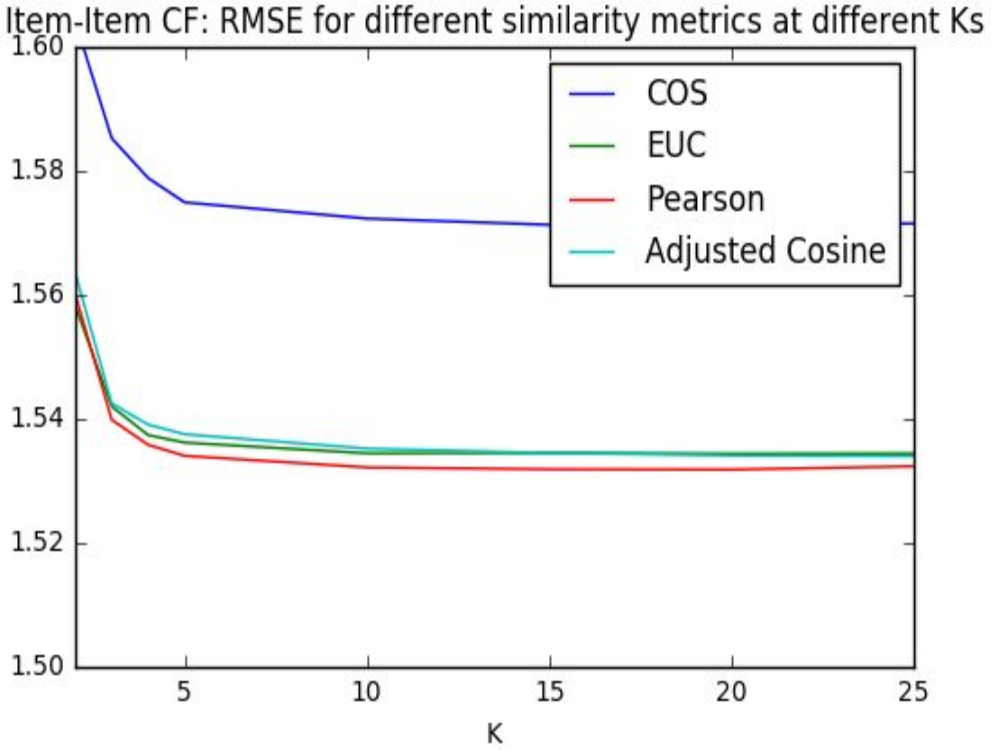
Item-Item CF Algorithm: Since the dataset had users who had on average rated only 3.3 books, we decided to perform *item-item CF algorithm* owing to the fact that more books were corated as compared to users who co-rated similar books.

Implementation from scratch: We implemented item-item CF algorithm from scratch with ability to consider similarity metrics like cosine similarity, pearson co-relation, euclidean distance (mean squared distance) and adjusted cosine similarity. A five fold cross validation was performed to evaluate metrics for the algorithms (MAE, RMSE).

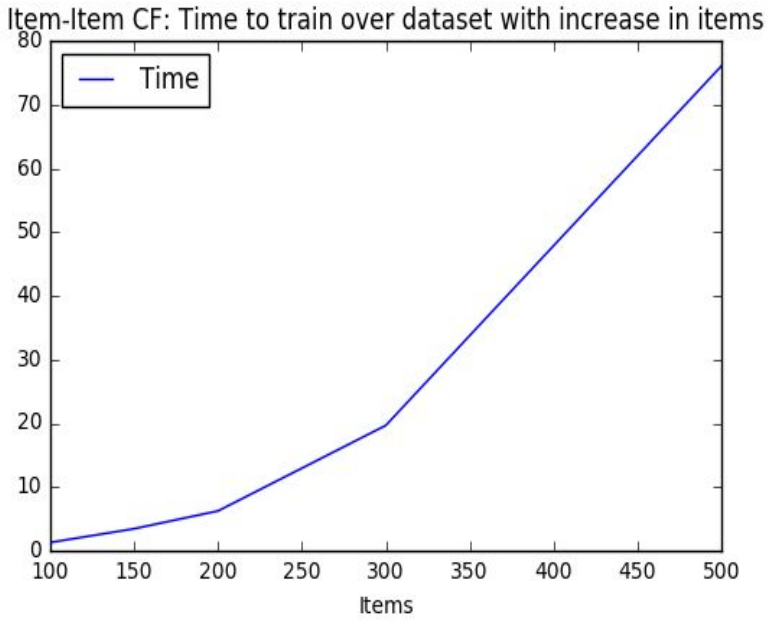
We obtained best MAE value of .963 using 'Euclidean distance' for k =10 (neighbours). Since, there was no major difference in MAE value at k = 5 and k = 10, we decided to use k =5 to decrease storage for online computations. Morever, k=5 seems logical as well since the average rated books per user is around 3.3. This was validated by performing grid search on k for values 2, 3, 4, 5, 10, 15, 20, 25. There was negligible change in MAE values with k>5.

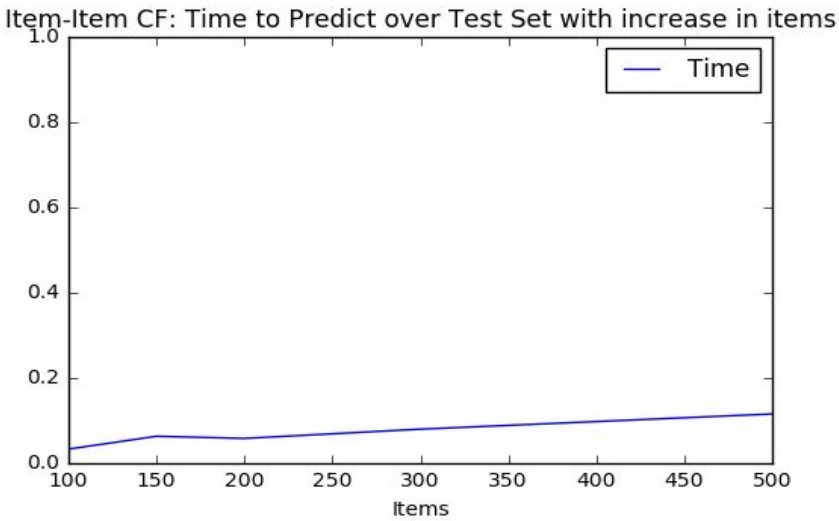
Note: For next sections, dataset with 100 items was used.





Scaling: As mentioned earlier we created 5 datasets with increasing number of items and user to test the scalability of our model. There were 100, 150, 200, 300 and 500 items respectively. Sparsity of the dataset decreases with increase in items. It was observed that the time to train increase exponentially with increase in number of items. We attribute it bigger item-item similarity matrix computation. However the time to predict ratings is similar in all cases suggesting usefulness in online recommendation computation. These computation were done using euclidean similarity metric.



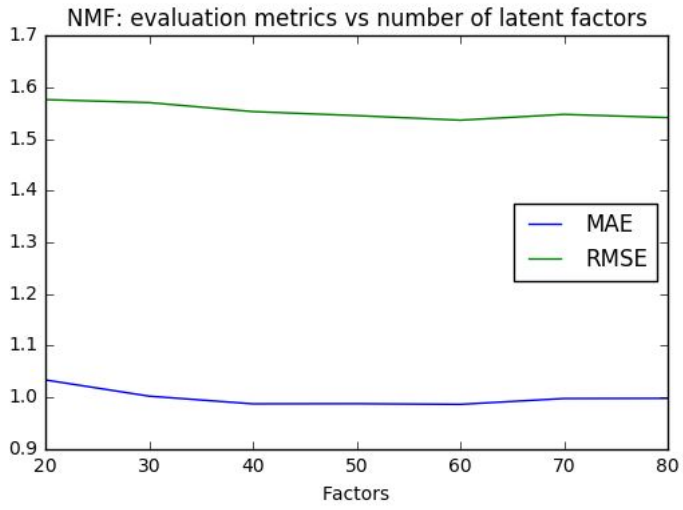


Coverage: We wanted check the coverage of our model, if it covers all books in the dataset. We ran experiments for neighbour sizes of 5, 8 10 and 15 and obtained coverage values of 92, 95, 99 and 100 % respectively. This suggests that our model covers wide range of items in the recommendation. However, this can be attributed to fewer number of items in our dataset.

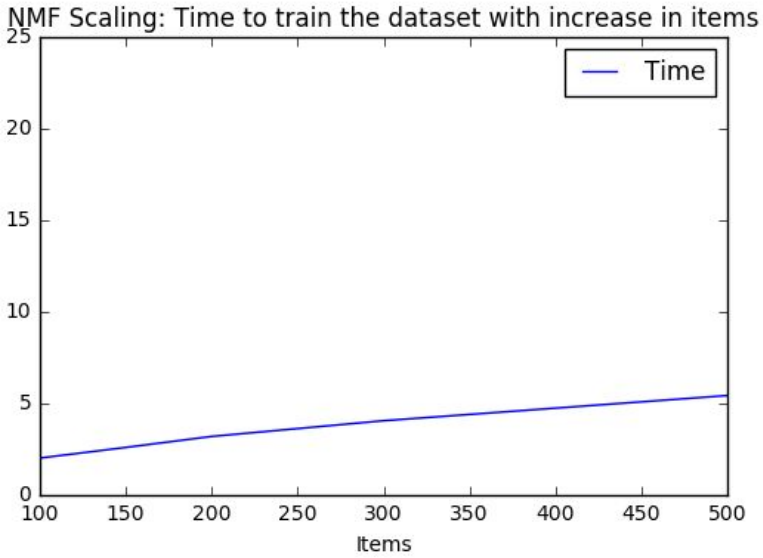
Model Based Algorithm

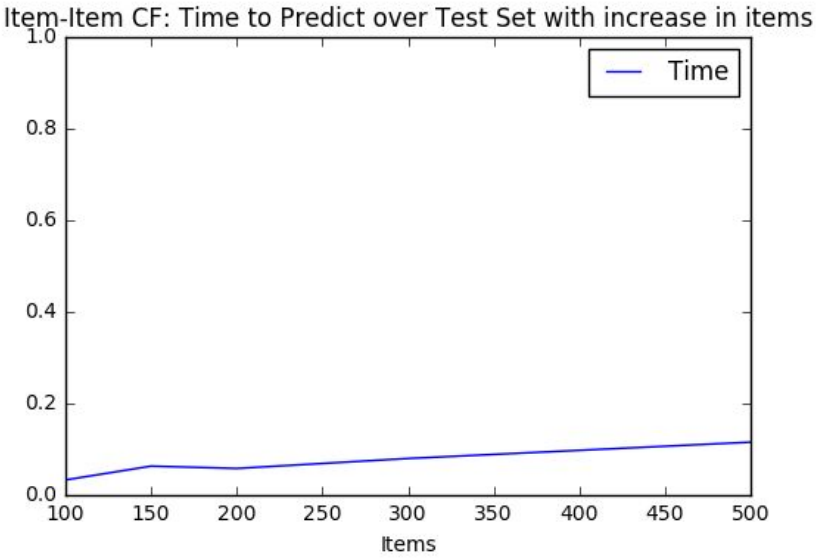
Non-Negative Matrix Factorisation: For model based approach we thought of exploring hidden features between the books and users using NMF. We implemented NMF from scratch but the evaluation metrics were not upto the mark, as a result we used python library scikit-surprise to execute NMF and tune its hyperparameters. We tuned for number of epochs between 40 to 60 in steps of 10, number of latent factors between 30 to 70 in steps of 10 and also regularised for user and book rating values.

For hyperparameter values : {'n_epochs': 60, 'reg_user': 0.001, 'n_factors': 50, 'reg_item': 3} we obtained best MAE value of 0.97. With suggested regularisation parameter above and varying number of latent factors we observe negligible change in MAE values beyond 50 latent factors. As a result, we use 50 latent factor for our model.



Scaling: Using the same dataset as mentioned above for scaling, we observe increase in time to train the model with increase in time, which is obvious but the increase in time is lower as compared to CF algorithm discussed above. The time take to predict barely changes with increase in number of items.





Coverage: The coverage of NMF is lower as compared to Item-Item CF Algorithm discussed above. We obtained coverage of 81, 87, 92 and 95% for neighbourhood size (k) 5, 8, 10 and 15 respectively. This suggests that NMF tends to repeat popular books as recommendation and some books are not at all recommended.

RESULTS:

All results are for a rating scale of 0-10

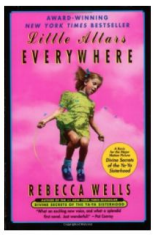
Algorithm	**Grid Search Values**	**Best MAE**
NMF (Our implementation)	k=2, 3, 4, 5, 10, 15, 20, 25	2.00
Naive Baseline		1.613
KNN with euclidean distance	k=2, 3, 4, 5, 10, 15, 20, 25	**0.965**
KNN with cosine similarity	k=2, 3, 4, 5, 10, 15, 20, 25	1.058
KNN with Pearson Correlation	k=2, 3, 4, 5, 10, 15, 20, 25	1.011
KNN with adjusted Cosine similarity	k=2, 3, 4, 5, 10, 15, 20, 25	0.987
NMF on Scikit Surprise	n_factors': [30, 40, 50, 60, 70], 'n_epochs': [40, 50, 60], 'reg_pu': [0.001, 0.1, 1], reg_qi': [0.1, 1, 3, 5]	0.97

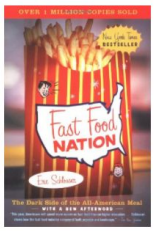
Visualization of recommendations

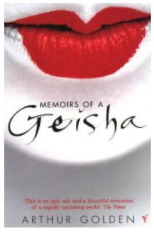
Reader 1 - Recommendation Results

```
User: 275638

 =x=x=x=x=x=x=x=x=x=x= BOOKS READ =x=x=x=x=x=x=x=x=x=x=
```

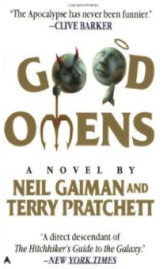


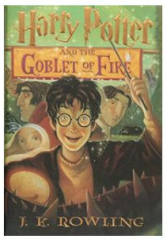




As can be seen from this figure, our user is someone who likes an eclectic mix of fiction and non-fiction. This looks the profile of a reader who likes different genres, is opinionated and afraid to try something new. Certainly, shifting from the world of the intense and darkly hopeful Memoirs of a Geisha to that of Fast Food Nation is not easy. Let's see what our recommendation algorithm picked for this person.

=x=x=x=x=x=x=x=x=x=x= BOOKS RECOMMENDED =x=x=x=x=x=x=x=x=x=x=

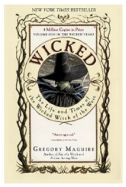


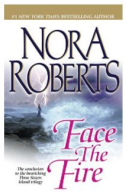


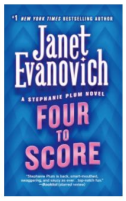
Voila! The suggestions include one bestseller, one-cult pop-fiction and the other a light tea-time or bedtime rom-com. This isn't a very bad recommendation after all. It offers some of the best books of the genres while keeping the recommendations very relevant given both the user's history and the correlation of books (item-item).

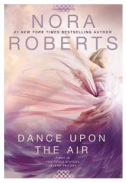
Reader 2 - Recommendation Results

=x=x=x=x=x=x=x=x=x=x= BOOKS READ =x=x=x=x=x=x=x=x=x=x=



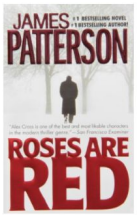


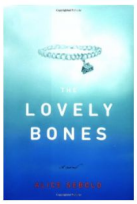


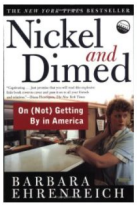


This looks the profile of a reader who, unlike Reader 1, prefers a single genre - something like romance, drama and fiction. Let’s see what our recommendation system as churned out:

=x=x=x=x=x=x=x=x=x=x= BOOKS RECOMMENDED =x=x=x=x=x=x=x=x=x=x=







The recommendations are very much in line with the preferences of the reader - Lovely Bones is a painful story and that has streaks of hope. Nickel and Dimed is another story of struggle and falls very much into the category of the books that the user already likes.

Conclusion and recommendation to business

We saw that our KNN implementation performs well in terms of MAE and also has good coverage as well as serendipity. Putting this model as it is in production isn't recommended because it takes very high time for scoring on large datasets. To solve this problem, we can either use approximate nearest neighbors or save and update distance matrices frequently.

To improve our current results, we plan on using user and book features. These features will largely come from the genre, author, synopsis and rating of the book. We will also use word embedding to capture information from text data. We propose implementation of a hybrid model for Phase II that collectively overcomes the shortcomings of individual models.