Goodreads Recommendation System

By Puneeta Chaturvedi

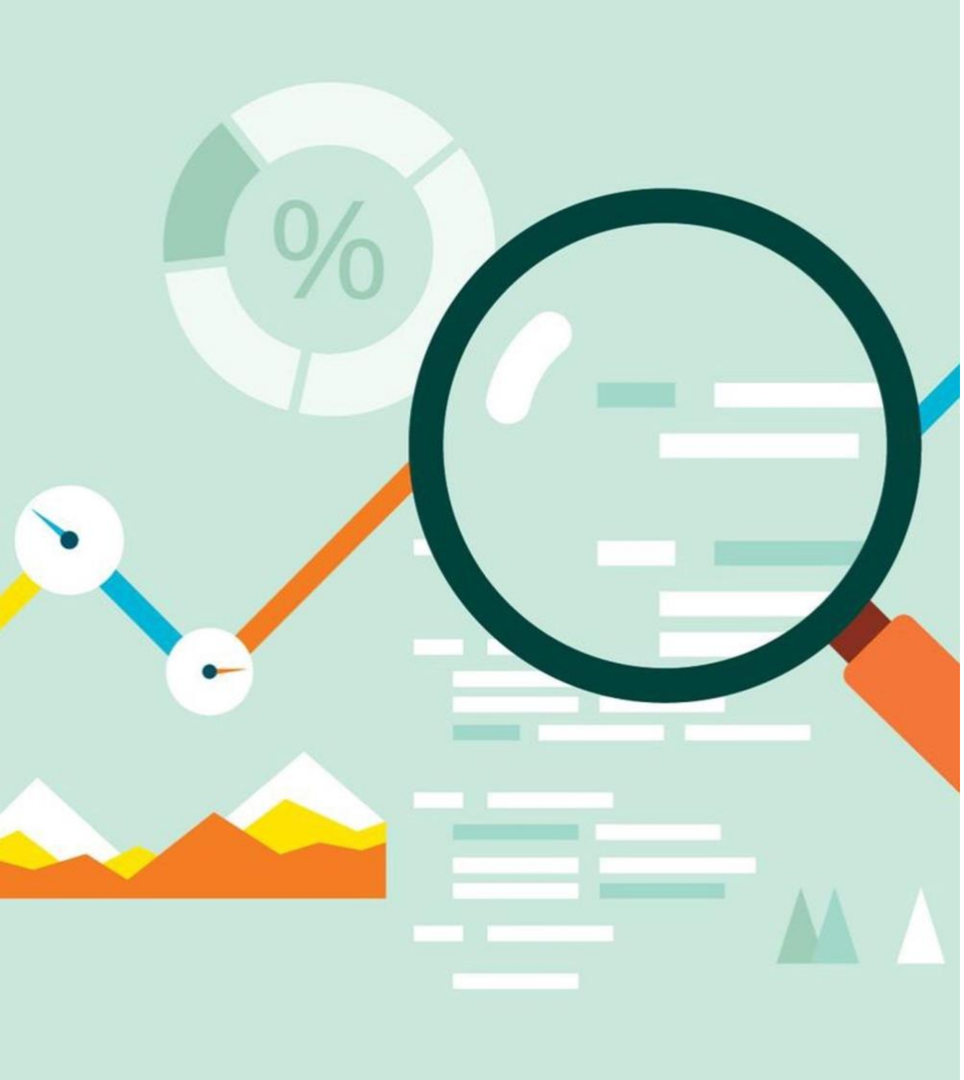
Problem Statement

- Our goal is to build a Recommendation System for Goodreads
  - A review aggregator for novels
- Main Problem: Given a user's read history, recommend to the user a book they would like.
  - Like is measured by the rating the user gives (from 0 to 5)

The Data

3 Main Sources

- Shelf Data: The shelf a user put a book in. Comparable to tags.
  - One user can put many tags on one book
- Review Data: Text of review and time associated with each one
  - Massive range. Some of thousands of reviews, others have a handful.
  - Each Review has an associated time and number of votes (likes) it received
- Interaction Data: Ratings by users on books
  - A Rating from 0-5
  - Includes timestamp
  - 22 Million Ratings in the dataset



Problems with and Preprocessing Data

- Problems
  - Massive Dataset. Interactions alone are 11 GB *compressed*

○ Would need expensive VM to process on

○ So we use sample: a subset of books that are Fantasy/sci-fi only:

○ Still 700K books, 22m interactions

- Preprocessing
  - Interactions matrix: row i, column j tells us the rating user i gave book j

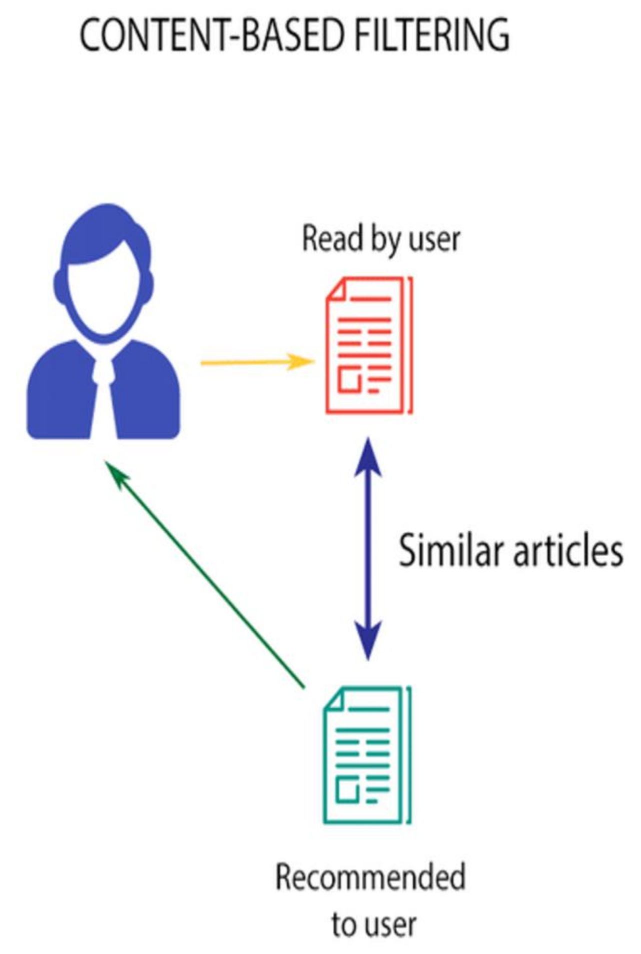
○ Tag Matrix: A TF_IDF matrix of the tags

○ Review Matrix: TF_IDF matrix of at most 10 reviews per book

Clustering (Part 1)

Content Based Filtering

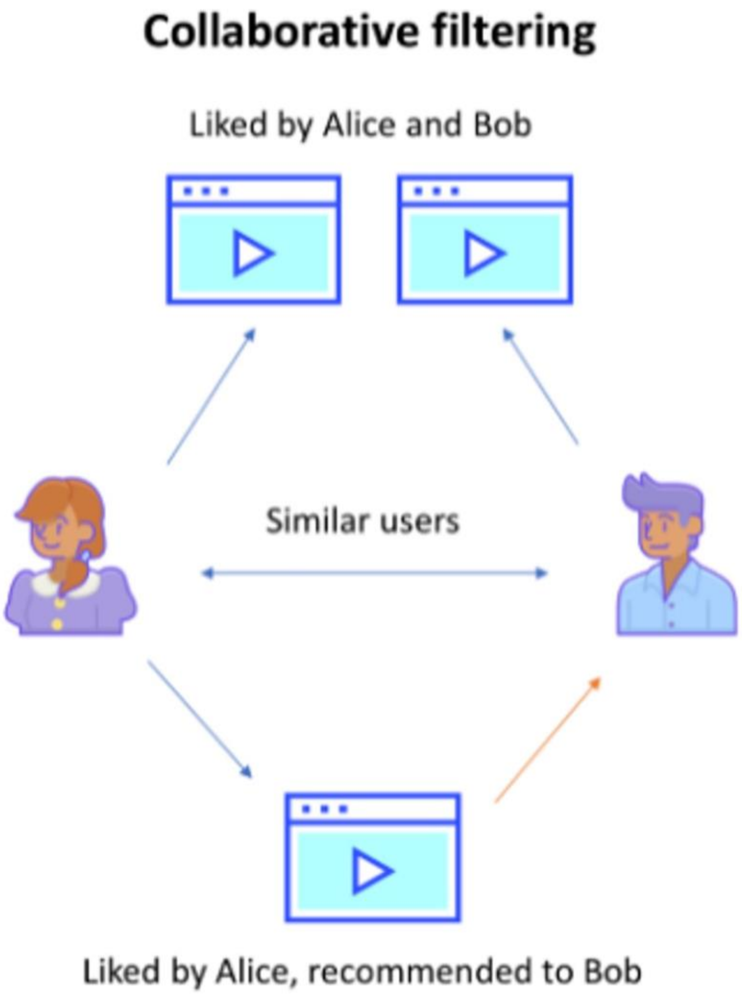
- Focus on Tag Matrix and Review Matrix
- Use clustering to group together
    - TF-IDF first
    - Dimensionality Reduction
        - 677000 to 300 with 83% variance explained
    - K-Means/GMM
    - Elbow Method to select clusters
- Hard to evaluate - can see entropy and use logic
- Good to use in “cold-start” case



Clustering (Part 2)

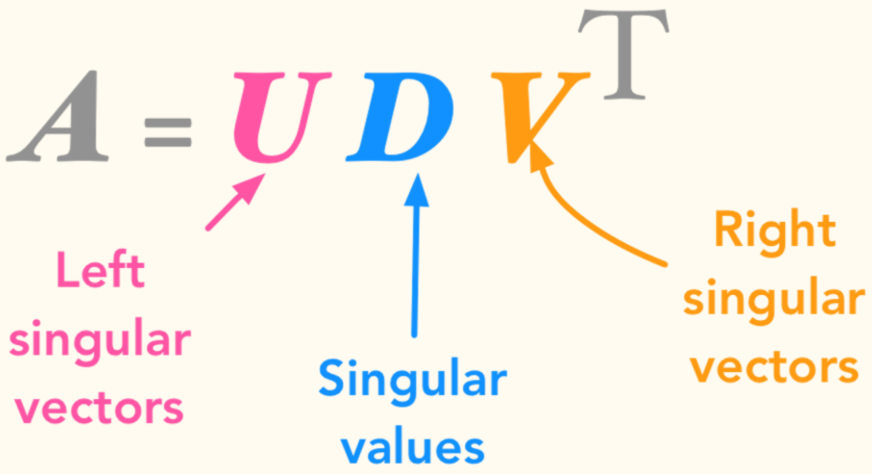
Collaborative Based Filtering:

- Focused Primarily on Interactions matrix
  - Sparse, high dimensional = curse of dimensionality
- K-Means
  - Poor Results. Elbow not appearing
- Solution: Use cosine similarity & each user is a cluster center
  - Sample users to find users similar within tolerance. This is our cluster
  - Build confidence interval for rating for each book for cluster
  - Recommend most confident ones
- Still hard to evaluate



Singular Value Decomposition

- We use Simon Funk’s SVD algorithm
  - Huge speedup compared to normal SVD. We use full dataset here.
- A is interaction matrix decomposed into:
  - U: NxN matrix of user
  - Diagonal Matrix of singular values
  - V: MxM matrix of all items
- Lets us very easily predict preference of user i for item j:
  - Dot product of row i and column j, scaled by diagonals
- MAE: 0.88
  - Very good overall



Dimension Reduction

- Need to use dimension Reduction to deal with the sparse matrix issue
- TruncatedSVD was employed in the process
- Reduce dimension to 100 to make most algorithms run quicker and will not represent too big of an information loss

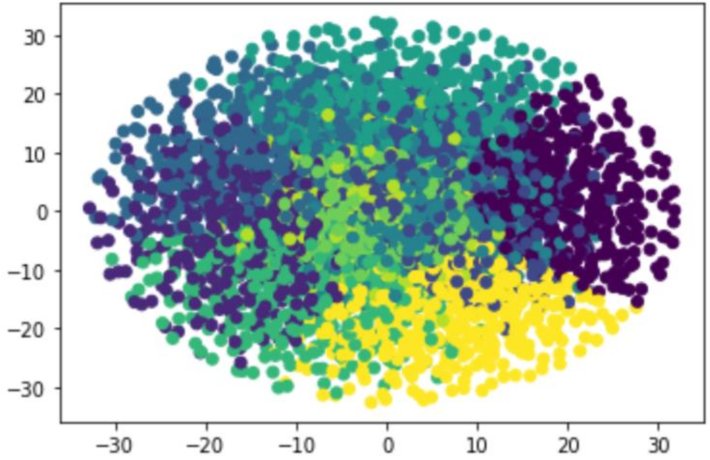


Fig. 5. TSNE dimension reduction

Supervised Model

Matrix B:MxN where M is the number of the book and N is number of unique tag

Matrix I:MxN where M is the number of users and N is the number of books

Matrix R: MxN where M is the number of books and N is the unique vocab

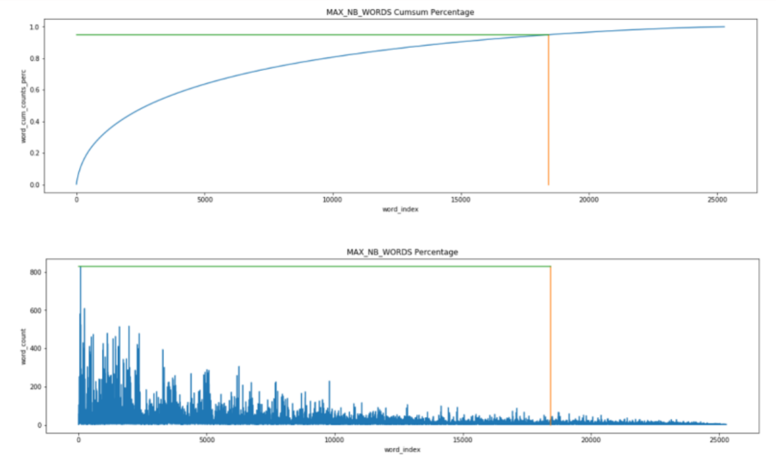
Independent Variables: The tags, the reviews, users' own history

Dependent Variables: The rating that the user gave

Data generation for Deep Learning

- Generate the reading book id for this user based on timestamp('read at')
- With sequence samples cut into the window size of 20, we will predict the next book the user would probably pick
- 95% of higher frequent book_id was chosen
- 18412 books in our prediction list
- 254388 training samples

	X	y
0	[888, 784, 588, 584, 580, 557, 865, 762, 381, ...	891
1	[784, 588, 584, 580, 557, 865, 762, 381, 509, ...	807
2	[588, 584, 580, 557, 865, 762, 381, 509, 377, ...	803
3	[584, 580, 557, 865, 762, 381, 509, 377, 646, ...	251
4	[580, 557, 865, 762, 381, 509, 377, 646, 272, ...	226



Skip-Gram and KNN

- With training the distributed representation(embedding) for book id by skip-gram model
- Recommend the top 10 similar books with trained word embedding (cosine similarity)

- Parameters setting
  - Negative sampling size: 5
  - Window size: 20

	X	y
0	[888, 784, 588, 584, 580, 557, 865, 762, 381, ...	891
1	[784, 588, 584, 580, 557, 865, 762, 381, 509, ...	807
2	[588, 584, 580, 557, 865, 762, 381, 509, 377, ...	803
3	[584, 580, 557, 865, 762, 381, 509, 377, 646, ...	251
4	[580, 557, 865, 762, 381, 509, 377, 646, 272, ...	226

```
1  wv_model.most_similar('784')
```

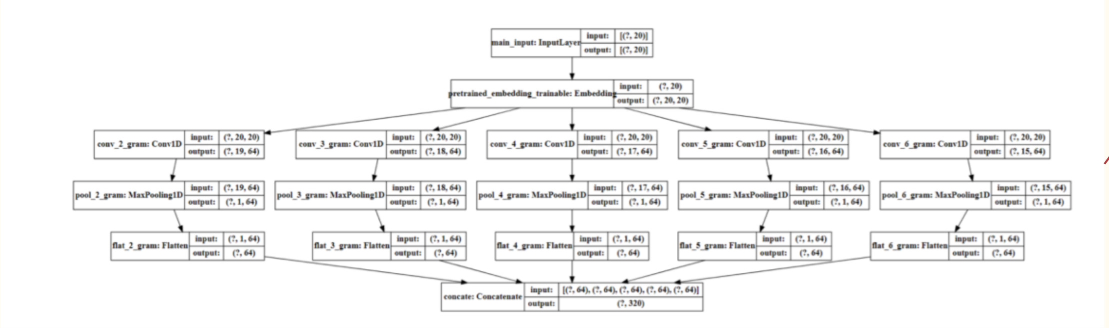
executed in 28ms, finished 20:57:57 2020-12-11

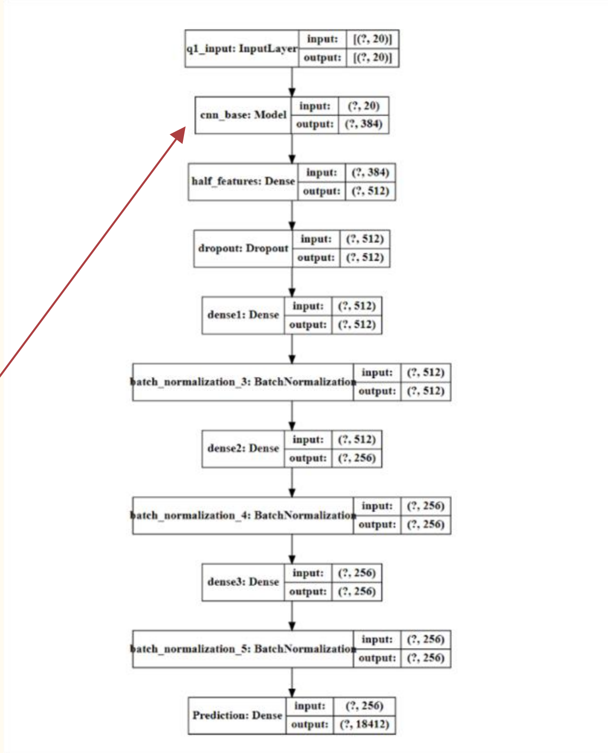
```
C:\Users\Administrator\Anaconda3\envs\py810\lib\site-
  """Entry point for launching an IPython kernel.

[('17274', 0.9996403455734253),
 ('46750', 0.999634861946106),
 ('741', 0.9996309280395508),
 ('15359', 0.9996287822723389),
 ('8381', 0.9996236562728882),
 ('97345', 0.9996221661567688),
 ('47857', 0.9996192455291748),
 ('26309', 0.9996187090873718),
 ('183831', 0.999618649482727),
 ('63200', 0.9996184706687927)]
```

Multichannel-CNN Structure

- NUM_FILTERS=64
- EMBEDDING_DIM = 20
- FILTER_SIZES= [2,3,4,5,6]
- BATCH_SIZE=128





Further improvement

- Pretrain the embedding and shrink the parameter size
- Enlarge the training set
- Build more effective evaluation method
- Combined more features from tasks 1 and 2, like the attributes of the book or users
- Try others model structure