



NEURAL COLLABORATIVE FILTERING BASED RECOMMENDATION SYSTEM

A Report for the Evaluation 3 of Project 2

Submitted by

ANUPAMA PANDEY

(1613101163/16SCSE101621)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Under the Supervision of

MS. GARIMA PANDEY

APRIL/MAY-2020



SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report “NEURAL COLLABORATIVE FILTERING
BASED RECOMMENDATION SYSTEM” is the bonafide work of
“ANUPAMA PANDEY (1613101163)” who carried out the project work under
my supervision.

SIGNATURE OF HEAD

Dr. MUNISH SHABARWAL,
PhD (Management), PhD (CS)
Professor & Dean,
School of Computing Science &
Engineering

SIGNATURE OF SUPERVISOR

Ms. Garima Pandey
Assistant Proffesor
School of Computing Science
and Engineering

ABSTRACT

Recommender System is a commercial purpose system that decides what should be recommended to the user . It is used in many fields. It helps to create relationship between user, product and identifies the most appropriate product for that user. In recent years, deep neural networks have yielded immense success on speech recognition, computer vision and linguistic communication processing. However, the recommender systems has gotten relatively less scrutiny from deep neural networks. DNN uses complex mathematical modelling for finding the output of an input. During this work, we try to develop techniques supported neural networks to tackle the key problem in recommendation — collaborative filtering — on the basis of implicit feedback.

By replacing the real product with a neural design which will take in a subjective capacity from information, we present a general framework named NCF, short for Neural network based Collaborative Filtering. NCF is nonexclusive and will express and sum up matrix factorization under its framework.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
1.	Abstract	iii
1.	Introduction	1
	1.1 General	1
	1.2 Classification	2
	1.3 Approach	7
3.	Existing System	18
4.	Proposed System	20
5.	Implementation or architectural diagrams	26
6.	Output/Result/Screenshot	34
7.	Conclusion	35
8.	References	36

INTRODUCTION

Recommender systems

Recommender systems aim to predict users' interests and recommend product items that quite likely are interesting for them. They are among the most powerful machine learning systems that online retailers implement in order to drive sales.

Data required for recommender systems stems from explicit user ratings after watching a movie or listening to a song, from implicit search engine queries and purchase histories, or from other knowledge about the users/items themselves.

Sites like Spotify, YouTube or Netflix use that data in order to suggest playlists, so-called **Daily mixes**, or to make **video recommendations**, respectively.

Why do we need recommender systems?

Companies using recommender systems focus on increasing sales as a result of very personalized offers and an enhanced customer experience.

Recommendations typically speed up searches and make it easier for users to access content they're interested in, and surprise them with offers they would have never searched for.

What is more, companies are able to gain and retain customers by sending out emails with links to new offers that meet the recipients' interests, or suggestions of films and TV shows that suit their profiles.

The user starts to feel known and understood and is more likely to buy additional products or consume more content. By knowing what a user wants, the company gains competitive advantage and the threat of losing a customer to a competitor decreases.

Providing that added value to users by including recommendations in systems and products is appealing. Furthermore, it allows companies to position ahead of their competitors and eventually increase their earnings.

How does a recommender system work?

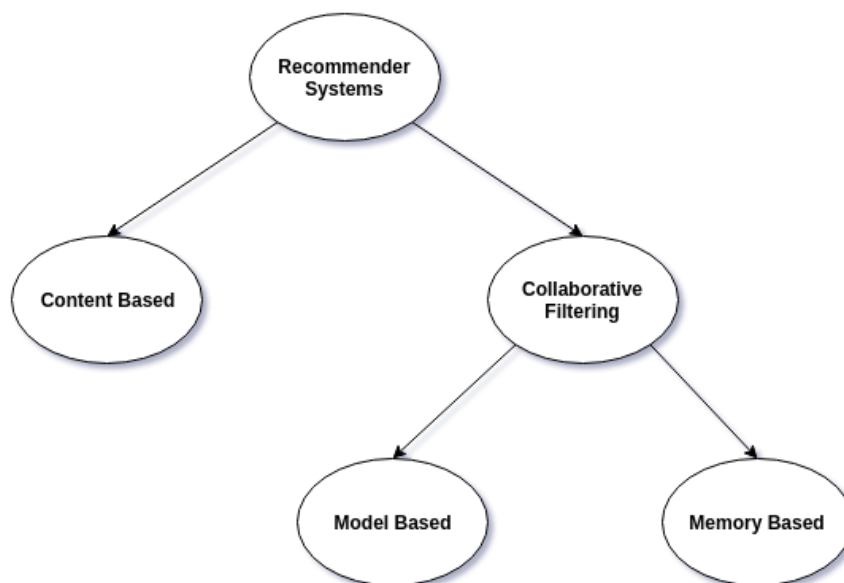
Recommender systems function with two kinds of information:

1. *Characteristic information*. This is information about items (keywords, categories, etc.) and users (preferences, profiles, etc.).
2. *User-item interactions*. This is information such as ratings, number of purchases, likes, etc.

Based on this, we can distinguish between three algorithms used in recommender systems:

- *Content-based* systems, which use characteristic information.
- *Collaborative filtering* systems, which are based on user-item interactions.

- *Hybrid systems*, which combine both types of information with the aim of avoiding problems that are generated when working with just one kind.



Content-based systems

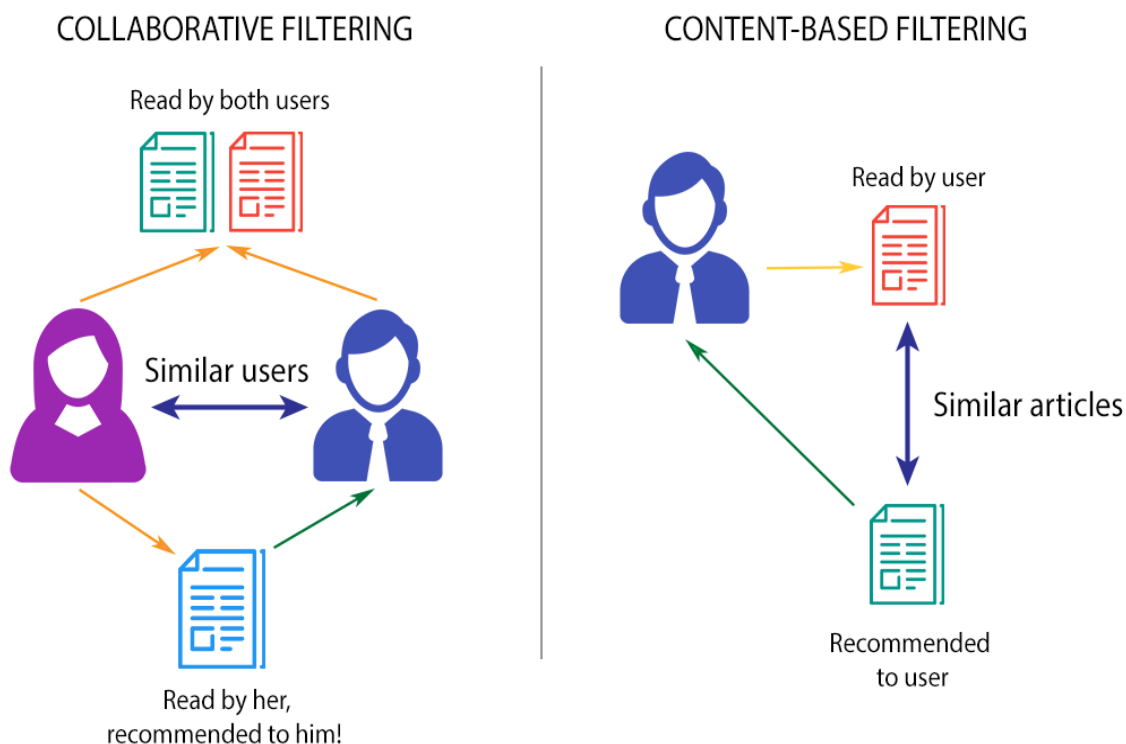
These systems make recommendations using a user's item and profile features. They hypothesize that if a user was interested in an item in the past, they will once again be interested in it in the future. Similar items are usually grouped based on their features. User profiles are constructed using historical interactions or by explicitly asking users about their interests. There are other systems, not considered purely content-based, which utilize user personal and social data.

One issue that arises is making obvious recommendations because of excessive specialization (user A is only interested in categories B, C, and D, and the

system is not able to recommend items outside those categories, even though they could be interesting to them).

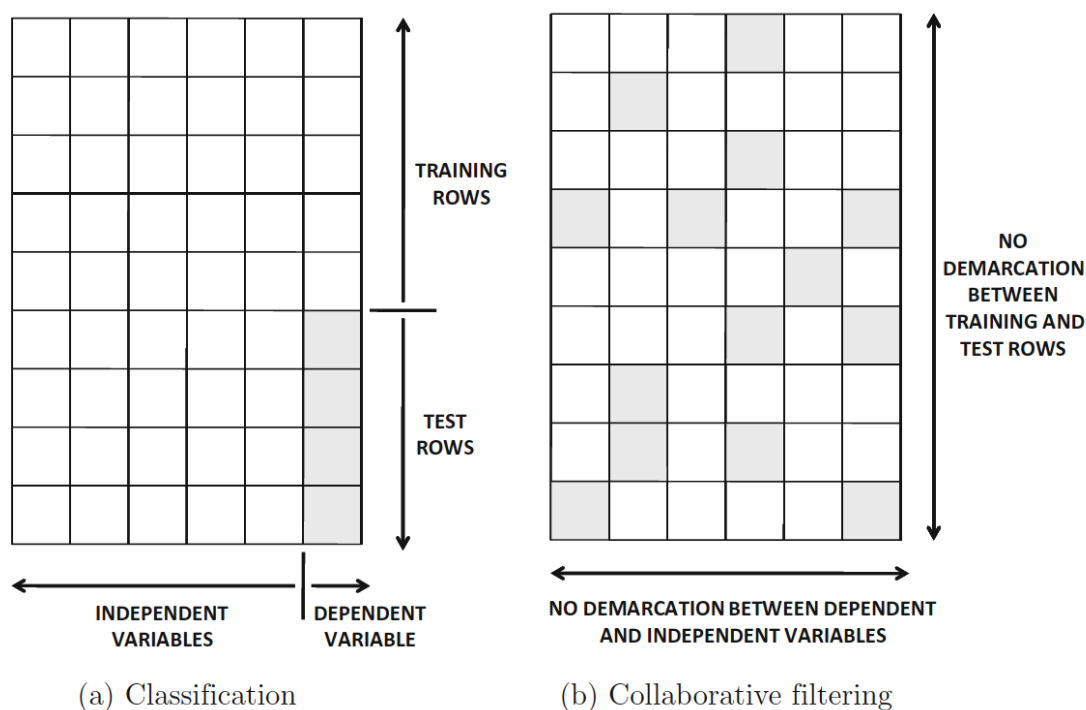
Collaborative filtering systems

Collaborative filtering is currently one of the most frequently used approaches and usually provides better results than content-based recommendations. Some examples of this are found in the recommendation systems of Youtube, Netflix, and Spotify.



These kinds of systems utilize user interactions to filter for items of interest. We can visualize the set of interactions with a matrix, where each entry $(i,$

$j)(i,j)$ represents the interaction between user ii and item jj . An interesting way of looking at collaborative filtering is to think of it as a generalization of classification and regression. While in these cases we aim to predict a variable that directly depends on other variables (features), in collaborative filtering there is no such distinction of feature variables and class variables. Visualizing the problem as a matrix, we don't look to predict the values of a unique column, but rather to predict the value of any given entry.



In short, collaborative filtering systems are based on the assumption that if a user likes item A and another user likes the same item A as well as another item, item B, the first user could also be interested in the second item. Hence, they aim to predict new interactions based on historical ones. There are two types of methods to achieve this goal: *memory-based* and *model-based*.

Memory-based

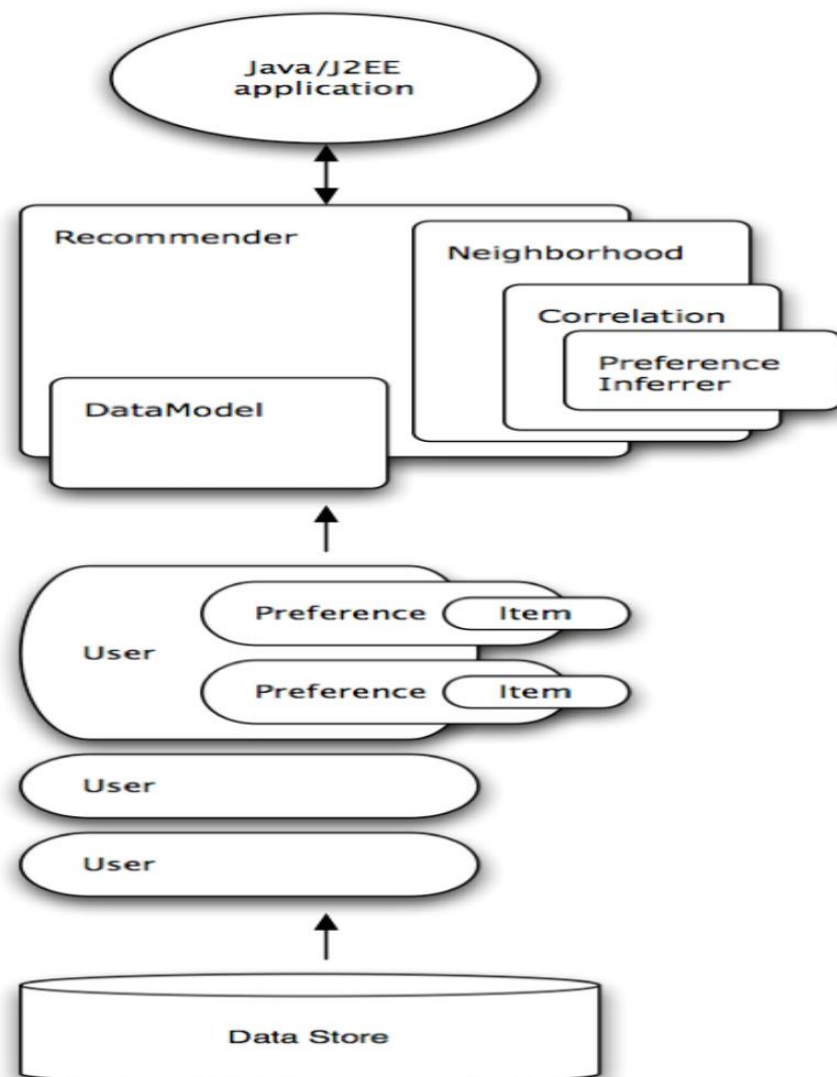
There are two approaches: the first one identifies clusters of users and utilizes the interactions of one specific user to predict the interactions of other similar users. The second approach identifies clusters of items that have been rated by user A and utilizes them to predict the interaction of user A with a different but similar item B. These methods usually encounter major problems with large sparse matrices, since the number of user-item interactions can be too low for generating high quality clusters.

Model-based

These methods are based on machine learning and data mining techniques. The goal is to train models to be able to make predictions. For example, we could use existing user-item interactions to train a model to predict the top-5 items that a user might like the most. One advantage of these methods is that they are able to recommend a larger number of items to a larger number of users, compared to other methods like memory-based. We say they have large *coverage*, even when working with large sparse matrices.

Alternative approaches using engineering algorithms:

- Taste: Taste is a flexible, fast collaborative filtering engine for Java. It takes the users' preferences for items and The engine takes users' preferences for items ("tastes") and recommends other similar items (Sean, 2008).



1. Vogoo: Vogoo is a php based collaborative filtering and recommendation library. It recommends items to users, which matches their tastes. It calculates similarities between users and creates communities based on them. The figure below shows the results of using vogoo to generate similar taste sharing users and recommendations made by the most similar users (Droux, 2008).

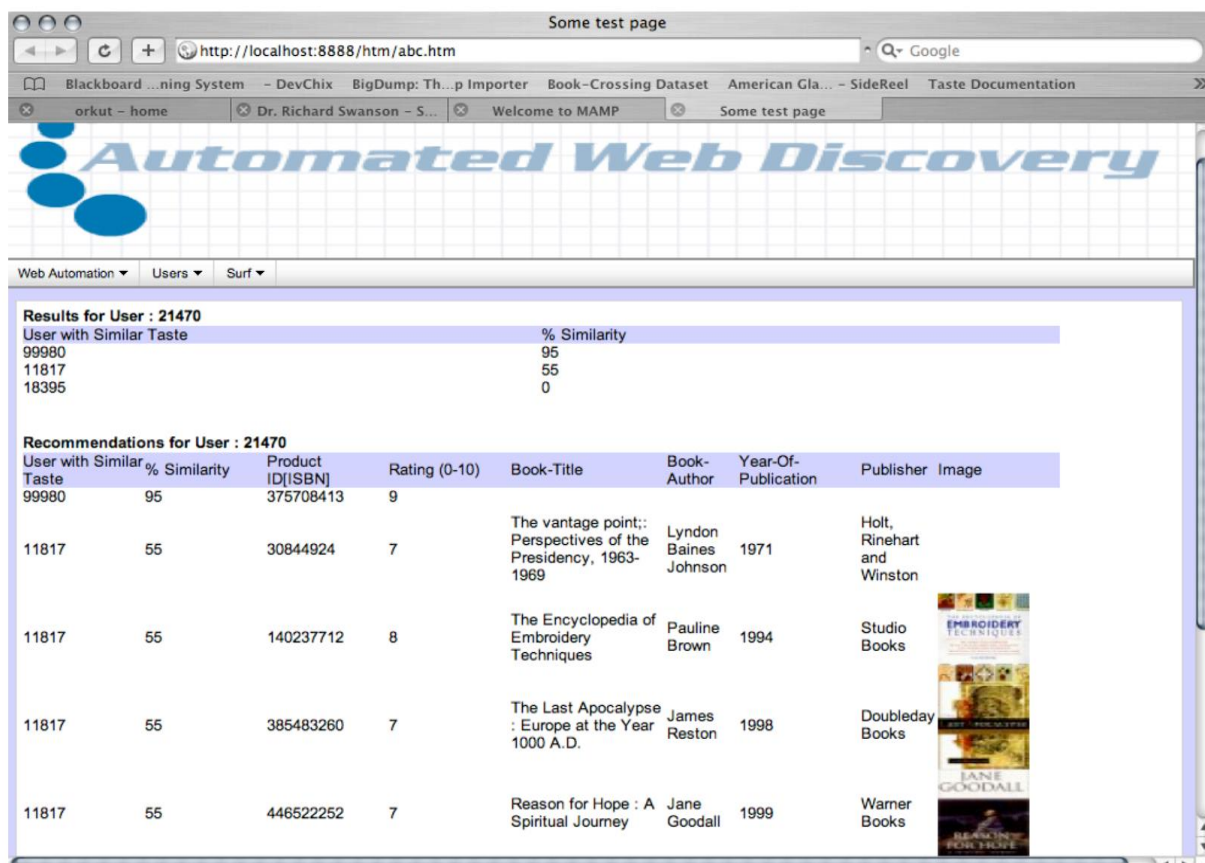


Figure 3: Vogoo implementation

- Fuzzy Logic: Here I tried to make use of fuzzy logic to calculate similar users. We use a hybrid approach (Christakou, 2005) and accept inputs from the users in three forms:

- Numeric rating between 0.0 – 1.0
- Three rating for context between 0.0 – 1.0
- Tags (free tagging)

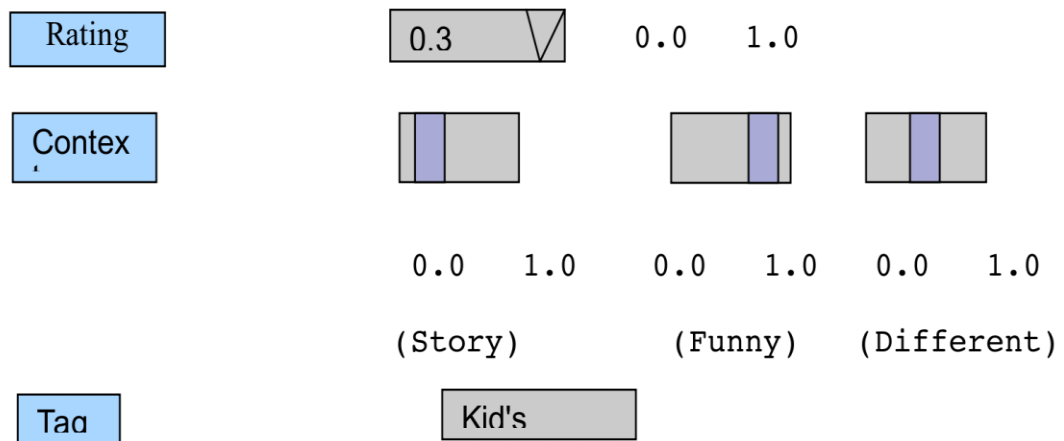


Figure 4: Movie rating parameters

In order to calculate similar users for the active user we first reduce the three ratings for any movie to a single movie rating between zero and one, after that we generate a user/movie matrix(Pereira, 2006) as shown in the following fig

		m1	m2	m3	m4
users	a	0.6	0.8	0.3	0.4
	b	0.2	0.7	0.4	0.6
	c	0.1	0.8	0.2	0.5
	d	0.3	0.6	0.1	0.3

Figure 5: User/Movie ratings matrix(Pereira, 2006)

Once the (user/movie rating) matrix is generated we apply fuzzy Context t Tag s
Kid's Movie Rating 0.3 17 logic to it and generate a user similarity matrix as
shown in the figure:

		a	b	c	d	e	f	g
Users	a	1	.8	0	.4	0	0	0
	b	.8	1	0	.4	0	0	0
	c	0	0	1	0	1	.9	.5
	d	.4	.4	0	1	0	0	0
	e	0	0	1	0	1	.9	.5
	f	0	0	.9	0	.9	1	.5
	g	0	0	.5	0	.5	.5	1

Figure 6: User similarity matrix(Klir, 1988)

The above figure shows the user similarity matrix in which the ratings between different users are listed. Now in order to calculate similar users we define to be a partition set where, $\alpha > 0$ for example let $\alpha = \{0.4, 0.5, 0.8, 0.9, 1.0\}$. Now for every value of α we will get a similar user group satisfying the condition example: $(ab=0.8) > (\alpha=0.4)$ so user 'a' and user 'b' are related (Klir, 1988). This is shown in the figure below

Currently used approach

User Request: - User makes a request for recommendation by clicking on the recommendation menu. User is asked to provide contextual information.

Server: - The information provided by the user is send to the server. The server is composed on 2 sub engines: user based collaborative filtering engine, and context based engine. The server sends users request to both the sub engines.

User based collaborative filtering engine: - calculates similar 19 users based on the numerical ratings of common items rated by the active users and other users of the system. The system achieves this by making user of the Pearson's correlation.

- **Pearson's Correlation:** is a way to find out similar users. The correlation is a way to represent data sets on graph. Pearson's correlation is x-y axis graph where we have a straight line known as the best fit as it comes as close to all the

items on the chart as possible. If two users rated the books identically then this would result as a straight line (diagonal) and would pass through every books rated by the users. The resultant score in this case is 1. The more the users disagree from each other the lower their similarity score would be from 1.

Pearson's Correlation helps correct grade inflation. Suppose a user 'A' tends to give high scores than user 'B' but both tend to like the book they rated. The correlation could still give perfect score if the differences between their scores are consistent.

Algorithm: The algorithm first finds all the common books rated by user 'A' and user 'B'. It then finds out the sums and sum of the squares of the ratings for both the users. It then finds the sum of the products of their ratings. These scores are then used to find out Pearson's correlation.

$$sim(c, p) = \frac{\sum_{j \in I_c} (r_{c,j} - \bar{r}_c)(r_{p,j} - \bar{r}_p)}{\sqrt{\sum_{j \in I_c} (r_{c,j} - \bar{r}_c)^2} \sqrt{\sum_{j \in I_c} (r_{p,j} - \bar{r}_p)^2}}$$

Figure 8: Pearson's Correlation formula.

Context Engine: - was initiated with an item based collaborative filtering approach example: Amazon related books etc. The item based collaborative filtering approach was build using Pearson's correlation, but instead of

calculating similarity between users here we calculated similarity between items. The results were good but it did not meet the goals set for the context-based engine initially. The system did not give good results due to lack of ratings, the system did not fill up the deficiencies of the CF based engine, the system did not do justice to the word 'related' items, because of all these reasons the below approach was followed. This engine makes use of contextual information provided by the user, synonyms, meta data about the products to find recommended items.

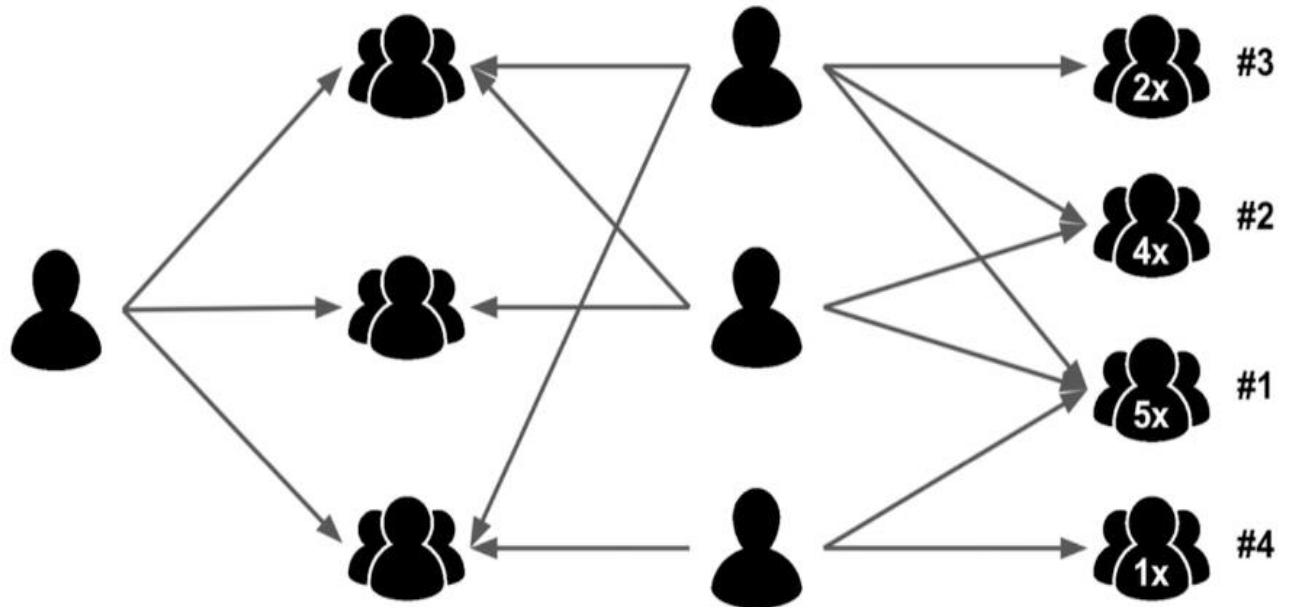
- The system first asks the user to provide context information example: author, publisher, and ISBN, and tags. The system does not expect the user to provide the complete 21 author, ISBN; publisher name example 'oxf' could be typed in as part of publisher name. The system then asks the user to type any free keywords. Once the user clicks the submit button. The information is first fed into the query engine, which makes use of the context information to narrow down the search results. The free keywords are fed to the Synonym Finder engine, which makes use of screen scraping techniques to find different senses of the entered keywords. This is done to find out the correct sense of the keyword used. All the results of the query parser (books) and Synonym Finder (senses) are then shown to the user. The user is then expected to see the results and if he/she is not yet satisfied, they can click on the 'refine' button, as soon as the refine button is clicked the results the Synonym Finder i.e. different senses

are fed to the query parser. Simultaneously a web service call is made to the Amazon Web Services to capture the editorial reviews of the books shown to the user earlier. Once this is done. The parser searches for these senses in the editorial reviews, if a match is found then the results (books) are shown in that category. The advantage of using this approach is that it helps to cover the disadvantages of the User based collaborative filtering engine like lack of user ratings, false ratings etc and deliver accurate predictions to the users.

Recommendation systems advise users on which items (movies, music, books etc.) they are more likely to be interested in. A good recommendation system may dramatically increase the number of sales of a firm or retain customers. For instance, 80% of movies watched on Netflix come from the recommender system of the company. Collaborative Filtering (CF) aims at recommending an item to a user by predicting how a user would rate this item. To do so, the feedback of one user on some items is combined with the feedback of all other users on all items to predict a new rating. For instance, if someone rated a few books, CF objective is to estimate the ratings he would have given to thousands of other books by using the ratings of all the other readers.

Collaborative filtering (CF) is a successful approach commonly used by many recommender systems. Conventional CF-based methods use the ratings given to items by users as the sole source of information for learning to make recommendation. However, the ratings are often very sparse in many

applications, causing CF-based methods to degrade significantly in their recommendation performance.



Collaborative Filtering

Although some recent work has employed deep learning for recommendation, they primarily used it to model auxiliary information, such as textual descriptions of items and acoustic features of musics. When it comes to model the key factor in collaborative filtering — the interaction between user and item features, they still resorted to matrix factorization and applied an inner product on the latent features of users and items.

By replacing the inner product with a neural architecture that can learn an arbitrary function from data, we present a general framework named NCF, short

for Neural network based Collaborative Filtering. NCF is generic and can express and generalize matrix factorization under its framework. To supercharge NCF modelling with non-linearities, we propose to leverage a multi-layer perceptron to learn the user–item interaction function. Extensive experiments on two real-world datasets show significant improvements of our proposed NCF framework over the state-of-the-art methods. Empirical evidence shows that using deeper layers of neural networks offers better recommendation performance.

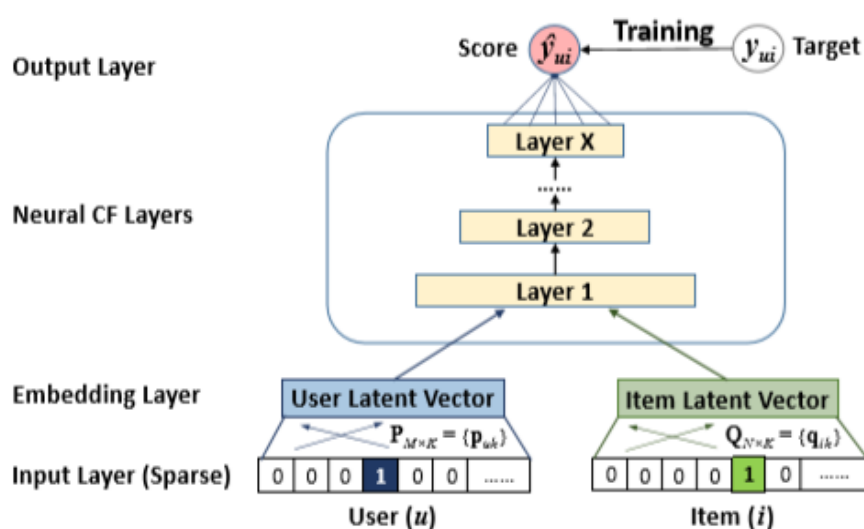


Figure 2: Neural collaborative filtering framework

Popularized by the Netflix Prize, MF has become the de facto approach to latent factor model-based recommendation. Much research effort has been devoted to enhancing MF, such as integrating it with neighbor-based models,

combining it with topic models of item content, and extending it to factorization machines for a generic modelling of features. Despite the effectiveness of MF for collaborative filtering, it is well-known that its performance can be hindered by the simple choice of the interaction function — inner product.

For example, for the task of rating prediction on explicit feedback, it is well known that the performance of the MF model can be improved by incorporating user and item bias terms into the interaction function¹. While it seems to be just a trivial tweak for the inner product operator, it points to the positive effect of designing a better, dedicated interaction function for modelling the latent feature interactions between users and items. The inner product, which simply combines the multiplication of latent features linearly, may not be sufficient to capture the complex structure of user interaction data.

Presenting a few “best” recommendations in a list requires a fine-level representation to distinguish relative importance among candidates with high recall. The ranking network accomplishes this task by assigning a score to each video according to a desired objective function using a rich set of features describing the video and user. The highest scoring videos are presented to the user, ranked by their score.

EXISTING SYSTEM

1. Substructure for Researcher Consideration Analysis

In figure 1, a hypothetical substructure is presented to analyze researcher consideration. Primarily, the data are collected from the researcher by using a sample abstract of research paper in which researcher presented an opinion about his experience with the research paper. Now the collected data are disorganized and needed some refinement before it can be evaluated.

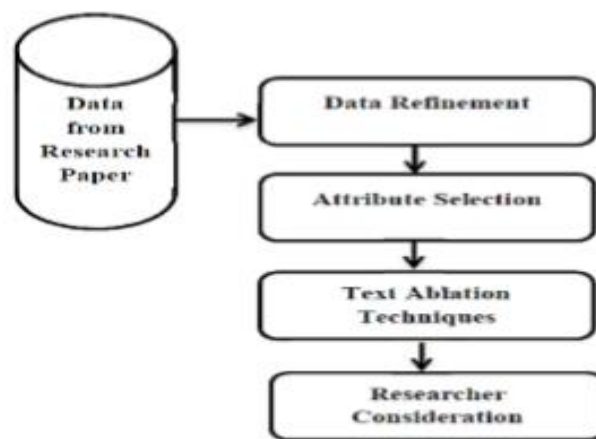


Figure 1.The Substructure for Researcher Consideration Analysis

Refinement phase is a dominant challenge and enormous time is exhausted during this phase. aforementioned textual refinement comprises purification steps, such as eliminate redundant characters, supersede special characters with spaces, eliminate stop words and word derive. From the purification data, attribute chosen is made and separated into numerical/categorical and textual attributes. The figure 1 a hypothetical substructure for researcher consideration, analysis. Researcher consideration analyzed in this segment is anticidently used in the next substructure to confer recommendations.

2. Substructure for Concept Clustering Based Approach

In figure 2, a hypothetical substructure for Concept clustering based approach is provided. For recommendations, another data set is gathered based on the sample abstract of research paper collected in the above-mentioned steps. In text mining, Concept dependant clustering intentions the sense of words/ phrase. Concept mining particularizes the role of words in the sense of the sentence, which indicates a copious adroit and sapient clustering. Concept probably a phrase or a conglomeration of a word which provides an evincive contribution to the text yet we can establish the variant concept in selfsame or other document, which provides identical or almost identical meaning. A collection of concept can be perceived as identical meaning, but varied word tokens. The clustering will be constituted based meaning of collection of concept [48].

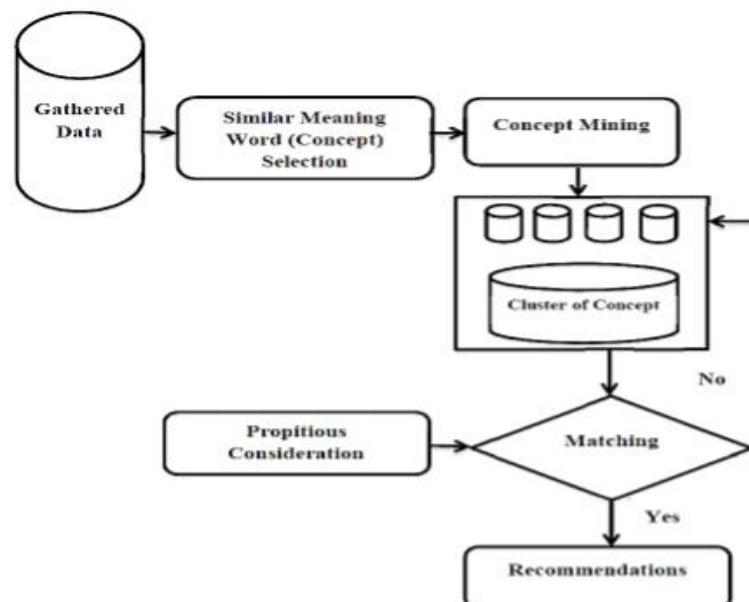


Figure 2.The Substructure for Concept Clustering Based Approach

PROPOSED MODEL

We first formalize the issue and talk about existing solutions for collaborative filtering with implicit feedback. We at that point shortly recapitulate the widely used MF model, featuring its restriction caused by using an input.

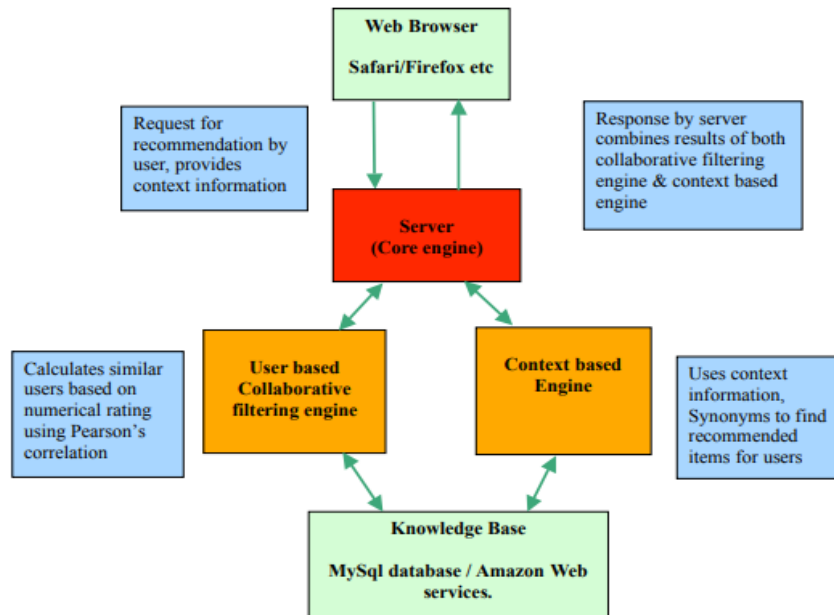


Figure 1: System Architecture

1. Learning from Implicit Data

Let M and N denote the amount of users and items, respectively. We define the user–item interaction matrix $Y \in \mathbb{R}^{M \times N}$ from users' implicit feedback as, fact

$$y_{ui} = \begin{cases} 1, & \text{if interaction (user } u, \text{ item } i) \text{ is observed;} \\ 0, & \text{otherwise.} \end{cases}$$

Here a value of 1 for y_{ui} shows that there's an communication between user u and item i ; be that as it may, it doesn't mean u really likes i . So also, a value of 0 doesn't really mean u doesn't care for i , it is that the user doesn't know about the item. This postures challenges in learning from implicit data, since it gives just noisy signals about users' inclination. While observed entries least reflect users' interest on items, the imperceptible entries might be simply missing information and there's a characteristic shortage of negative criticism

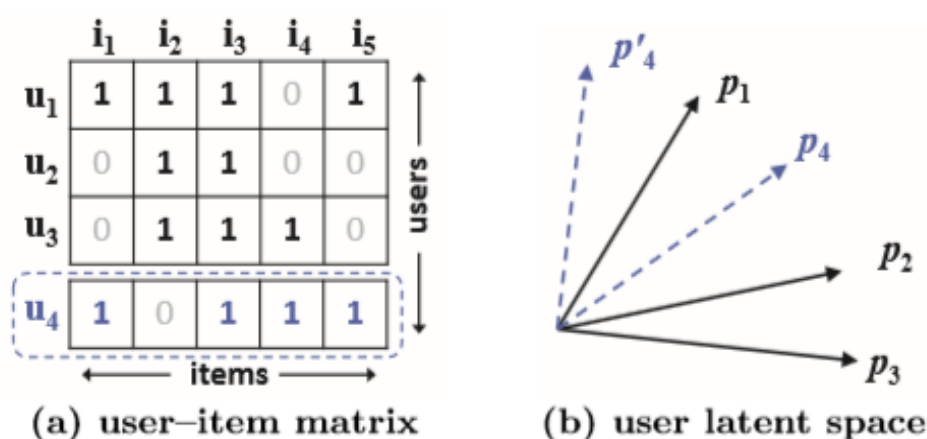


Fig 1: An example illustrates MF's limitation. From data matrix (a), u_4 is most similar to u_1 , followed by u_3 , and lastly u_2 . However in the latent space (b), placing p_4 closest to p_1 makes p_4 closer to p_2 than p_3 , incurring a large ranking loss.

2. Matrix Factorization

MF connects each user and item with a real-valued vector of latent features.

Let p_u and q_i denote the latent vector for user u and item i , separately; MF

appraises a collaboration y_{ui} as the inner product of p_u and q_i :

$$\hat{y}_{ui} = f(u, i | \mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u^T \mathbf{q}_i = \sum_{k=1}^K p_{uk} q_{ik},$$

where K means the dimension of the latent space. As we can see, MF models the two-way cooperation of user and item latent factors, assuming each dimension of the latent space is free of one another of each other and linearly combining them with the identical weight. As such, MF is deemed as a linear model of latent factors.

Neural Collaborative Filtering

This work addresses the previously mentioned research issues by formalizing a neural network demonstrating approach for collaborative filtering. We center

around implicit feedback, which indirectly reflects users' inclination through behaviors like watching videos,

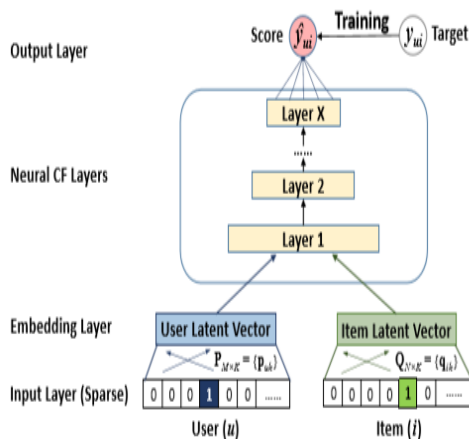
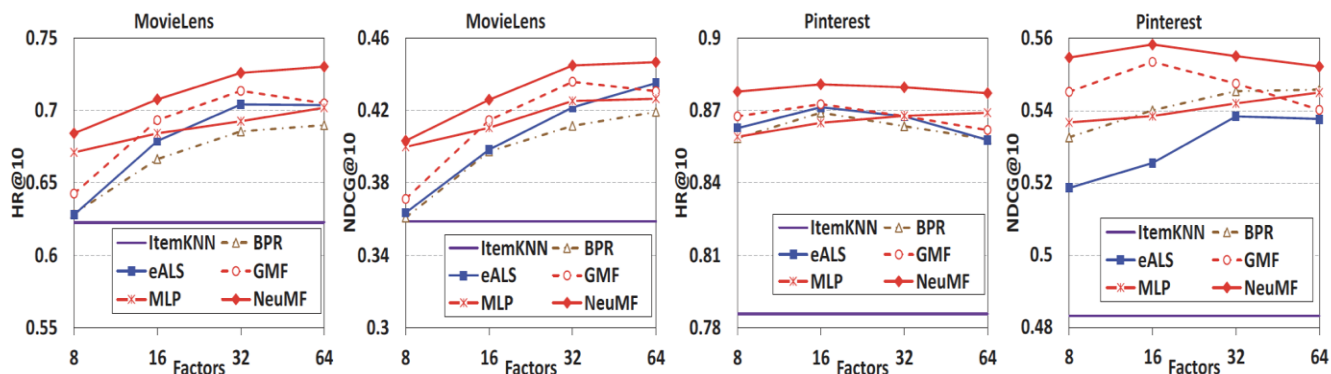


Figure 2: Neural collaborative filtering framework

purchasing items and clicking things. Compared to explicit feedback (i.e., ratings and reviews), implicit feedback can be followed automatically and is thus lot simpler to collect for content suppliers. Nonetheless, it is more challenging to use, since user satisfaction is not observed and there is a natural scarcity of negative feedback. In this paper, we investigate the focal theme of how to use DNNs to demonstrate noisy implicit feedback signals.



(a) MovieLens — HR@10 (b) MovieLens — NDCG@10 (c) Pinterest — HR@10 (d) Pinterest — NDCG@10
Figure 4: Performance of HR@10 and NDCG@10 *w.r.t.* the number of predictive factors on the two datasets.

NeuMF

In order to introduce additional non-linearity, the final model proposed, NeuMF, includes a Multiple-layer Perceptron (MLP) module apart from the Generalized Matrix Factorization (GMF) layer.

GMF that applies the linear kernel to model user-item interaction like vanilla MF

MLP that uses multiple neural layers to layer non linear interactions

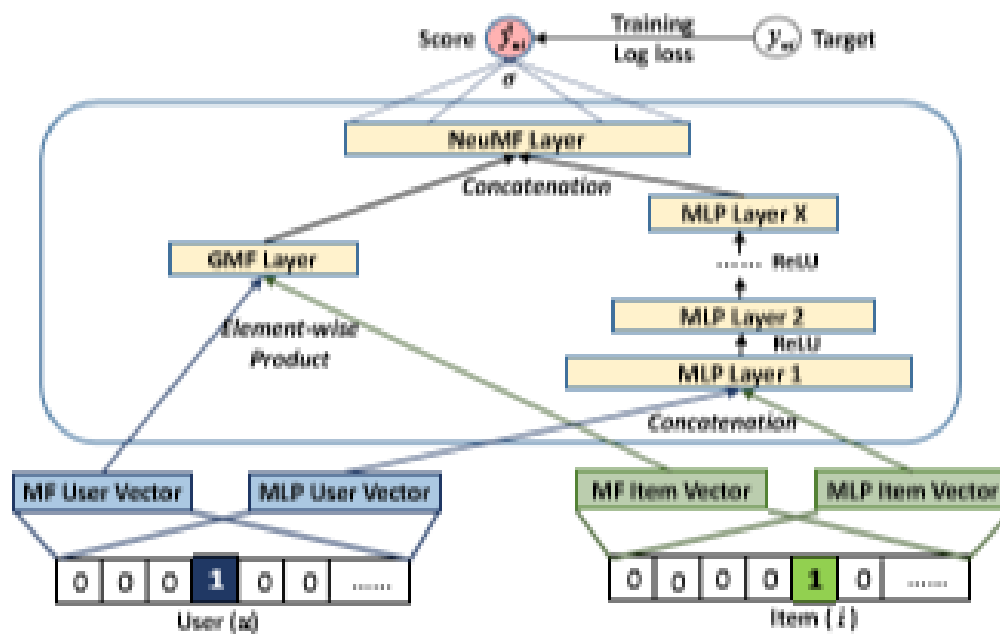


Figure 3: Neural matrix factorization model

The performance comparison is shown in the figure below. In all cases, NeuMF performs better than the other models. In addition, the paper demonstrates the effectiveness of pre-training the individual modules for NeuMF. After training

GMF and MLP separately, they set the weight of the trained GMF and MLP as the initialization of NeuMF.

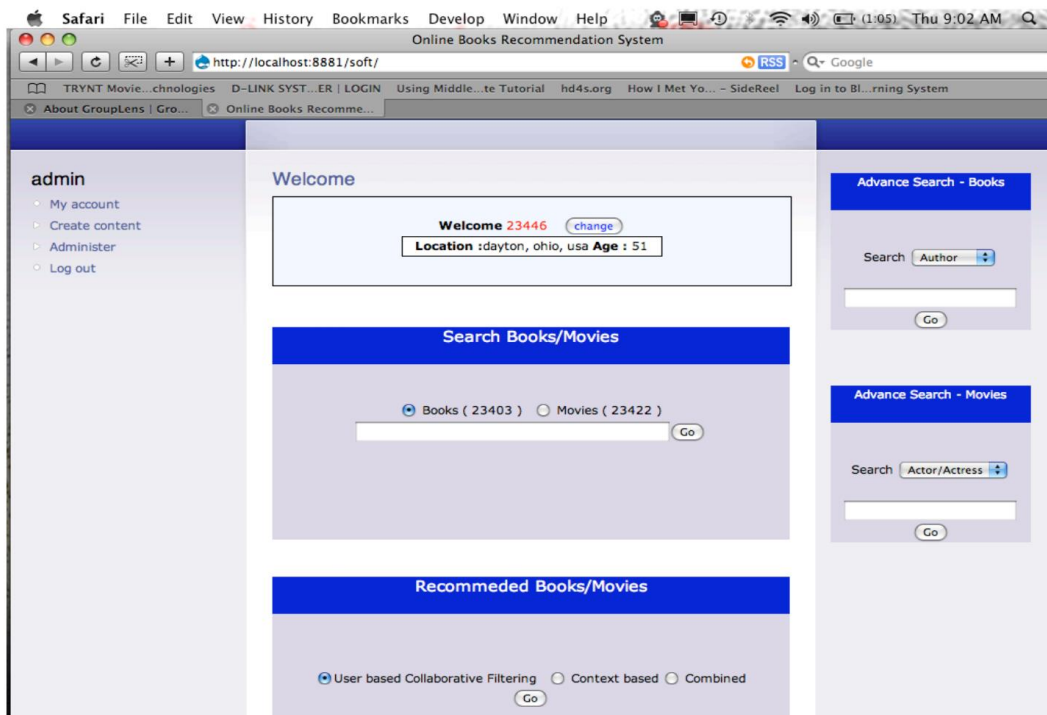
Is Deep Learning Helpful?

(RQ3) As there is little work on learning user–item interaction function with neural networks, it is curious to see whether using a deep network structure is beneficial to the recommendation task. Towards this end, we further investigated MLP with different number of hidden layers. The results are summarized in Table 3 and 4. The MLP-3 indicates the MLP method with three hidden layers (besides the embedding layer), and similar notations for others. As we can see, even for models with the same capability, stacking more layers are beneficial to performance. This result is highly encouraging, indicating the effectiveness of using deep models for collaborative recommendation. We attribute the improvement to the high non-linearities brought by stacking more non-linear layers. To verify this, we further tried stacking linear layers, using an identity function as the activation function. The performance is much worse than using the ReLU unit. For MLP-0 that has no hidden layers (i.e., the embedding layer is directly projected to predictions), the performance is very weak and is not better than the non-personalized ItemPop. This verifies our argument in Section 3.3 that simply concatenating user and item latent vectors is insufficient for modelling their feature interactions, and thus the necessity of transforming it with hidden layers.

IMPLEMENTATION

1. System Screenshots

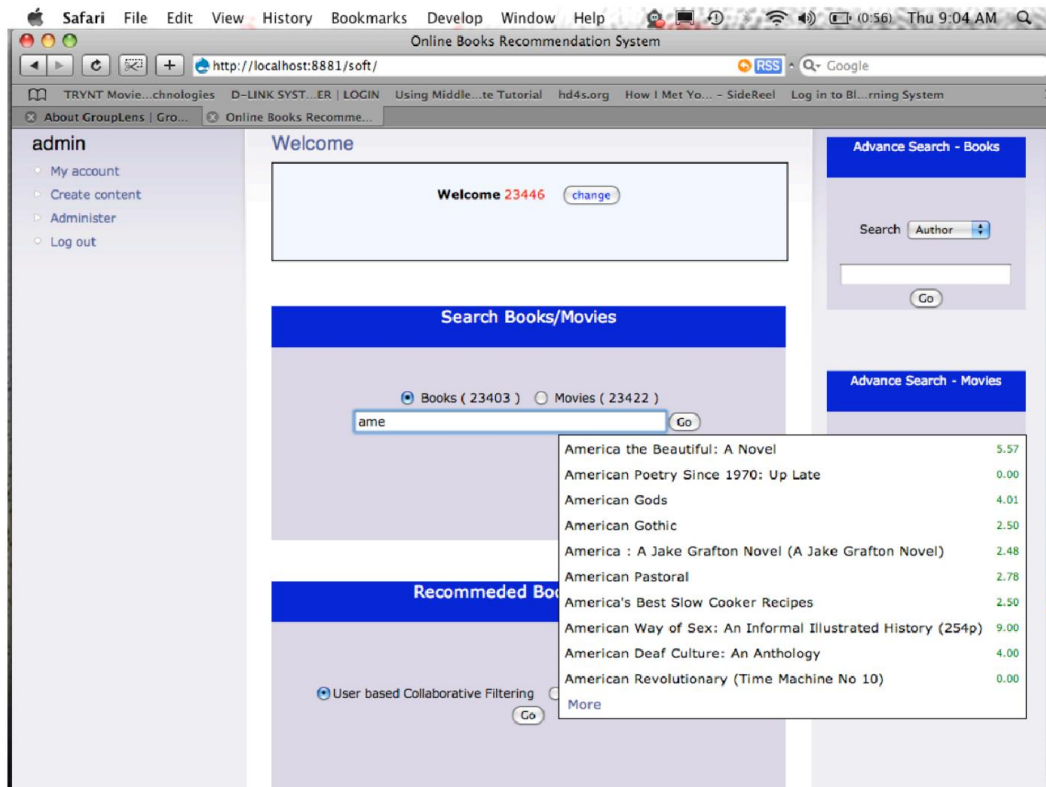
1.1 Home Screen



This is how the home screen for the online recommendation system looks like.

To begin recommendation process the user first has to enter the 'userID'. We can see this in the above figure where User '23446' has just logged. The session for this user has to remain active through out the recommendation process in order for the system to make recommendations.

1.2 Book Search



The above figure shows the implementation of the auto search feature 24 as described above, the figure displays 10 books with their average ratings along side matching the keyword 'ame' entered by the active user. If the match is not seen the more link can be clicked to see other matching results.

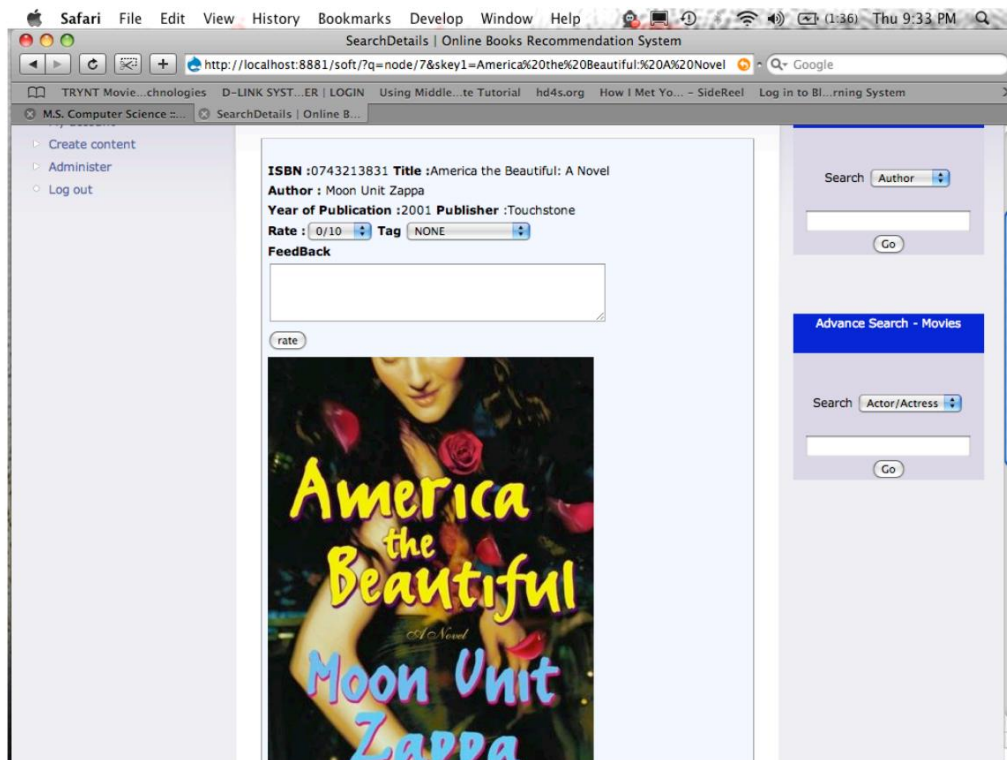
1.3 'More' Keyword

The screenshot shows a web browser window with the following details:

- Browser:** Safari
- Page Title:** DisplayAll | Online Books Recommendation System
- Address Bar:** <http://localhost:8881/soft/?q=node/9&mkey=ame>
- Page Content:**
 - Header:** Online Books Recommendation System
 - Navigation:** Home, DisplayAll (View, Edit)
 - Admin Panel (Left):** admin, My account, Create content, Administer, Log out
 - Search Panels (Right):** Advance Search - Books (Search Author), Advance Search - Movies (Search Actor/Actress)
 - Book Grid:**
 - Book 1:** Title :America the Beautiful: A Novel
 - Book 2:** Title :American Poetry Since 1970: Up Late
 - Book 3:** Title :American Gods
 - Book 4:** Title :American Gothic
 - Book 5:** Title :America : A Jake Grafton Novel (A Jake Grafton Novel)
 - Book 6:** Title :American Pastoral

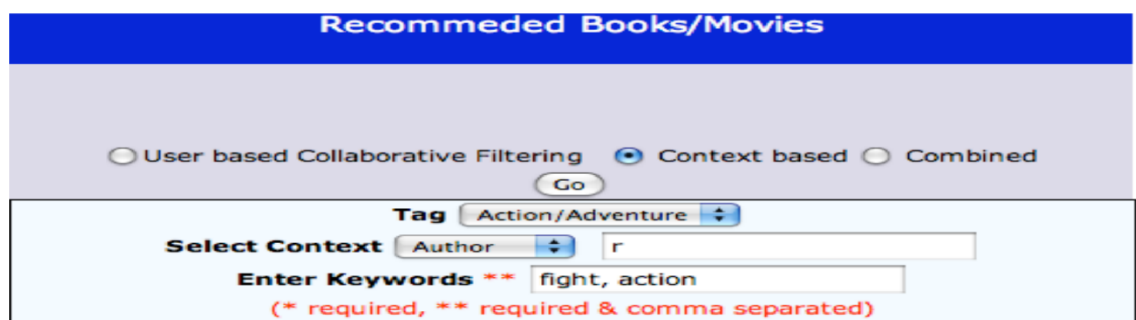
The above figure shows the results of top books matching the keyword 'ame' when the more link is clicked.

1.4 Results Book Search

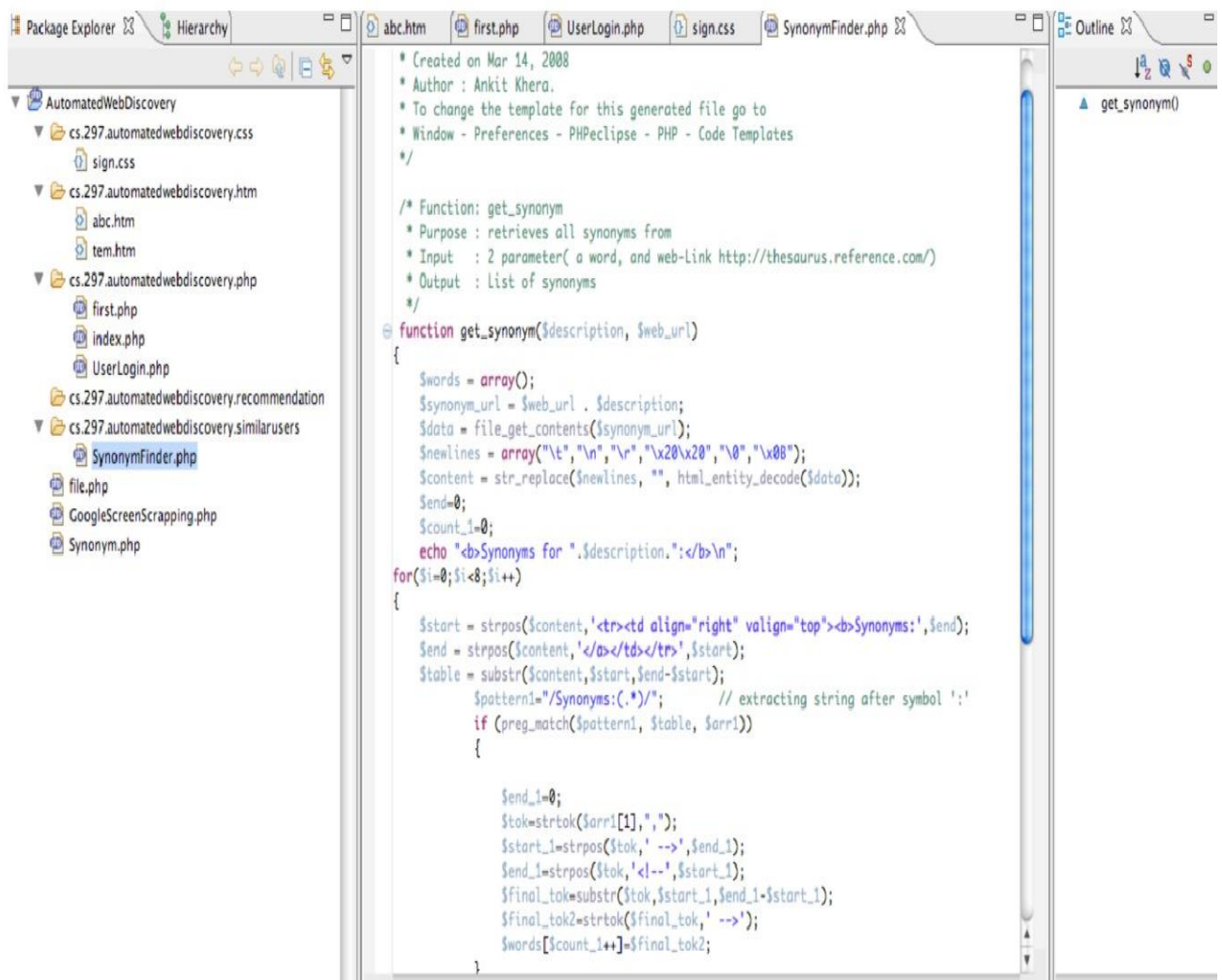


The above figure shows the details of the book like isbn, title, author, year of publication, publisher, rating, tag, feedback, and description [not visible in snapshot due to lack of space] etc. The user can rate the new book or update his current ratings here.

1.5 Recommendation



The above figure shows the initial screen shown to the user where the context information is gathered from the user. The active user chooses the tag, selects the parent context category, enters keyword to be searched under the parent context category and finally enters the free keywords, which he/she might be of interested in.



```

Created on Mar 14, 2008
Author : Ankit Khera.
To change the template for this generated file go to
Window - Preferences - PHPeclipse - PHP - Code Templates
*/

/* Function: get_synonym
 * Purpose : retrieves all synonyms from
 * Input  : 2 parameter( a word, and web-Link http://thesaurus.reference.com/)
 * Output : List of synonyms
 */
function get_synonym($description, $web_url)
{
    $words = array();
    $synonym_url = $web_url . $description;
    $data = file_get_contents($synonym_url);
    $newlines = array("\t", "\n", "\r", "\x20\x20", "\0", "\x00");
    $content = str_replace($newlines, "", html_entity_decode($data));
    $send=0;
    $count_1=0;
    echo "<b>Synonyms for ".$description."</b>\n";
    for($i=0;$i<8;$i++)
    {
        $start = strpos($content, '<tr><td align="right" valign="top"><b>Synonyms: ', $send);
        $end = strpos($content, '</td></tr>', $start);
        $stable = substr($content, $start, $end-$start);
        $pattern1="/Synonyms:(.*)/"; // extracting string after symbol ':'
        if (preg_match($pattern1, $stable, $arr1))
        {
            $send_1=0;
            $tok=strtok($arr1[1], ",");
            $start_1=strpos($tok, "-->", $send_1);
            $end_1=strpos($tok, '<!--', $start_1);
            $final_tok=substr($tok, $start_1, $end_1-$start_1);
            $final_tok2=strtok($final_tok, "-->");
            $words[$count_1++]=$final_tok2;
        }
    }
}

```

The purpose of this php script is to screen scrap synonyms from a website and use it for recommendations. The script captures the first keyword (synonym) in each sense amongst the number of keywords in each sense.

```

import os
import shutil
# import papermill as pm
import pandas as pd
import numpy as np
import tensorflow as tf
from reco_utils.recommender.ncf.ncf_singlenode import NCF
from reco_utils.recommender.ncf.dataset import Dataset as NCFDataset
from reco_utils.dataset import movielens
from reco_utils.dataset.python_splitters import python_chrono_split
from reco_utils.evaluation.python_evaluation import (rmse, mae, rsquared, exp_var, map_at_k, ndcg_at_k, precision_at_k,
                                                    recall_at_k, get_top_k_items)
from reco_utils.common.constants import SEED as DEFAULT_SEED

# top k items to recommend
TOP_K = 10

# Select MovieLens data size: 100k, 1m, 10m, or 20m
MOVIELENS_DATA_SIZE = '100k'

# Model parameters
EPOCHS = 100
BATCH_SIZE = 256

SEED = DEFAULT_SEED # Set None for non-deterministic results

# loading movie leans data
df = movielens.load_pandas_df(size=MOVIELENS_DATA_SIZE, header=["userID", "itemID", "rating", "timestamp"])

```

	userID	itemID	prediction
0	1.0	149.0	0.050952
1	1.0	88.0	0.472271
2	1.0	101.0	0.125187
3	1.0	110.0	0.225378
4	1.0	103.0	0.005792

```

# chronological split on every user
train, test = python_chrono_split(df, 0.75)

# preparing the data
# This function by default takes sampling on unobserved entries to account for negative feedback
# It only takes rows which atleast have user id, item id and rating
# I know we discussed the benefits of using implicit data but NCF works for explicit data too
# This data contains rating and hence is the textbook definition of explicit data
data = NCFDataset(train=train, test=test, seed=SEED)

model = NCF(n_users=data.n_users, n_items=data.n_items, model_type="NeuMF", n_factors=4, layer_sizes=[16, 8, 4], \
            n_epochs=EPOCHS, batch_size=BATCH_SIZE, learning_rate=1e-3, verbose=10, seed=SEED)
# n_factors (int): Dimension of latent space.
# layer_sizes (list): Number of layers for MLP.

# training the model
model.fit(data)

predictions = [[row.userID, row.itemID, model.predict(row.userID, row.itemID)]
               for (_, row) in test.iterrows()]

# saving the predictions in a dataframe
predictions = pd.DataFrame(predictions, columns=['userID', 'itemID', 'prediction'])
predictions.head()

```

	userID	itemID	rating	timestamp
0	196	242	3.0	881250949
1	186	302	3.0	891717742
2	22	377	1.0	878887116
3	244	51	2.0	880606923
4	166	346	1.0	886397596

Advantages of the System

1) The System would benefit those users who have to use search engines to locate relevant content. They have to scroll through pages of results to find relevant content.

2) Rather than searching for quality web pages, the users of this system would be directly taken to quality web pages matching their personal interests and preferences.

3) The system would deliver quality web pages as it is not just dependent on the rating given by other users which could be deceiving at times.

RESULT

To demonstrate the utility of pre-training for NeuMF, we compared the performance of two versions of NeuMF — with and without pre-training. For NeuMF without pretraining, we used the Adam to learn it with random initializations. As shown in Table, the NeuMF with pretraining achieves better performance in most cases; only for MovieLens with a small predictive factors of 8, the pretraining method performs slightly worse. The relative improvements of the NeuMF with pre-training are 2.2% and 1.1% for MovieLens and Pinterest, respectively. This result justifies the usefulness of our pre-training method for initializing NeuMF.

Table 2: Performance of NeuMF with and without pre-training.

Factors	With Pre-training		Without Pre-training	
	HR@10	NDCG@10	HR@10	NDCG@10
MovieLens				
8	0.684	0.403	0.688	0.410
16	0.707	0.426	0.696	0.420
32	0.726	0.445	0.701	0.425
64	0.730	0.447	0.705	0.426
Pinterest				
8	0.878	0.555	0.869	0.546
16	0.880	0.558	0.871	0.547
32	0.879	0.555	0.870	0.549
64	0.877	0.552	0.872	0.551

CONCLUSION AND FUTURE WORK

In this work, we explored neural network architectures for collaborative filtering. We devised a general framework NCF and proposed three instantiations — GMF, MLP and NeuMF — that model user–item interactions in different ways. Our framework is simple and generic; it is not limited to the models presented in this paper, but is designed to serve as a guideline for developing deep learning methods for recommendation. This work complements the mainstream shallow models for collaborative filtering, opening up a new avenue of research possibilities for recommendation based on deep learning. In future, we will study pairwise learners for NCF models and extend NCF to model auxiliary information, such as user reviews, knowledge bases, and temporal signals. While existing personalization models have primarily focused on individuals, it is interesting to develop models for groups of users, which help the decision-making for social groups. Moreover, we are particularly interested in building recommender systems for multi-media items, an interesting task but has received relatively less scrutiny in the recommendation community. Multi-media items, such as images and videos, contain much richer visual semantics that can reflect users’ interest. To build a multi-media recommender system, we need to develop effective methods to learn from multi-view and multi-modal data.

REFERENCES

1. I. Bayer, X. He, B. Kanagal, and S. Rendle. A generic coordinate descent framework for learning from implicit feedback. In WWW, 2017.
2. A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In NIPS, pages 2787–2795, 2013.
3. T. Chen, X. He, and M.-Y. Kan. Context-aware image tweet modelling and recommendation. In MM, pages 1018–1027, 2016.
4. S. Sedhain, A. K. Menon, S. Sanner, and L. Xie. Autorec: Autoencoders meet collaborative filtering. In WWW, pages 111–112, 2015.
5. R. Socher, D. Chen, C. D. Manning, and A. Ng. Reasoning with neural tensor networks for knowledge base completion. In NIPS, pages 926–934, 2013.