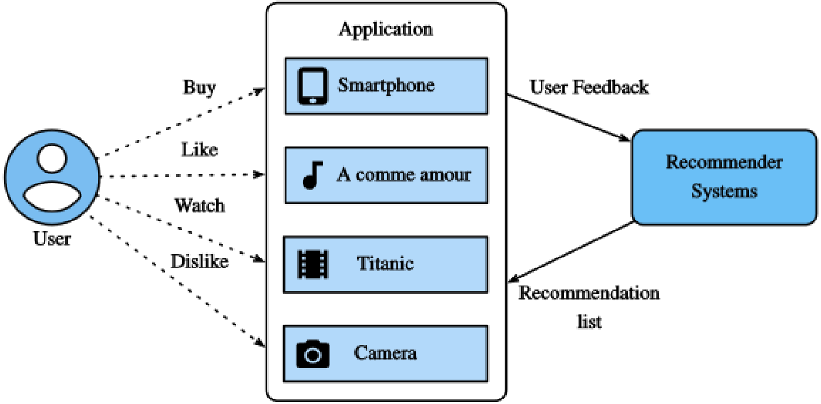
SOFTWARE ENGINEERING AND MACHINE LEARNING

Gerasimos
gerasimos@pitt.edu

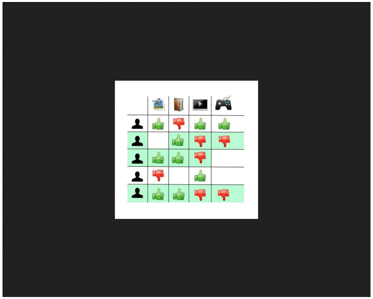
University of Pittsburgh
Instructor Dr. S. K. Chang

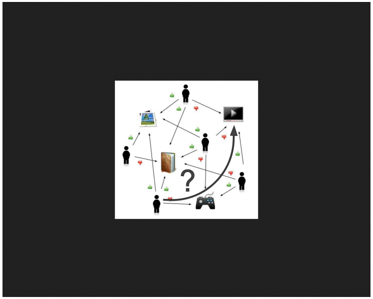
October 27, 2020

RECOMMENDATION



FROM RANKING TO DL





SIMPLE RECOMMENDER: SORT THEM FIRST

Simple recommenders are basic systems that recommend the top items based on a certain metric or score. We will build a simplified clone of IMDB Top 250 Movies using metadata collected from IMDB.
The following are the steps involved:

– Decide on the metric or score to rate movies on.
– Calculate the score for every movie.
– Sort the movies based on the score and output the top results.

DATASET: FIRST REVIEW

The dataset file

- contains metadata for all 45,000 movies listed in the Full MovieLens Dataset.
- consists of movies released on or before July 2017.
- captures feature points like cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts, and vote averages.

RECOMMENDER 1: SIMPLE STATISTICAL METRIC

These feature points could be potentially used to train your machine learning models for content and collaborative filtering.

USING RATING AS A METRIC. Why should we or shouldn't we?

NAIVE OUTPUT

```
                          title                        vote_average
21642   Ice Age Columbus: Who                                  10.0
        Were the First Americans?

15710   If God Is Willing and                                  10.0
          da Creek Don't Rise

22396   Meat the Truth                                         10.0

22395   Marvin Hamlisch:                                       10.0
        What He Did For Love
```

RECOMMENDER 1: SIMPLE STATISTICAL METRIC

The metric used is

$$\mathit{WeightedRating}(\mathrm{WR}) = \left(\frac{v}{v+m} \cdot R\right) + \left(\frac{m}{v+m} \cdot C\right) \quad (1)$$

– v is the number of votes for the movie (ALREADY HAVE IT);
– m: minimum votes required to be listed (ALREADY HAVE IT);
– R is the average rating of the movie;
– C is the mean vote across the whole report.

Determining an appropriate value for m is a hyperparameter that you can choose accordingly since there is no right value for m. You can consider it as a preliminary negative filter.
The selectivity of your filter is up to your discretion.

DATASET: MORE ON THIS

The dataset columns look like this:

TITLES: adult belongs_to_collection budget genres homepage id imdb_id original_language original_title overview ... release_date revenue runtime spoken_languages status tagline title video vote_average vote_count

VALUES: 0 False {'id': 10194, 'name': 'Toy Story Collection', ... 30000000 [{'id': 16, 'name': 'Animation'}, {'id': 35, '... http://toystory.disney.com/toy-story 862 tt0114709 en Toy Story Led by Woody, Andy's toys live happily in his ... ... 1995-10-30 373554033.0 81.0 [{'iso_639_1': 'en', 'name': 'English'}] Released NaN Toy Story False 7.7 5415.0

CODE I

```
def weighted_rating(x, m=m, C=C):
    v = x['vote_count']
    R = x['vote_average']
    return (v/(v+m) * R) + (m/(m+v) * C)

import pandas as pd
metadata = pd.read_csv('movies_metadata.csv',
                               low_memory=False)
metadata.head(3)
C = metadata['vote_average'].mean()
m = metadata['vote_count'].quantile(0.90)
q_movies = metadata.copy().loc
                  [metadata['vote_count'] >= m]
q_movies.shape
print(q_movies.shape)
metadata.shape
```

CODE II

```
q_movies['score'] = q_movies.apply
                     (weighted_rating, axis=1)

q_movies = q_movies.sort_values('score',
                                ascending=False)

q_movies[['title', 'vote_count', 'vote_average',
                                'score']].head(20)

metadata['overview'].head()
```

OUTPUT: RECOMMENDATIONS ACCORDING SCORE

```
              title           vote      vote          score
                              count     average
314  The  Shawshank           8358.0     8.5          8.445869
        Redemption
834  The  Godfather           6024.0     8.5          8.425439

10309  Dilwale                 661.0     9.1          8.421453
       Dulhania
       Le  Jayenge
12481  The  Dark             12269.0     8.3          8.265477
       Knight
2843   Fight  Club            9678.0     8.3          8.256385

292  Pulp  Fiction            8670.0     8.3          8.251406

522  Schindler ' s            4436.0     8.3          8.206639
     List
```

RECOMMENDER 2: NLP AS THE MACHINE LEARNING METHOD

Aside from voting, which other type of recommendation is there? Learn how to build a system that **recommends movies that are SIMILAR to** a particular movie. How to achieve this?

– The plot description is available to you as the overview feature in your metadata dataset.

– compute pairwise cosine similarity scores for all movies based on their plot descriptions

– recommend movies based on that similarity score threshold.

RECOMMENDER 2: INPUT TO NLP SYSTEM

```
0 Led by Woody, Andy's toys live happily in his ...
1 When siblings Judy and Peter discover an encha...
2 A family wedding reignites the ancient feud be...
3 Cheated on, mistreated and stepped on, the wom...
4 Just when George Banks has recovered from his ...
Name: overview, dtype: object
```

RECOMMENDER 2: PRE PROCESSING FOR NLP ALGORITHM

Which score to use?

- Import the Tfidf module using scikit-learn (scikit-learn gives you a built-in TfIdfVectorizer class that produces the TF-IDF matrix);

- Remove stop words like 'the', 'an', etc. since they do not give any useful information about the topic;

- Replace not-a-number values with a blank string;

- Finally, construct the TF-IDF matrix on the data.

CODE I

```
#Import TfIdfVectorizer from scikit-learn
from sklearn.feature_extraction.text
                                        import TfidfVectorizer

#Define a TF-IDF Vectorizer Object. Remove all
                   English stop words such as 'the', 'a'
tfidf = TfidfVectorizer(stop_words='english')

#Replace NaN with an empty string
metadata['overview'] = metadata['overview'].
                                                       fillna('')

#Construct the required TF-IDF matrix by fitting
                                             and transforming the data

tfidf_matrix = tfidf.fit_transform(
                                                      metadata['overview'])
```

CODE II

```
#Output the shape of tfidf_matrix
tfidf_matrix.shape

#Array mapping from feature integer indices
                                    to feature name.
tfidf.get_feature_names()[5000:5010]

# Import linear_kernel
from sklearn.metrics.pairwise import
                                    linear_kernel

# Compute the cosine similarity matrix
cosine_sim = linear_kernel(tfidf_matrix,
                                    tfidf_matrix)
```

NAIVE OUTPUT

```
get_recommendations('The Dark Knight Rises')

12481 The Dark Knight
150   Batman Forever
1328  Batman Returns
15511 Batman: Under the Red Hood
585   Batman
21194 Batman Unmasked: The Psychology of the
                                      Dark Kn...
9230  Batman Beyond: Return of the Joker
18035 Batman: Year One
19792 Batman: The Dark Knight Returns, Part 1
3095  Batman: Mask of the Phantasm
```

– **GOAL ACHIEVED** A decent job was done of finding movies with **SIMILAR plot descriptions**

NAIVE OUTPUT

```
get_recommendations('The Dark Knight Rises')
12481 The Dark Knight
150   Batman Forever
1328  Batman Returns
15511 Batman: Under the Red Hood
585   Batman
21194 Batman Unmasked: The Psychology of the
                                   Dark Kn...
9230  Batman Beyond: Return of the Joker
18035 Batman: Year One
19792 Batman: The Dark Knight Returns, Part 1
3095  Batman: Mask of the Phantasm
```

– However the quality of recommend. is not great: "The Dark Knight Rises" returns all Batman movies while it is more likely that the people who liked that movie are more inclined to enjoy other Christopher Nolan movies.

RECOMMENDER 2: NLP

So what did we just use?
– As the name suggests, word vectors are vectorized representation of words in a document. The vectors carry a semantic meaning with it. For example, man & king will have vector representations close to each other while man & woman would have representation far from each other.
– Compute Term Frequency-Inverse Document Frequency (TF-IDF) vectors for each document. This will give you a matrix where each row represents a word in the overview vocabulary (all the words that appear in at least one document), and each column represents a movie, as before.
– The TF-IDF score: is the frequency of a word occurring in a document, down-weighted by the number of documents in which it occurs.
– Reduces the importance of words that frequently occur in plot overviews and, therefore, their significance in computing the final similarity score.

RECOMMENDER 2: NLP ALGORITHM COMPLETED

– Get the index of the movie given its title.

– Get the list of cosine similarity scores for that particular movie with all movies. Convert it into a list of tuples where the first element is its position, and the second is the similarity score.

– Sort the aforementioned list of tuples based on the similarity scores; that is, the second element.

– Get the top 10 elements of this list. Ignore the first element as it refers to self (the movie most similar to a particular movie is the movie itself).

– Return the titles corresponding to the indices of the top elements.

RECOMMENDER 2: NLP ALGORITHM IMPROVED

```
credits = pd.read_csv('credits.csv')
keywords = pd.read_csv('keywords.csv')

# Remove rows with bad IDs.
metadata = metadata.drop([19730, 29503, 35587])

# Convert IDs to int. Required for merging
keywords['id'] = keywords['id'].astype('int')
credits['id'] = credits['id'].astype('int')
metadata['id'] = metadata['id'].astype('int')
```

RECOMMENDER 2: NLP ALGORITHM OUTPUT

```
get_recommendations('The Dark Knight Rises',
                                        cosine_sim2)
```

```
12589          The Dark Knight
10210            Batman Begins
9311                    Shiner
9874           Amongst Friends
7772                  Mitchell
516         Romeo Is Bleeding
11463             The Prestige
24090                Quicksand
25038                 Deadfall
41063                     Sara
Name: title, dtype: object
```

RECOMMENDER 3: FM AS THE DEEP LEARNING METHOD

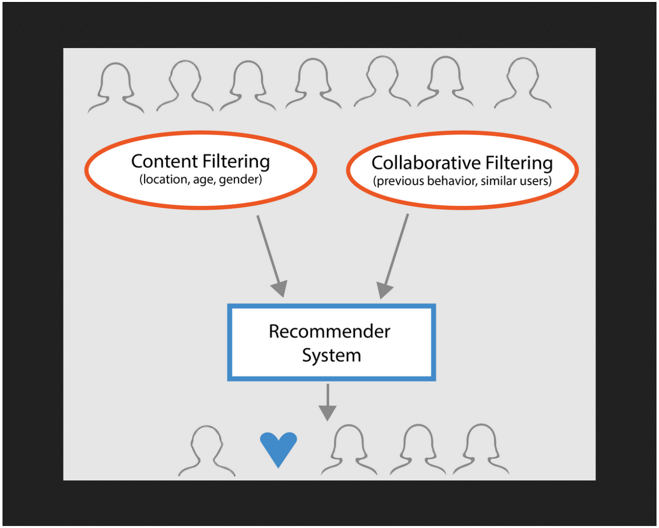


Figure: FM

RECOMMENDER 3: FM AS THE DEEP LEARNING METHOD

– Item-based (Filtering): similar to the content recommendation engine that we built. These systems identify similar items based on how people have rated it in the past. E.g., if Alice, Bob, & Eve have given 5 stars to The Lord of the Rings & The Hobbit, the system identifies the items as similar. Therefore, if someone buys The Lord of the Rings, the system also recommends The Hobbit to him or her.

Which other type of recommendation is there?
– User-based (Filtering): recommends products that similar users have liked.

– E.g. Alice & Bob have a similar interest in books (i.e. they largely like & dislike the same books). Now, a new book has been launched into the market & Alice has read & loved it. It is, thus, likely that Bob will like it too. So the system recommends it.

FM

The architecture of DeepFM is illustrated below.

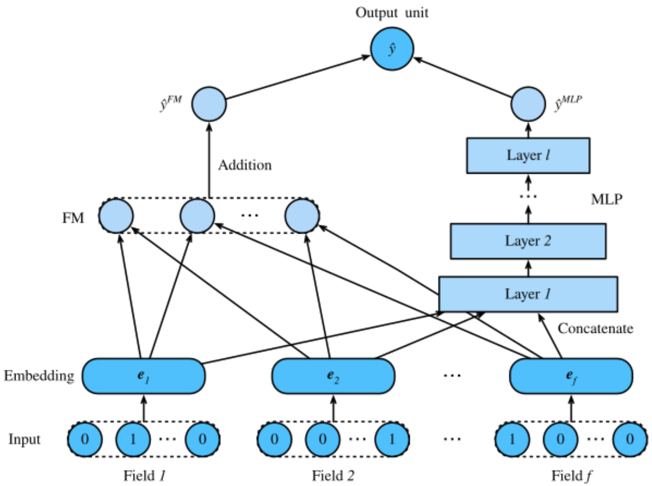


Figure: FM

DEEP LEARNING RECOMMENDER 3: FM AS A ML METHOD

– Learning effective feature combinations is critical to the success of click-through rate prediction task. Factorization machines model feature interactions in a linear paradigm (e.g., bilinear interactions).

– This is often insufficient for real-world data where inherent feature crossing structures are usually very complex and nonlinear.

– What's worse, second-order feature interactions are generally used in factorization machines in practice.

– Modeling higher degrees of feature combinations with factorization machines is possible theoretically but it is usually not adopted due to numerical instability and high computational complexity. One effective solution is using deep neural networks.

ANALYSIS OF FM AS DL SYSTEM

– Deep neural networks are powerful in feature representation learning and have the potential to learn sophisticated feature interactions.

– As such, it is natural to integrate deep neural networks to factorization machines. Adding nonlinear transformation layers to factorization machines gives it the capability to model both low-order feature combinations and high-order feature combinations.

– Moreover, non-linear inherent structures from inputs can also be captured with deep neural networks. In this section, we will introduce a representative model named deep factorization machines (DeepFM) [Guo et al., 2017] which combine FM and deep neural networks.

CODE I

```
from d2l import mxnet as d2l
from mxnet import init, gluon, np, npx
from mxnet.gluon import nn
import os
import sys
npx.set_np()


class DeepFM(nn.Block):
    def __init__(self, field_dims, num_factors, mlp
        super(DeepFM, self).__init__()
        num_inputs = int(sum(field_dims))
        self.embedding = nn.Embedding(num_inputs, n
        self.fc = nn.Embedding(num_inputs, 1)
        self.linear_layer = nn.Dense(1, use_bias=Tr
        input_dim = self.embed_output_dim = len(fiel
        self.mlp = nn.Sequential()
```

CODE II

```
batch_size = 2048
data_dir = d2l.download_extract('ctr')
train_data = d2l.CTRDataset(os.path.join(data_dir,
test_data = d2l.CTRDataset(os.path.join(data_dir, '
                           feat_mapper=train_data.fe
                           defaults=train_data.defa
field_dims = train_data.field_dims
train_iter = gluon.data.DataLoader(
    train_data, shuffle=True, last_batch='rollover'
    num_workers=d2l.get_dataloader_workers())
test_iter = gluon.data.DataLoader(
    test_data, shuffle=False, last_batch='rollover'
    num_workers=d2l.get_dataloader_workers())
devices = d2l.try_all_gpus()
net = DeepFM(field_dims, num_factors=10, mlp_dims=[
net.initialize(init.Xavier(), ctx=devices)
lr, num_epochs, optimizer = 0.01, 30, 'adam'
```

OUTPUT

Try to implement it in your own dataset and compare to previous systems.

DEEP LEARNING RECOMMENDER 3: FM

The implementation of DeepFM is similar to that of FM. We keep the FM part unchanged and use an MLP block with relu as the activation function. Dropout is also used to regularize the model. The number of neurons of the MLP can be adjusted with the mlp_dims hyperparameter.
The data loading process is the same as that of FM. We set the MLP component of DeepFM to a three-layered dense network with the a pyramid structure (30-20-10). All other hyperparameters remain the same as FM.

DEEP LEARNING RECOMMENDER 4: CF

IT IS FUN AND REQUIRES CODING. TRY IT AT HOME!

WHY RECOMMENDER SYSTEMS?

Are among the most popular applications of data science today.

Are used to predict the "rating" or "preference" that a user would give to an item.

Almost every major tech company has applied them: Amazon uses it to suggest products to customers and Facebook uses it to recommend pages to like and people to follow.

For some companies like Netflix, Amazon Prime, Hulu, and Hotstar, the business model and its success revolves around the potency of their recommendations.

Popular systems for domains like restaurants, movies, and online dating.

Have also been developed to explore research articles & experts, collaborators & financial services.

YouTube: uses it at a large scale to suggest you videos based on your history. E.g. if you watch a lot of anime videos, it would suggest similar ones.

AWARDS

Netflix even offered a million dollars in 2009 to anyone who could improve its system by 10%.