Movie Recommender

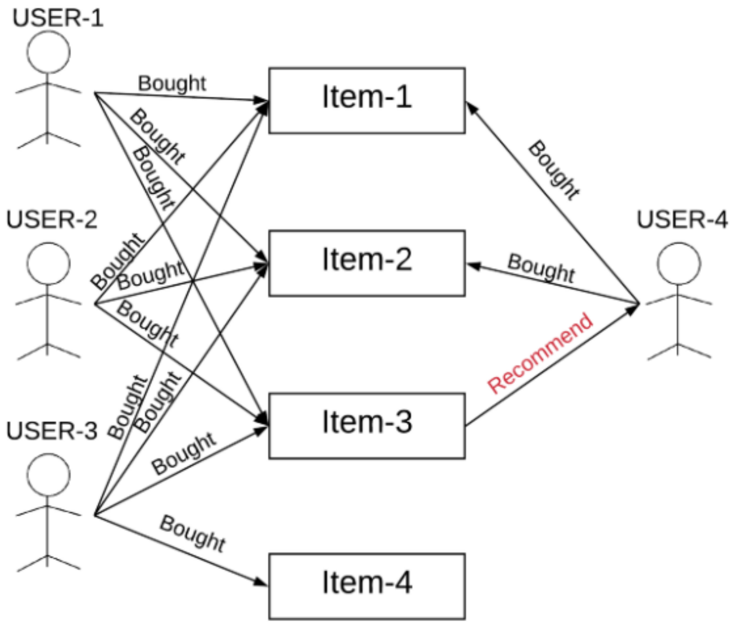
Group member： Zhengqi Dong, Yuntian He

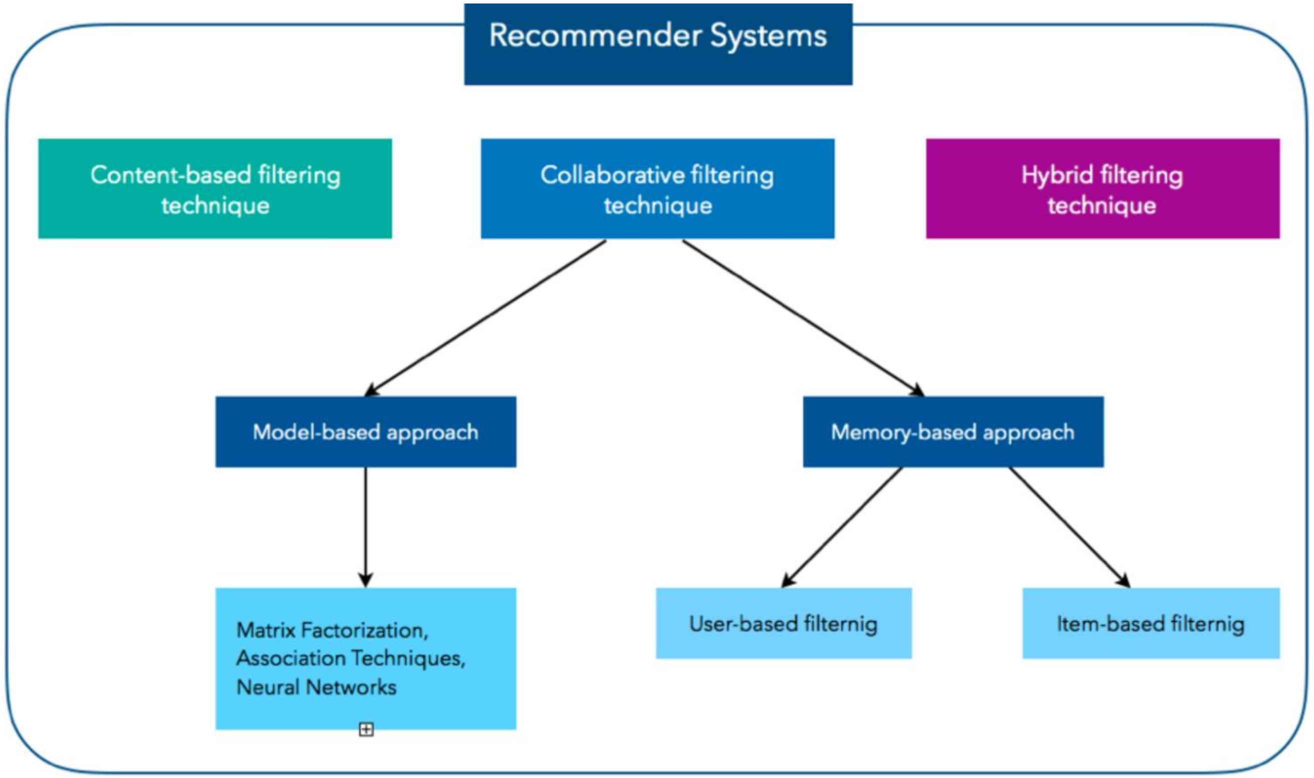
Email: {dong.760, he.1773}@osu.edu

Background

Type of Recommender Systems

- Recommender System:
  - "A recommender system is an information filtering system that seeks to predicts the "rating" or "preference" a user would give to an item."
- Type of Recommender Systems:
  - Content-Based Filtering
  - Collaborative Filtering(CF)
    - Memory-Based Collaborative Filtering, e.g., User-based CF, Item-based CF
    - Model-Based Collaborative Filtering, e.g., Matric factorization, Neural Network
  - Hybrid Filtering





Courtesy: 1) https://d2l.ai/chapter_recommender-systems/recsys-intro.html 2) https://dl.acm.org/doi/pdf/10.1155/2009/421425 3) https://arxiv.org/pdf/1707.07435.pdf, 4) https://humboldt-wi.github.io/blog/research/applied_predictive_modeling_19/causalrecommendersystem/

Dataset description

- **The Movies Dataset** from Kaggle
    - **26M** ratings from **270K** users on **45K** movies
- Content
    - **Text**: Each movie has an overview (a paragraph)
    - **Rating**: A tuple (UserID, MovieID, Rating, Timestamp)
    - **Other Attributes**:
        - Genre: e.g. Action, Animation, Romance, …
        - Credits: (cast, crew)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45466 entries, 0 to 45465
Data columns (total 24 columns):
adult                    45466 non-null object
belongs_to_collection    4494 non-null object
budget                   45466 non-null object
genres                   45466 non-null object
homepage                 7782 non-null object
id                       45466 non-null object
imdb_id                  45449 non-null object
original_language        45455 non-null object
original_title           45466 non-null object
overview                 44512 non-null object
popularity               45461 non-null object
poster_path              45080 non-null object
production_companies     45463 non-null object
production_countries     45463 non-null object
release_date             45379 non-null object
revenue                  45460 non-null float64
runtime                  45203 non-null float64
spoken_languages         45460 non-null object
status                   45379 non-null object
tagline                  20412 non-null object
title                    45460 non-null object
video                    45460 non-null object
vote_average             45460 non-null float64
vote_count               45460 non-null float64
dtypes: float64(4), object(20)
memory usage: 8.3+ MB
```

* https://www.kaggle.com/rounakbanik/the-movies-dataset

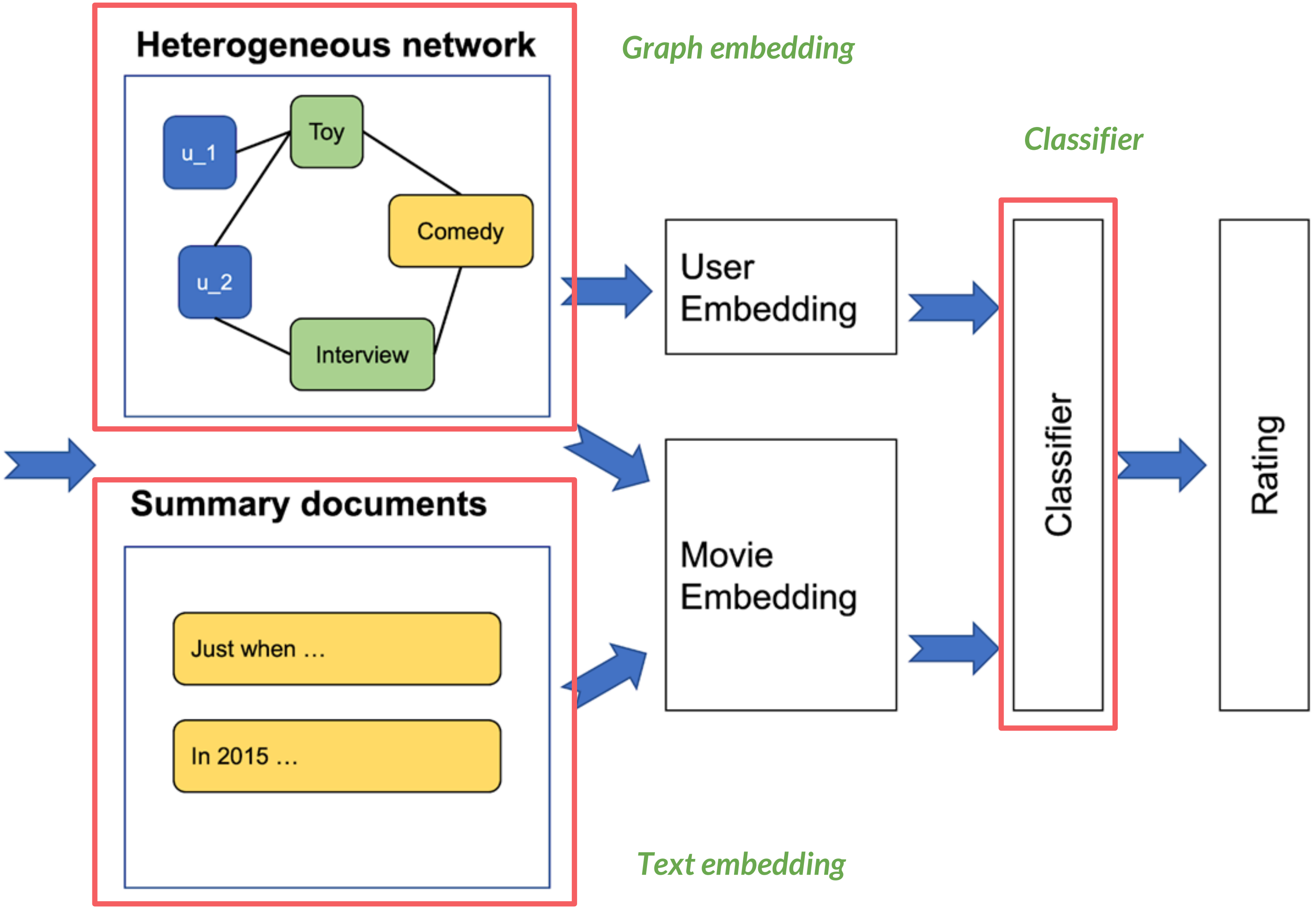
System framework

Ratings

User	Movie	Rating
u_1	Toy	4.5
u_2	Toy	3.0
u_2	Interview	5.0
…	…	…

Movies' metadata

Movie	Genre	Summary
Toy	Comedy	Just when…
Interview	Comedy	In 2015, …
…	…	…



Preprocessing

- Removing data in incorrect format
  - 3 of 45K movies are deleted
- Index adjustment
  - Consecutive IDs for convenience
- Attribute selection
  - **Cast**: Only top 8 casts (cast order included in the raw data)
  - **Crew**: Only use ‘director’

Text Embedding: Doc2vec

- **Goal:** Use Doc2Vec to learning the main content of movies' metadata, and represent it as an $e_{mt,}$ 128 dimensional vector for each movie. (subscript mt is for movie text embedding, and mg is for movie graph embedding)
- **Reason why we used Doc2Vec**
  - It can learn vector representation from unlabeled data and generalized well on the data that do not have enough labels.
  - Dov2vec takes word orders into consideration while learning the semantic meaning of documents.
- **Implementation:**
  - Use gensim package to
  - movie's overview ➔ embedding vectors

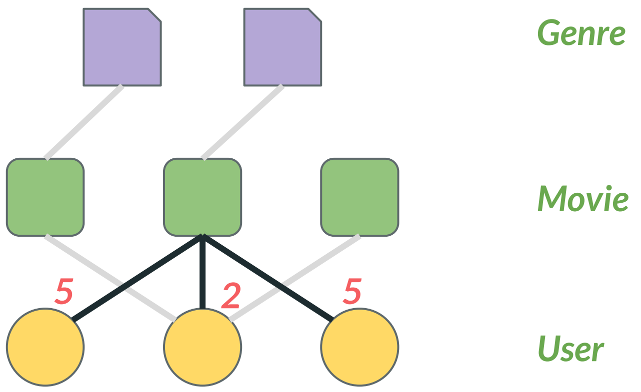
```
print(data["overview"][0])

Led by Woody, Andy's toys live happily in his room until Andy's birthday brings Buzz
Lightyear onto the scene. Afraid of losing his place in Andy's heart, Woody plots aga
inst Buzz. But when circumstances separate Buzz and Woody from their owner, the duo e
ventually learns to put aside their differences.
```

```
Doc2vec_embedding_matrix.txt
0 -0.0260 -0.0769 -0.2531 0.2378 -0.0977 0.1073 0.0766 -0.0402 -0.0968 -0.0743 0.
0076 0.3043 -0.0377 -0.1901 -0.0563 0.0135 -0.2543 0.1872 0.0707 0.0082 -0.0239 0.
1547 -0.1330 -0.0969 0.0660 -0.0376 -0.1378 0.0054 -0.0430 -0.0382 -0.1439 0.0795
-0.0056 0.1860 -0.2107 0.0491 -0.0633 0.0485 0.1126 0.2309 0.1633 -0.0138 -0.3363 0.
0048 0.1191 0.1839 -0.0509 0.0909 -0.0877 0.0866 0.0135 0.1298 0.2585 0.0618 0.0731
0.0932 -0.0329 0.0865 -0.0955 0.2919 -0.2163 0.1959 0.0218 0.1546 0.1474 -0.1151 -0.
2635 -0.0995 0.0317 -0.0847 0.0485 -0.1609 -0.2231 0.1598 -0.2788 0.0882 0.0860 -0.
0617 -0.0340 -0.0745 -0.1872 -0.1337 0.0653 -0.1567 0.0113 0.0696 -0.1352 0.0969 -0.
2050 0.0239 0.0465 -0.0564 0.1535 0.3068 -0.0988 -0.0828 0.0740 0.1495 0.0356 0.
0152 0.1359 -0.2032 -0.0177 0.0057 -0.1460 0.3397 -0.2135 0.0720 -0.0155 -0.1562 -0.
2065 0.1816 0.1081 -0.1329 0.2477 -0.0457 0.1134 -0.1095 -0.1067 -0.0606 0.1615 -0.
0252 0.0567 -0.1241 0.1605 -0.1169 0.1958 0.0713
1 0.0263 0.3395 0.0522 -0.3038 -0.5287 0.0148 -0.0076 -0.1212 0.0351 -0.0467 -0.
0226 0.2438 0.0446 -0.1897 -0.2528 0.0826 -0.4141 0.2273 0.1703 0.1529 -0.0072 0.
0414 0.0070 0.0882 0.3636 -0.1131 -0.0632 0.2656 -0.2260 -0.1883 -0.0675 0.3536 0.
2135 0.0263 -0.0633 -0.0676 -0.0075 0.3716 -0.1841 0.3402 0.1353 -0.3193 -0.1247 -0.
3526 0.0307 -0.0486 -0.0869 0.0427 0.1255 0.1750 -0.2116 0.0084 0.2995 0.0068 -0.
0289 0.2223 0.4006 0.0186 0.1798 -0.0643 -0.1386 0.2952 -0.0762 -0.0061 0.0208 0.
3525 0.0256 0.0085 -0.2165 -0.0769 0.2975 -0.1760 -0.1034 0.3836 -0.0486 0.2048 -0.
1554 0.3606 0.0044 -0.1758 -0.4472 -0.0106 0.0636 -0.0042 -0.0700 0.0142 -0.4559 -0.
0449 -0.1318 0.1850 0.1216 -0.0464 -0.0387 0.1168 0.0426 0.3842 0.1222 -0.1123 0.
1257 -0.2279 -0.0511 -0.3579 0.1174 -0.3187 0.0031 0.0963 0.1261 0.3283 0.0168 0.
1616 -0.0902 -0.1821 0.3913 -0.0432 -0.1317 0.0310 0.1259 0.2729 -0.0415 -0.2106 -0.
1944 -0.0375 0.3309 -0.1496 -0.2304 -0.3824 -0.2119 0.4304
```

Graph embedding: Metapath2vec

- Heterogeneous information network
  - User (U), Movie (M), Genre (G), Cast/crew (C)
- Metapath2vec-based sampling
  - Preserve semantic relationships between nodes
  - U-M-U, U-M-G-M-U, U-M-C-M-U



- **Rating-aware** sampling policy

$$P(s_{t+1} = m | s_t = u) = \begin{cases} 1/|N_M(u)| & , \quad t = 0 \\ \text{softmax}(-|R(u,m) - R(u',m')|) & , \quad \text{else} \end{cases}$$

Similarly sample for P(m->u).

* "metapath2vec: Scalable representation learning for heterogeneous networks." Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining. 2017.

Classification

- Goal:
  - A 3-layey MLP is used as a classifier for our last module to take the text embedding vector and graph embedding to predict the rating score.
- Layer size:
  - 128, 32, 10.
  - Note: input layer takes 128-dimensional vector, and 32 neurons for hidden layer, and 10 neurons for output layer.
- Epochs: 5
- Learning rate: 1*10^-3
- Computing Environment: OSC Owens cluster with single GPU node

Conclusion

- **Methods**
  - Use text embedding only (TEXT)
  - Use graph embedding only (GRAPH)
  - Use both text and graph embedding (BOTH)
- **Key Takeaway**
  - BOTH methods outperforms other two in all metrics
  - BOTH method takes 5.61% longer to train than the other twos on average.
  - ACC (or MSE/MAE) decrease as batch size increases (or increase)
  - Training faster as batch size increases

Table 1: Overall Performance

Name	MAE	MSE	ACC	Time (sec)
TEXT	0.738	1.092	32.014	735.380
GRAPH	0.719	1.061	33.043	740.422
BOTH	**0.713**	**1.034**	**33.052**	781.779

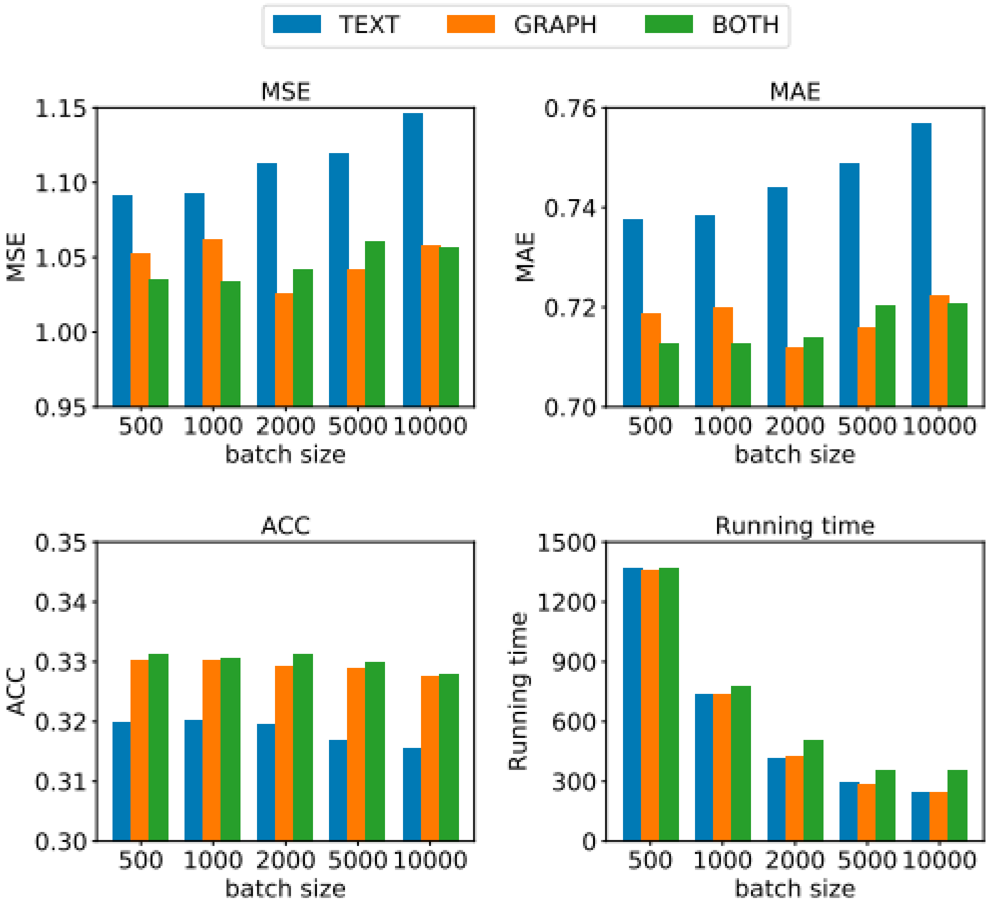


Figure 2: Performance with varying batch sizes

Thanks! Any Question?

Reference

[1]Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deepbidirectional transformers for language understanding.arXiv preprint arXiv:1810.04805, 2018.

[2]Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec: Scalable representationlearning for heterogeneous networks. InProceedings of the 23rd ACM SIGKDD internationalconference on knowledge discovery and data mining, pages 135–144, 2017.

[3]Yoon Kim.Convolutional neural networks for sentence classification.arXiv preprintarXiv:1408.5882, 2014.

[4]Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. InInternational conference on machine learning, pages 1188–1196, 2014.

[5]Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systemshandbook. InRecommender systems handbook, pages 1–35. Springer, 2011.

[6]Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques.Advancesin artificial intelligence, 2009, 2009.

[7]Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: Asurvey and new perspectives.ACM Computing Surveys (CSUR), 52(1):1–38, 2019.