Meeting of the LF AI & Data Technical Advisory Council (TAC)

June 29, 2023

LF AI & DATA

Antitrust Policy

- Linux Foundation meetings involve participation by industry competitors, and it is the intention of the Linux Foundation to conduct all of its activities in accordance with applicable antitrust and competition laws. It is therefore extremely important that attendees adhere to meeting agendas, and be aware of, and not participate in, any activities that are prohibited under applicable US state, federal or foreign antitrust and competition laws.

- Examples of types of actions that are prohibited at Linux Foundation meetings and in connection with Linux Foundation activities are described in the Linux Foundation Antitrust Policy available at http://www.linuxfoundation.org/antitrust-policy. If you have questions about these matters, please contact your company counsel, or if you are a member of the Linux Foundation, feel free to contact Andrew Updegrove of the firm of Gesmer Undergone LLP, which provides legal counsel to the Linux Foundation.

Recording of Calls

Reminder:

TAC calls are recorded and available for viewing on the TAC Wiki

LF AI & DATA

Reminder: LF AI & Data Useful Links

- Web site: lfaidata.foundation
- Wiki: wiki.lfaidata.foundation
- GitHub: github.com/lfaidata
- Landscape: https://landscape.lfaidata.foundation or https://l.lfaidata.foundation
- Mail Lists: https://lists.lfaidata.foundation
- Slack: https://slack.lfaidata.foundation
- Youtube: https://www.youtube.com/channel/UCfasaeqXJBCAJMNO9HcHfbA
- LF AI Logos: https://github.com/lfaidata/artwork/tree/master/lfaidata
- LF AI Presentation Template: https://drive.google.com/file/d/1eiDNJvXCqSZHT4Zk_-czASlz2GTBRZk2/view?usp=sharing

- Events Page on LF AI Website: https://lfaidata.foundation/events/
- Events Calendar on LF AI Wiki (subscribe available): https://wiki.lfaidata.foundation/pages/viewpage.action?pageId=12091544
- Event Wiki Pages: https://wiki.lfaidata.foundation/display/DL/LF+AI+Data+Foundation+Events

Agenda

- Roll Call (1 mins)
- Approval of Minutes from previous meeting (2 mins)
- Recommenders from Microsoft (40 minutes)
- Open Discussion

TAC Voting Members - Please note

Please ensure that you do the following to facilitate smooth procedural quorum and voting processes:

- Change your Zoom display name to include your First/Last Name, Company/Project Represented
  - example: Nancy Rausch, SAS
- State your First/Last Name and Company/Project when submitting a motion
  - example: First motion, Nancy Rausch/SAS

TAC Voting Members - Please note

- TAC members must attend consistently to maintain their voting status
- After 2 absences voting members will lose voting privileges
- Voting privileges will only be reinstated after attending 2 meetings in a row

TAC Voting Members

Note: we still need a few designated backups specified on wiki

Member Company or Graduated Project	Membership Level or Project Level	Voting Eligibility	Country	TAC Representative	Designated TAC Representative Alternates
4paradigm	Premier	Voting Member	China	Zhongyi Tan	
Baidu	Premier	Voting Member	China	Jun Zhang	Daxiang Dong, Yanjun Ma
Ericsson	Premier	Voting Member	Sweden	Rani Yadav-Ranjan	
Huawei	Premier	Voting Member	China	Howard (Huang Zhipeng)	Charlotte (Xiaoman Hu), Leon (Hui Wang)
Nokia	Premier	Voting Member	Finland	@Michael Rooke	@Jonne Soininen
OPPO	Premier	Voting Member	China	Jimmy (Hongmin Xu)	
SAS	Premier	Voting Member	USA	*Nancy Rausch	Liz McIntosh
ZTE	Premier	Voting Member	China	Wei Meng	Liya Yuan
Adversarial Robustness Toolbox Project	Graduated Technical Project	Voting Member	USA	Beat Buesser	Kevin Eykholt
Angel Project	Graduated Technical Project	Voting Member	China	Jun Yao	
Egeria Project	Graduated Technical Project	Voting Member	UK	Mandy Chessell	Nigel Jones, David Radley, Maryna Strelchuk, Ljupcho Palashevski, Chris Grote
Flyte Project	Graduated Technical Project	Voting Member	USA	Ketan Umare	
Horovod Project	Graduated Technical Project	Voting Member	USA	Travis Addair	
Milvus Project	Graduated Technical Project	Voting Member	China	Xiaofan Luan	Jun Gu
ONNX Project	Graduated Technical Project	Voting Member	USA	Alexandre Eichenberger	Andreas Fehlner, Prasanth Pulavarthi, Jim Spohrer
Pyro Project	Graduated Technical Project	Voting Member	USA	Fritz Obermeyer	

LF AI & DATA

Minutes approval

LF AI & DATA

Approval of June 15, 2023 Minutes

Draft minutes from the June 15 TAC call were previously distributed to the TAC members via the mailing list

Proposed Resolution:

- That the minutes of the June 15 meeting of the Technical Advisory Council of the LF AI & Data Foundation are hereby approved.

Recommenders @ LF AI

Miguel Fierro (representing maintainers of recommenders at Microsoft)

Recommendations everywhere

Why contribute Recommenders to the LF

Neutral holding ground

- Vendor-neutral, not for profit

Open governance model

- Transparent and open governance model
- Instill trust in contributors and adopters in the management of the project
- Neutral management of projects' assets by the foundation

Growing community

- Increase visibility of project through LF ecosystem
- Increase contributors by converting new & existing users
- Opportunities to collaborate with other hosted projects

Recommendation Systems in Modern Business and Academic Research

“35% of what consumers purchase on Amazon and 75% of what they watch on Netflix come from recommendations algorithms”

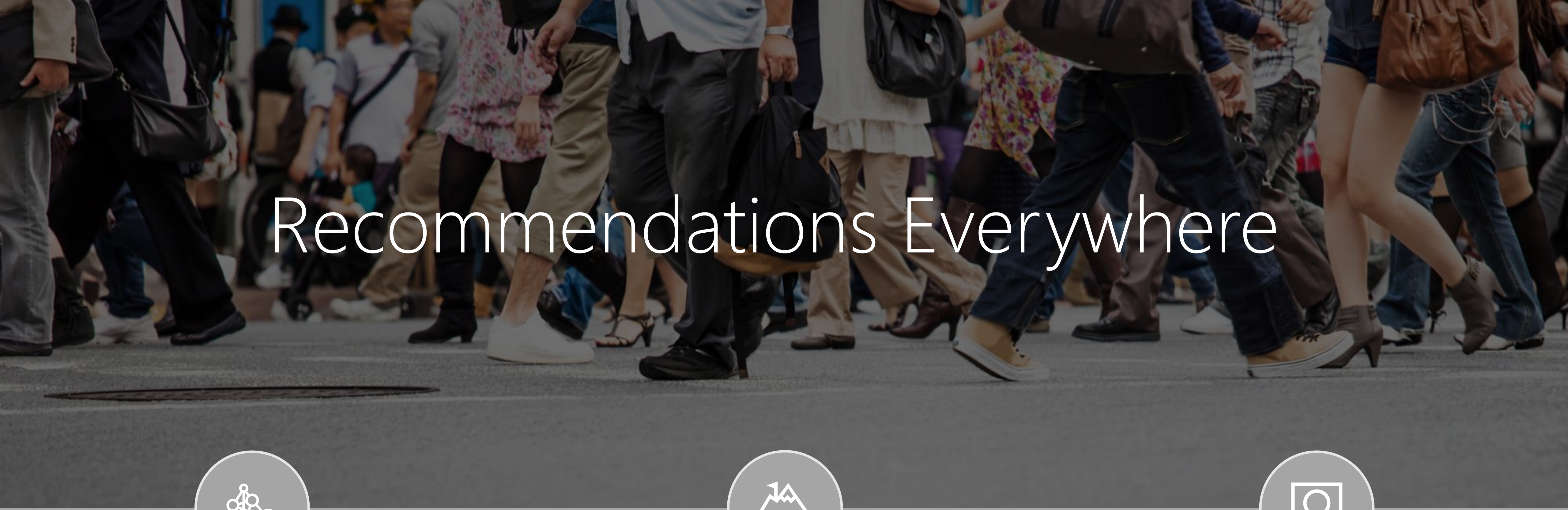
McKinsey & Co

“60% of video clicks on the YouTube homepage comes from recommendations”

Emerj

“BestBuy reported an online sales growth of 23.7% due to a speedier checkout process, better navigation, and relevant product recommendations”

CNBC



Recommendations Everywhere

Brand/news/product recommendation

Indirectly drive revenue by increasing customer engagement, networking effect, etc.

Business metric prediction

Directly drive revenue through ad clicks, internet traffics, etc.

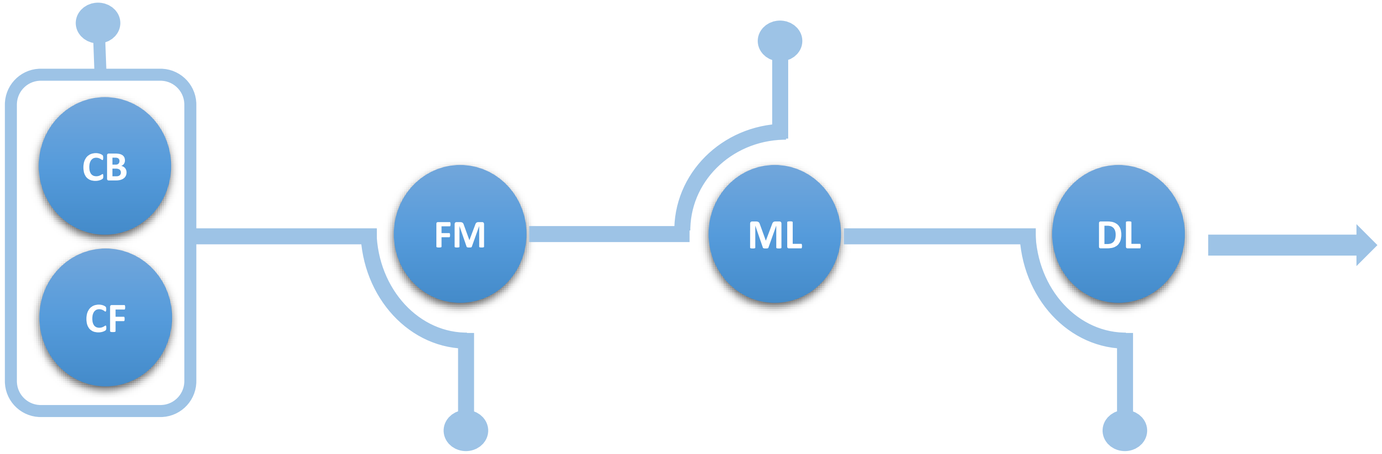
Customer segmentation and personalization

Indirectly drive revenue by precisely reaching customers with market campaign or product.

History

1990s (Tapestry, GroupLens)
Content based filtering
Collaborative filtering

2010 (Various data competitions)
Hybrid models with machine learning
LR, FM, GBDT, etc.
Pair-wise ranking



2006 (Netflix prize)
Factorization-based Models
SVD++

2015 (Deep learning)
Flourish with neural models
PNN, Wide&Deep, DeepFM, xDeepFM, etc.

Explainable recommendation
Knowledge enhanced recommendation
Reinforcement learning
Transfer learning
...

Challenges in Recommendation Systems

Limited resource	Fragmented solutions	Fast-growing area
There is *limited* reference and guidance to build a recommender system on scale to support enterprise-grade scenarios	Packages/tools/modules off-the-shelf are very fragmented, not scalable, and not well compatible with each other	New algorithms sprout every day – not many people have such expertise to implement and deploy a recommender by using the state-of-the-arts algorithms

Description of Recommenders

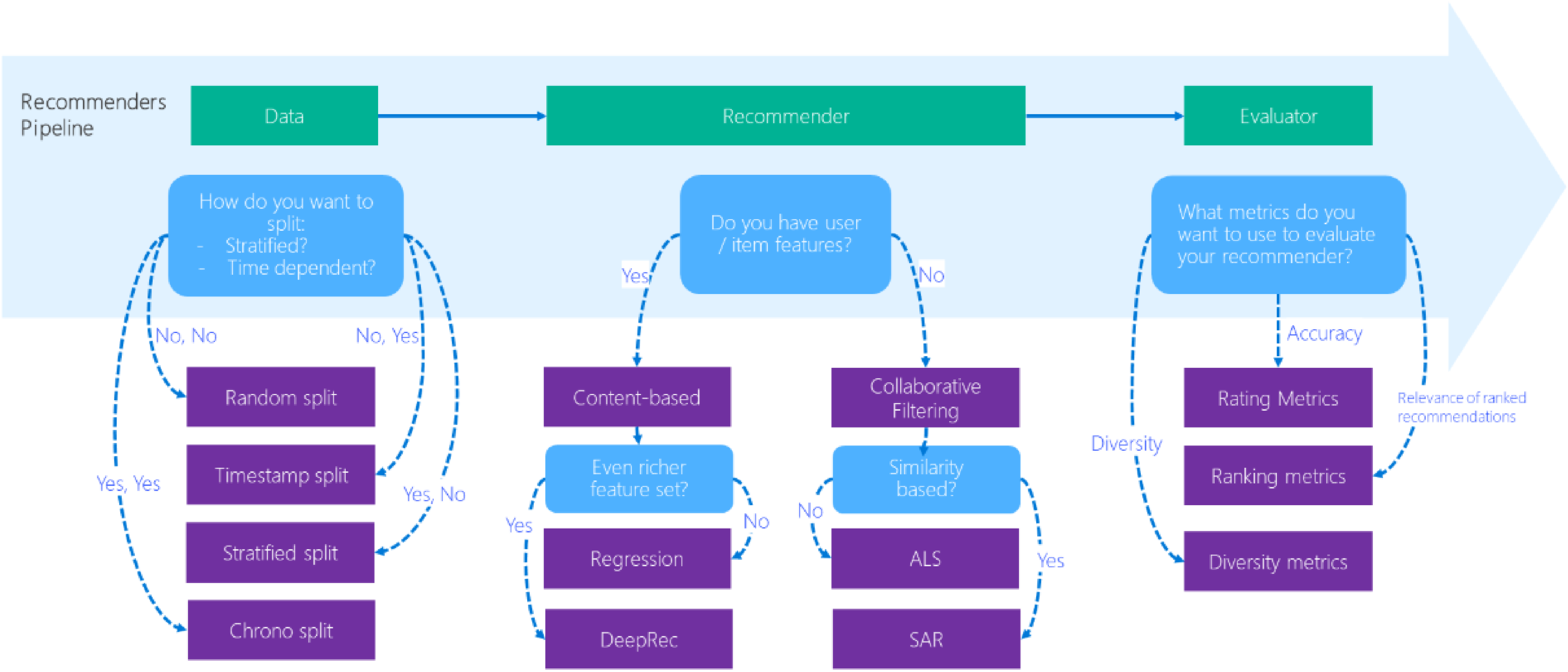
What is Recommenders

- Collaborative development efforts of Microsoft Cloud & AI data scientists, Microsoft Research researchers, academia researchers etc.
- Github url: https://github.com/Microsoft/Recommenders
- Contents
  - Utilities: modular functions for model creation, data manipulation, evaluation etc.
  - Algorithms: SVD, SAR, ALS, NCF, Wide&Deep, xDeepFM, DKN etc.
  - Notebooks: how-to examples for building end-to-end recommendation systems

Goals of Recommenders

- "Taking recommendation technology to the masses"
- Helping researchers and developers to quickly select, prototype, demonstrate, and productionize a recommender system
- Accelerating enterprise-grade development and deployment of a recommender system into production
- Systematic overview of the recommendation technology from a pragmatic perspective
- State-of-the-art academic research in recommendation algorithms
- Best practices (with example codes) in developing recommender systems

Best practice workflow



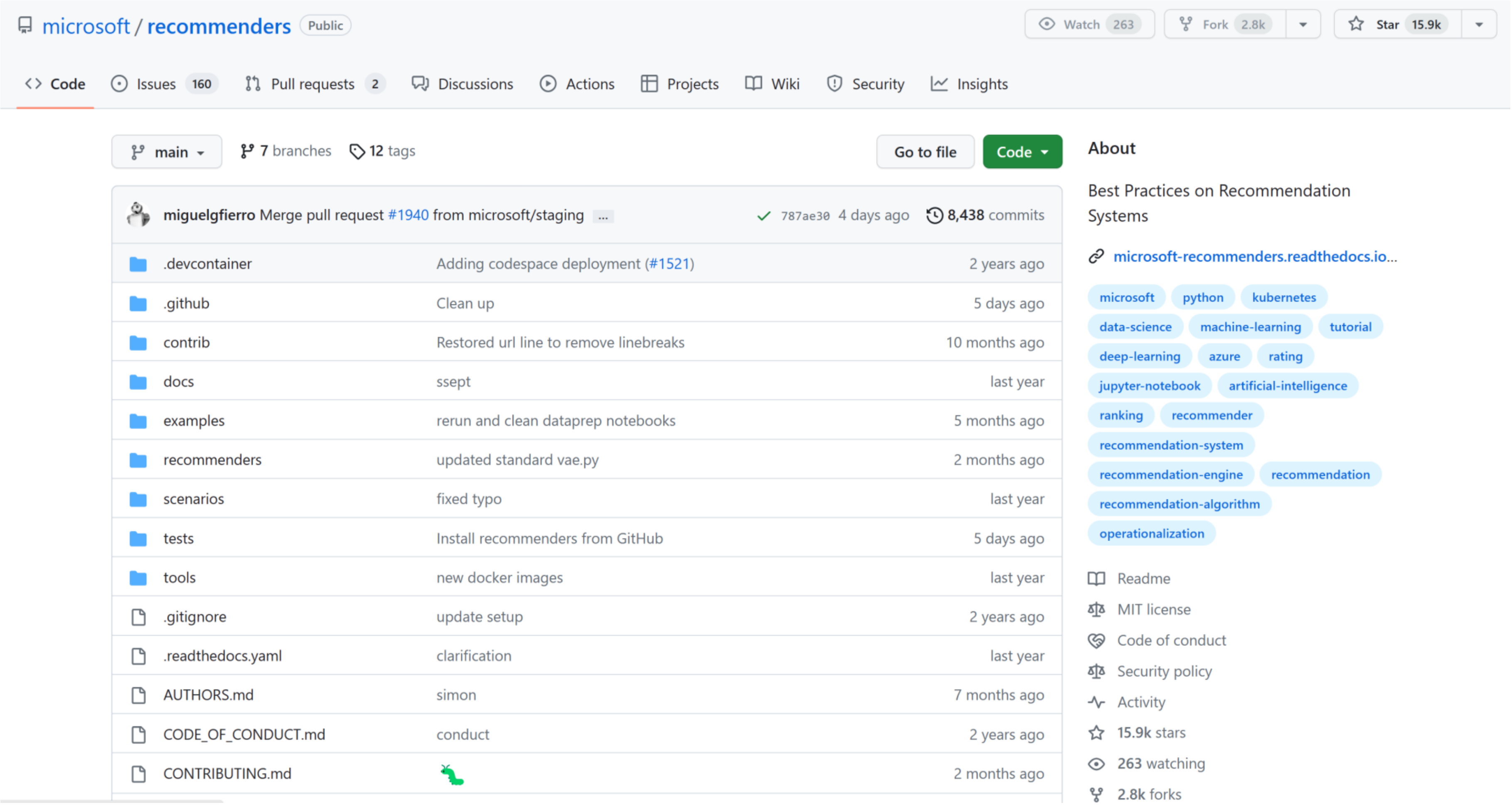
30+ recommendation algorithms

- Alternating Least Squares (ALS)
- Attentive Asynchronous Singular Value Decomposition (A2SVD)
- Cornac/Bayesian Personalized Ranking (BPR)
- Cornac/Bilateral Variational Autoencoder (BiVAE)
- Convolutional Sequence Embedding Recommendation (Caser)
- Deep Knowledge-Aware Network (DKN)
- Extreme Deep Factorization Machine (xDeepFM)
- FastAI Embedding Dot Bias (FAST)
- LightFM/Hybrid Matrix Factorization
- LightGBM/Gradient Boosting Tree
- LightGCN
- GeoIMC
- GRU4Rec
- Multinomial VAE
- Neural Recommendation with Long- and Short-term User Representations (LSTUR)
- Neural Recommendation with Attentive Multi-View Learning (NAML)
- Neural Collaborative Filtering (NCF)
- Neural Recommendation with Personalized Attention (NPA)
- Neural Recommendation with Multi-Head Self-Attention (NRMS)
- Next Item Recommendation (NextItNet)
- Restricted Boltzmann Machines (RBM)
- Riemannian Low-rank Matrix Completion (RLRMC)
- Simple Algorithm for Recommendation (SAR)
- Self-Attentive Sequential Recommendation (SASRec)
- Short-term and Long-term Preference Integrated Recommender (SLi-Rec)
- Multi-Interest-Aware Sequential User Modeling (SUM)
- Sequential Recommendation Via Personalized Transformer (SSEPT)
- Standard VAE
- Surprise/Singular Value Decomposition (SVD)
- Term Frequency - Inverse Document Frequency (TF-IDF)
- Vowpal Wabbit (VW)
- Wide and Deep
- xLearn/Factorization Machine (FM) & Field-Aware FM (FFM)

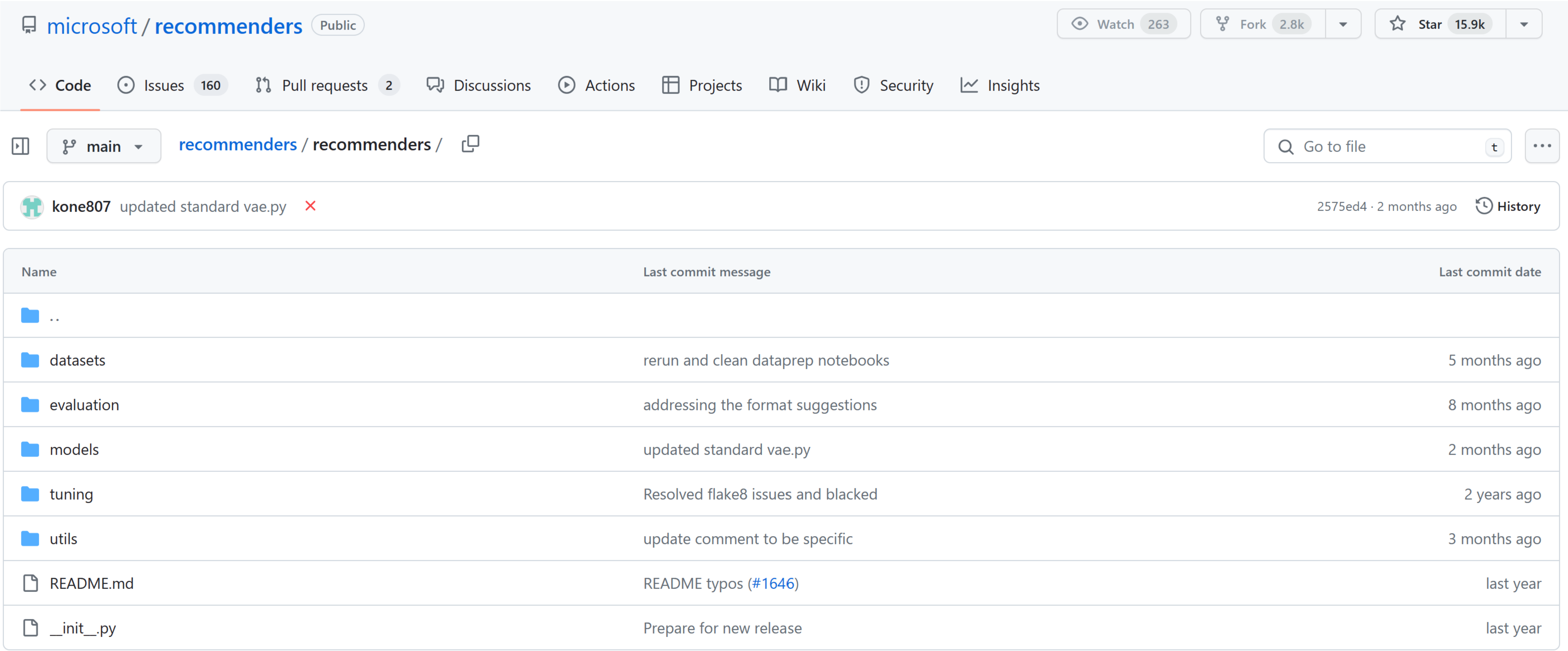
Types of Algorithms in Recommenders

	Collaborative filtering	Content-based filtering	Hybrid
Python	SAR, SVD	LightGBM	
Python + Spark	ALS	LightGBM	
Python + GPU	NCF, FastAI, RBM	Wide and Deep	xDeepFM, DKN

Recommenders Repository



Recommenders Library



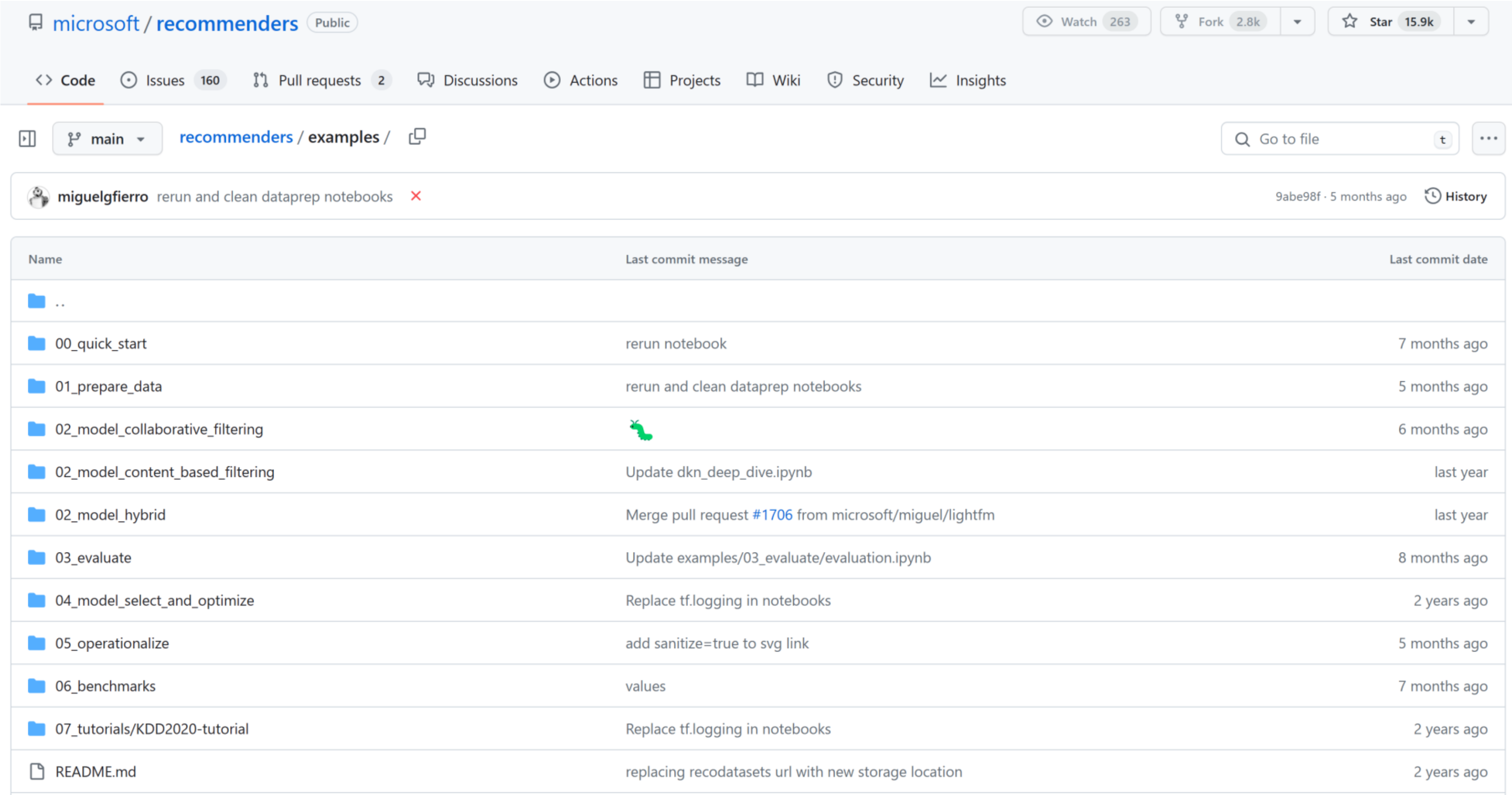
Example Class

recommenders / recommenders / models / ncf / ncf_singlenode.py

Code Blame 450 lines (373 loc) · 15.7 KB

```python
class NCF:
    def fit(self, data):
        """Fit model with training data

        Args:
            data (NCFDataset): initilized Dataset in ./dataset.py
        """

        # get user and item mapping dict
        self.user2id = data.user2id
        self.item2id = data.item2id
        self.id2user = data.id2user
        self.id2item = data.id2item

        # loop for n_epochs
        for epoch_count in range(1, self.n_epochs + 1):

            # negative sampling for training
            train_begin = time()

            # initialize
            train_loss = []

            # calculate loss and update NCF parameters
            for user_input, item_input, labels in data.train_loader(self.batch_size):

                user_input = np.array([self.user2id[x] for x in user_input])
                item_input = np.array([self.item2id[x] for x in item_input])
                labels = np.array(labels)

                feed_dict = {
                    self.user_input: user_input[..., None],
                    self.item_input: item_input[..., None],
                    self.labels: labels[..., None],
                }

                # get loss and execute optimization
                loss, _ = self.sess.run([self.loss, self.optimizer], feed_dict)
                train_loss.append(loss)
            train_time = time() - train_begin
```

Recommenders Notebook Examples



Name	Last commit message	Last commit date
..		
00_quick_start	rerun notebook	7 months ago
01_prepare_data	rerun and clean dataprep notebooks	5 months ago
02_model_collaborative_filtering	🐛	6 months ago
02_model_content_based_filtering	Update dkn_deep_dive.ipynb	last year
02_model_hybrid	Merge pull request #1706 from microsoft/miguel/lightfm	last year
03_evaluate	Update examples/03_evaluate/evaluation.ipynb	8 months ago
04_model_select_and_optimize	Replace tf.logging in notebooks	2 years ago
05_operationalize	add sanitize=true to svg link	5 months ago
06_benchmarks	values	7 months ago
07_tutorials/KDD2020-tutorial	Replace tf.logging in notebooks	2 years ago
README.md	replacing recodatasets url with new storage location	2 years ago

Value of Notebook Examples

- Notebooks can be used as a starting point for data scientists.
- Data scientists can replace the dataset downloaded in the notebook with their own and easily get a recommendation system up and running.
- They offer an efficient way to implement these algorithms for research purposes, as POCs or in production.

Example Structure: Description + Code

main · recommenders / examples / 02_model_collaborative_filtering / ncf_deep_dive.ipynb

Preview | Code | Blame | 1149 lines (1149 loc) · 36 KB | Raw

1.1 The GMF model

In ALS, the ratings are modeled as follows:

$$\hat{r}_{u,i} = q_i^T p_u$$

GMF introduces a neural CF layer as the output layer of standard MF. In this way, MF can be easily generalized and extended. For example, if we allow the edge weights of this output layer to be learnt from data without the uniform constraint, it will result in a variant of MF that allows varying importance of latent dimensions. And if we use a non-linear function for activation, it will generalize MF to a non-linear setting which might be more expressive than the linear MF model. GMF can be shown as follows:

$$\hat{r}_{u,i} = a_{out}\left(h^T\left(q_i \odot p_u\right)\right)$$

where $\odot$ is element-wise product of vectors. Additionally, a_{out} and h denote the activation function and edge weights of the output layer respectively. MF can be interpreted as a special case of GMF. Intuitively, if we use an identity function for a_{out} and enforce h to be a uniform vector of 1, we can exactly recover the MF model.

1.2 The MLP model

NCF adopts two pathways to model users and items: 1) element-wise product of vectors, 2) concatenation of vectors. To learn interactions after concatenating of users and items latent features, the standard MLP model is applied. In this sense, we can endow the model a large level of flexibility and non-linearity to learn the interactions between p_u and q_i. The details of MLP model are:

For the input layer, there is concatenation of user and item vectors:

$$z_1 = \phi_1\left(p_u, q_i\right) = \begin{bmatrix} p_u \\ q_i \end{bmatrix}$$

So for the hidden layers and output layer of MLP, the details are:

$$\phi_l\left(z_l\right) = a_{out}\left(W_l^T z_l + b_l\right), (l = 2, 3, \ldots, L-1)$$

and:

$$\hat{r}_{u,i} = \sigma\left(h^T \phi\left(z_{L-1}\right)\right)$$

where W_l, b_l, and a_{out} denote the weight matrix, bias vector, and activation function for the l-th layer's perceptron, respectively. For activation functions of MLP layers, one can freely choose sigmoid, hyperbolic tangent (tanh), and Rectifier (ReLU), among others. Because

Example Structure: Description + Code

3.1 Load and split data

To evaluate the performance of item recommendation, we adopt the leave-one-out evaluation.

For each user, we held out his/her last interaction as the test set and utilized the remaining data for training. Since it is too time-consuming to rank all items for every user during evaluation, we followed the common strategy that randomly samples 100 items that are not interacted by the user, ranking the test item among the 100 items. Our test samples will be constructed by `NCFDataset`.

We also show an alternative evaluation method, splitting the data chronologically using `python_chrono_split` to achieve a 75/25% training and test split.

In [3]:

```
df = movielens.load_pandas_df(
    size=MOVIELENS_DATA_SIZE,
    header=["userID", "itemID", "rating", "timestamp"]
)

df.head()
```

```
100%|██████████| 4.81k/4.81k [00:00<00:00, 16.9kKB/s]
```

Out[3]:

	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

In [4]:

```
train, test = python_chrono_split(df, 0.75)
```

Example Structure: Description + Code

3.3 Train NCF based on TensorFlow

The NCF has a lot of parameters. The most important ones are:

`n_factors`, which controls the dimension of the latent space. Usually, the quality of the training set predictions grows with as n_factors gets higher.

`layer_sizes`, sizes of input layer (and hidden layers) of MLP, input type is list.

`n_epochs`, which defines the number of iteration of the SGD procedure. Note that both parameter also affect the training time.

`model_type`, we can train single `"MLP"`, `"GMF"` or combined model `"NCF"` by changing the type of model.

We will here set `n_factors` to 4, `layer_sizes` to `[16,8,4]`, `n_epochs` to 100, `batch_size` to 256. To train the model, we simply need to call the `fit()` method.

In [ ]:

```
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
)
```

In [10]:

```
with Timer() as train_time:
    model.fit(data)

print("Took {} seconds for training.".format(train_time.interval))
```

```
Took 615.3995804620008 seconds for training.
```

Example Structure: Description + Code

3.4.2 Generic Evaluation

We remove rated movies in the top k recommendations To compute ranking metrics, we need predictions on all user, item pairs. We remove though the items already watched by the user, since we choose not to recommend them again.

In [12]:

```
with Timer() as test_time:

    users, items, preds = [], [], []
    item = list(train.itemID.unique())
    for user in train.userID.unique():
        user = [user] * len(item)
        users.extend(user)
        items.extend(item)
        preds.extend(list(model.predict(user, item, is_list=True)))

    all_predictions = pd.DataFrame(data={"userID": users, "itemID":items, "prediction":preds})

    merged = pd.merge(train, all_predictions, on=["userID", "itemID"], how="outer")
    all_predictions = merged[merged.rating.isnull()].drop('rating', axis=1)

print("Took {} seconds for prediction.".format(test_time.interval))
```

```
Took 2.7729760599977453 seconds for prediction.
```

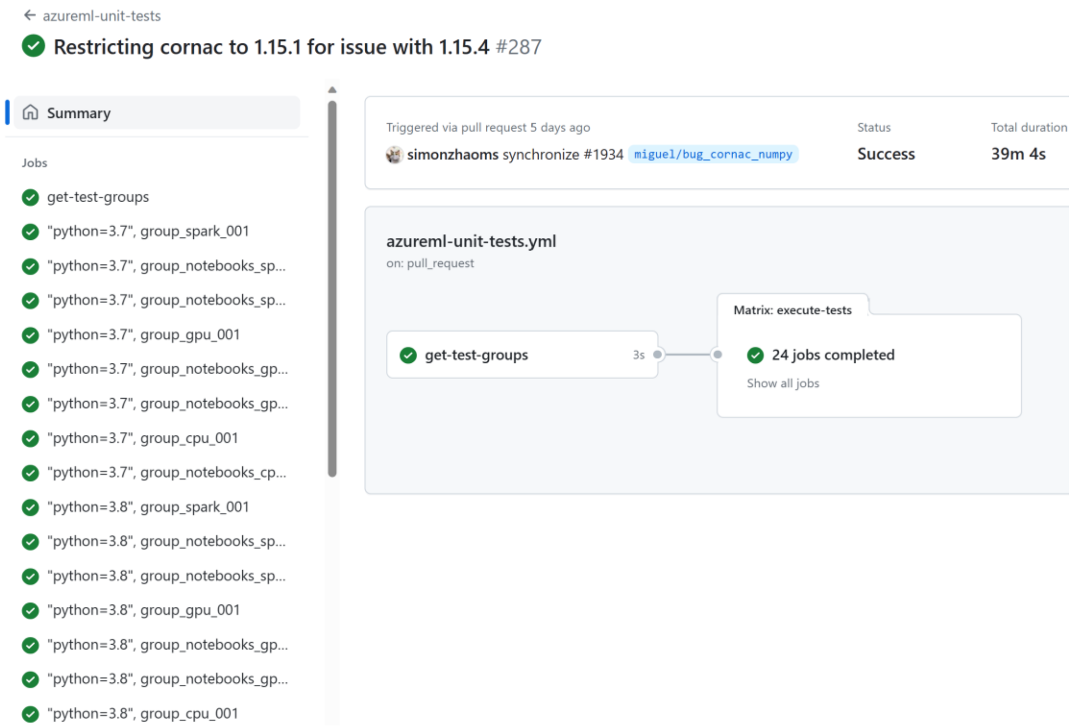
In [13]:

```
eval_map = map_at_k(test, all_predictions, col_prediction='prediction', k=TOP_K)
eval_ndcg = ndcg_at_k(test, all_predictions, col_prediction='prediction', k=TOP_K)
eval_precision = precision_at_k(test, all_predictions, col_prediction='prediction', k=TOP_K)
eval_recall = recall_at_k(test, all_predictions, col_prediction='prediction', k=TOP_K)

print("MAP:\t%f" % eval_map,
      "NDCG:\t%f" % eval_ndcg,
      "Precision@K:\t%f" % eval_precision,
      "Recall@K:\t%f" % eval_recall, sep='\n')
```

```
MAP:	0.048144
NDCG:	0.198384
Precision@K:	0.176246
Recall@K:	0.098700
```

Recommenders Tests

PR gates are tests executed after doing a pull request and they should be quick. The objective is to validate that the code is not breaking before merging it.



The nightly builds are tests executed asynchronously and can take hours. Some tests take so long that they cannot be executed in a PR gate, therefore they are executed asynchronously in the nightly builds.

Build Type	Branch	Status		Branch	Status
Linux CPU	main	azureml-cpu-nightly passing		staging	azureml-cpu-nightly passing
Linux GPU	main	azureml-gpu-nightly passing		staging	azureml-gpu-nightly passing
Linux Spark	main	azureml-spark-nightly passing		staging	azureml-spark-nightly passing

Test Categories

- **Data validation tests**: In the data validation tests, we ensure that the schema for input and output data for each function in the pipeline matches the desired prespecified schema, that the data is available and has the correct size.
- **Unit tests**: In the unit tests we just make sure the python utilities and notebooks run correctly. Unit tests are fast, ideally less than 5min and are run in every pull request
- **Functional tests**: These tests make sure that the components of the project not just run but their function is correct. For example, we want to test that an ML model evaluation of RMSE gives a positive number.
- **Integration tests**: We want to make sure that the interaction between different components is correct. For example, the interaction between data ingestion pipelines and the compute where the model is trained, or between the compute and a database.
- **Smoke tests**: The smoke tests are gates to the slow tests in the nightly builds to detect quick errors. If we are running a test with a large dataset that takes 4h, we want to create a faster version of the large test (maybe with a small percentage of the dataset or with 1 epoch) to ensure that it runs end-to-end without obvious failures. Smoke tests can run sequentially with functional or integration tests in the nightly builds, and should be fast, ideally less than 20min.
- **Performance test**: The performance tests are tests that measure the computation time or memory footprint of a piece of code and make sure that this is bounded between some limits.
- **Responsible AI tests**: Responsible AI tests are test that enforce fairness, transparency, explainability, human-centeredness, and privacy.
- **Security tests**: Security tests are tests that make sure that the code is not vulnerable to attacks. These can detect potential security issues either in python packages or the underlying OS, in addition to scheduled scans in the production pipelines.
- **Regression tests**: In some situations, we are migrating from a deprecated version to a new version of the code, or maybe we are maintaining two versions of the same library (e.g. TensorFlow v1 and v2). Regression tests make sure that the code works in both versions of the code. These types of tests sometimes are done locally, before upgrading to the new version, or they can be included in the tests pipelines if we want to execute them recurrently.

Recommenders Coding Guidelines

- Test Driven Development
- Do not Repeat Yourself
- Single Responsibility
- Python and Docstrings Style
- The Zen of Python
- Evidence-Based Software Design
- You are not going to need it
- Minimum Viable Product
- Publish Often Publish Early
- If our code is going to fail, let it fail fast

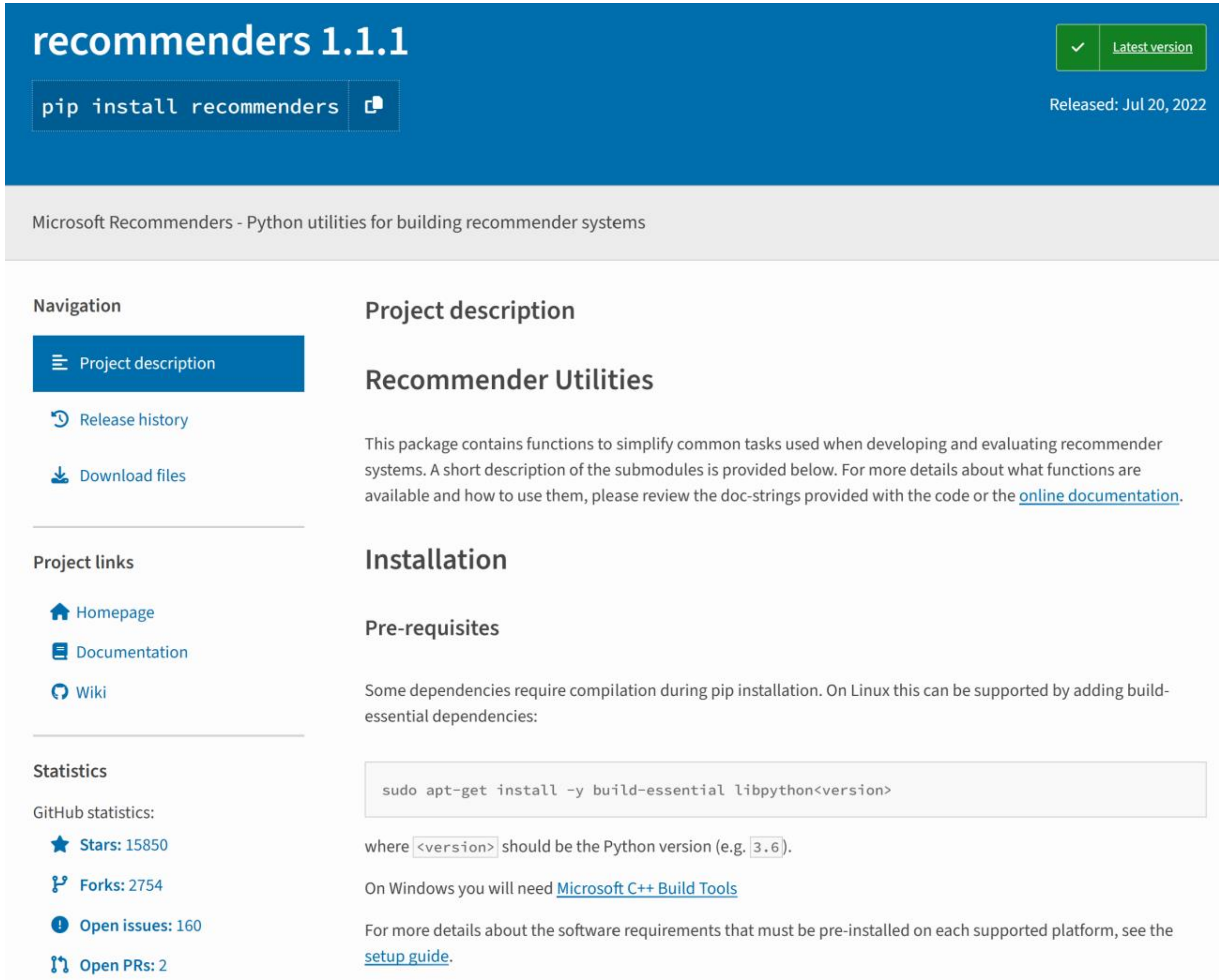
https://github.com/Microsoft/Recommenders/wiki/Coding-Guidelines

Options to Try Out Recommenders

Options	Prerequisite	Pros	Cons
Local machine (Linux/Windows/MacOS)	Jupyter notebook	Users can use tools they are familiar with in the local machine	Limited by the environment (e.g. OS) and hardware of the local machine (e.g. GPU)
Docker container	Docker	Portable and system-independent	Require Docker to be pre-installed
Remote machine (Linux/Windows)	Jupyter notebook	Users can build solutions remotely	More costly solution
Spark compute (Synapse/Databricks)	Spark compute	Efficient computation of big data workloads	More costly solution

Recommenders in the Community

Recommenders pip Package



recommenders 1.1.1

```
pip install recommenders
```

Latest version

Released: Jul 20, 2022

Microsoft Recommenders - Python utilities for building recommender systems

Navigation

- Project description
- Release history
- Download files

Project links

- Homepage
- Documentation
- Wiki

Statistics

GitHub statistics:

- **Stars:** 15850
- **Forks:** 2754
- **Open issues:** 160
- **Open PRs:** 2

Project description

Recommender Utilities

This package contains functions to simplify common tasks used when developing and evaluating recommender systems. A short description of the submodules is provided below. For more details about what functions are available and how to use them, please review the doc-strings provided with the code or the online documentation.

Installation

Pre-requisites

Some dependencies require compilation during pip installation. On Linux this can be supported by adding build-essential dependencies:

```
sudo apt-get install -y build-essential libpython<version>
```

where `<version>` should be the Python version (e.g. `3.6`).

On Windows you will need Microsoft C++ Build Tools

For more details about the software requirements that must be pre-installed on each supported platform, see the setup guide.

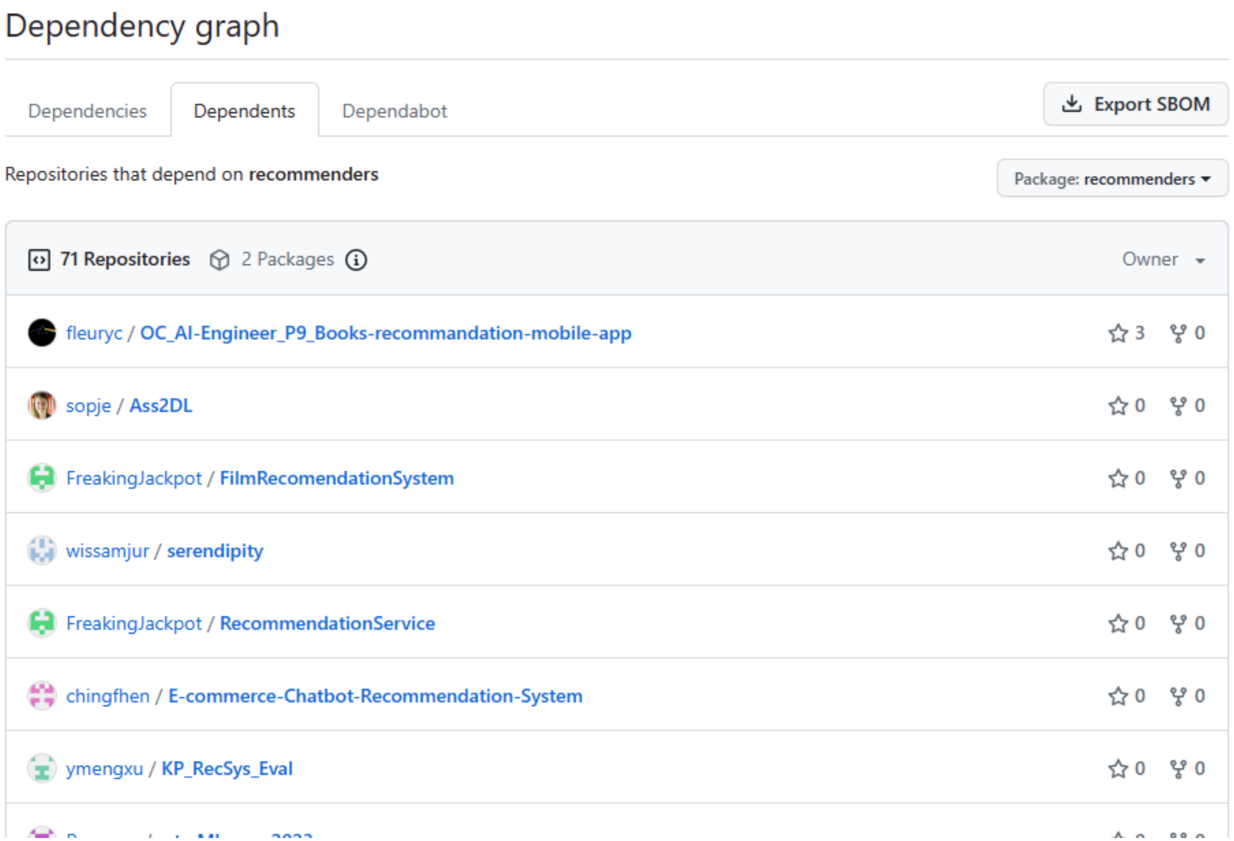
Engagement with the Community

- Collaborative development efforts of
  - Microsoft Cloud & AI data scientists
  - Microsoft Research researchers
  - academic researchers
  - data scientists from other enterprises

- 16K stars on GitHub, 2.8K forks.
- *Recommenders* is the most popular open-source repository in the field of recommendation systems.
- Used by academics who submit papers to RecSys (the top conference on recommendation systems) `https://github.com/ACMRecSys/recsys-evaluation-frameworks`
- Featured in *YC Hacker News, O'Reilly Data Newsletter, GitHub weekly trending list* etc.

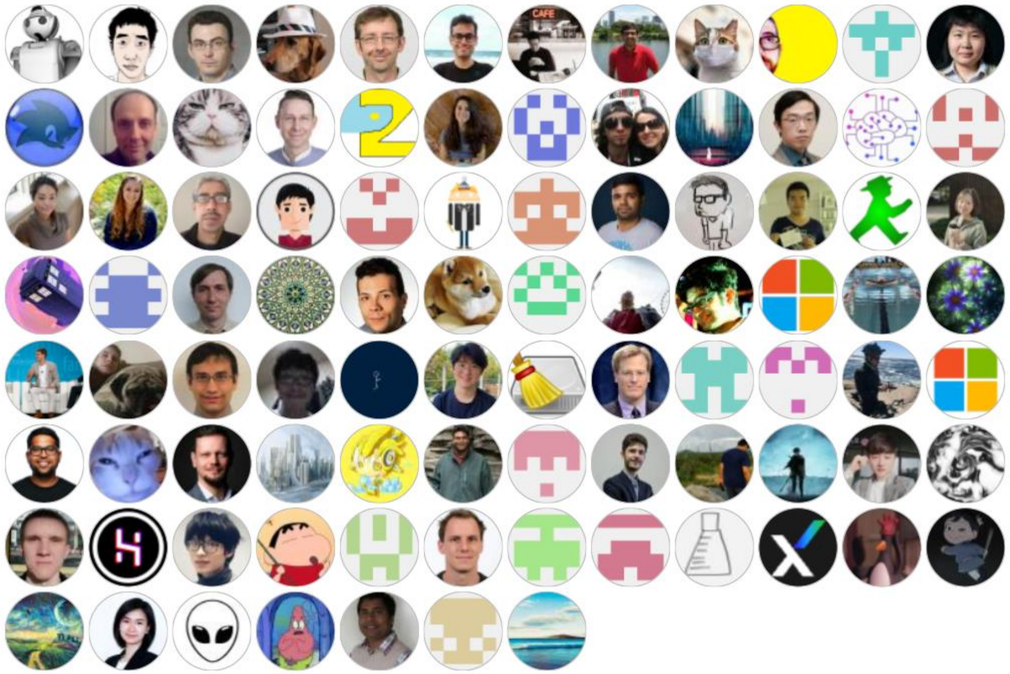
Engagement with the Community (contd.)

- Referenced in *paperswithcode.com, towardsdatascience.com* etc.
- 70 repositories (from outside Microsoft) depend on recommenders



Contributors

- 7 maintainers
- 89 contributors to date



Summary

Recommendation systems are ubiquitous in e-commerce and other industries.

Recommenders helps solve the challenge of easily building recommendation systems.

The repository is composed of a library, examples in the form of Jupyter notebooks and a test pipeline.

The open-source community has embraced and contributed to the repository.

Future Opportunities

Implement new cutting-edge algorithms
from recommendations research, LLMs etc.

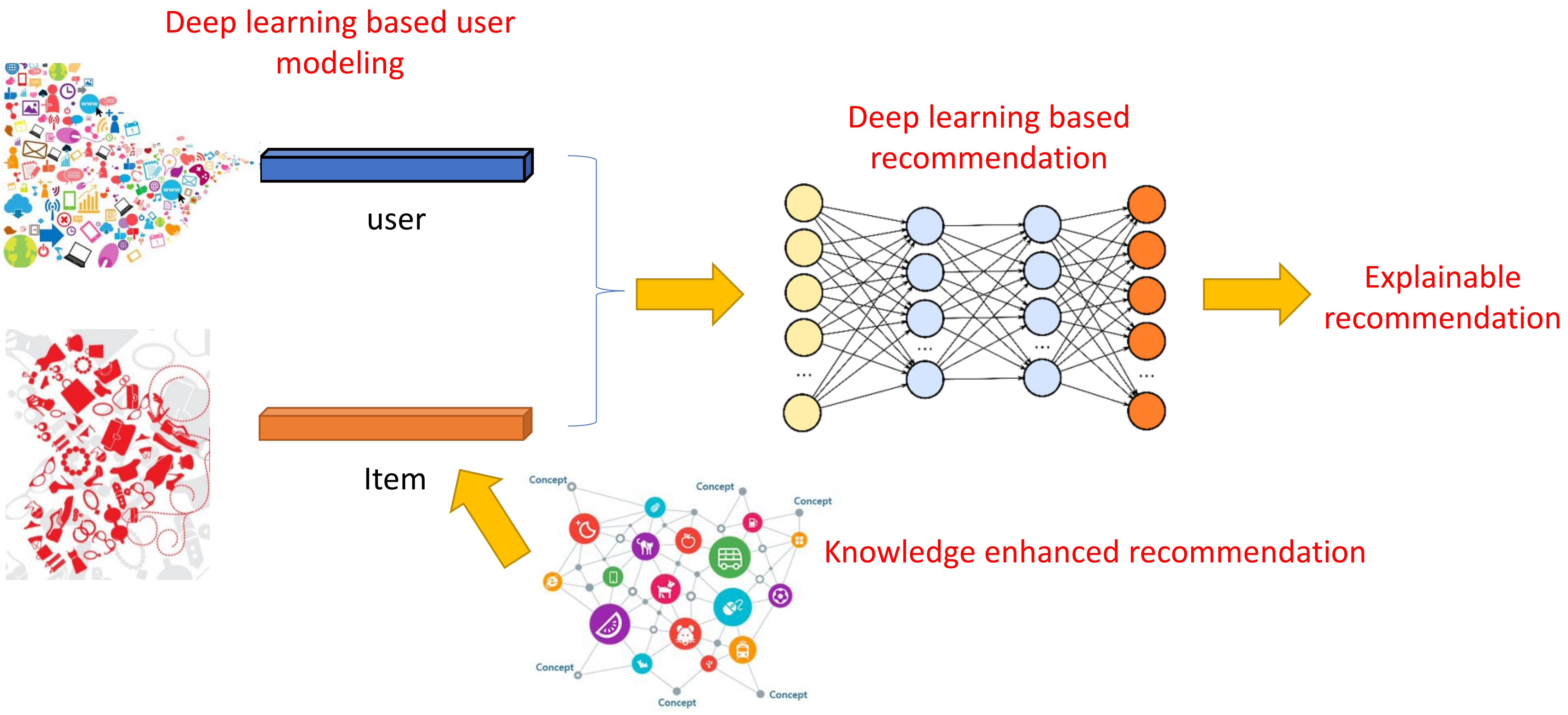
Performance improvements and upgrade of dependencies
(such as TensorFlow / PyTorch, Spark MLLib)

Customized examples for specific industry or research
scenarios

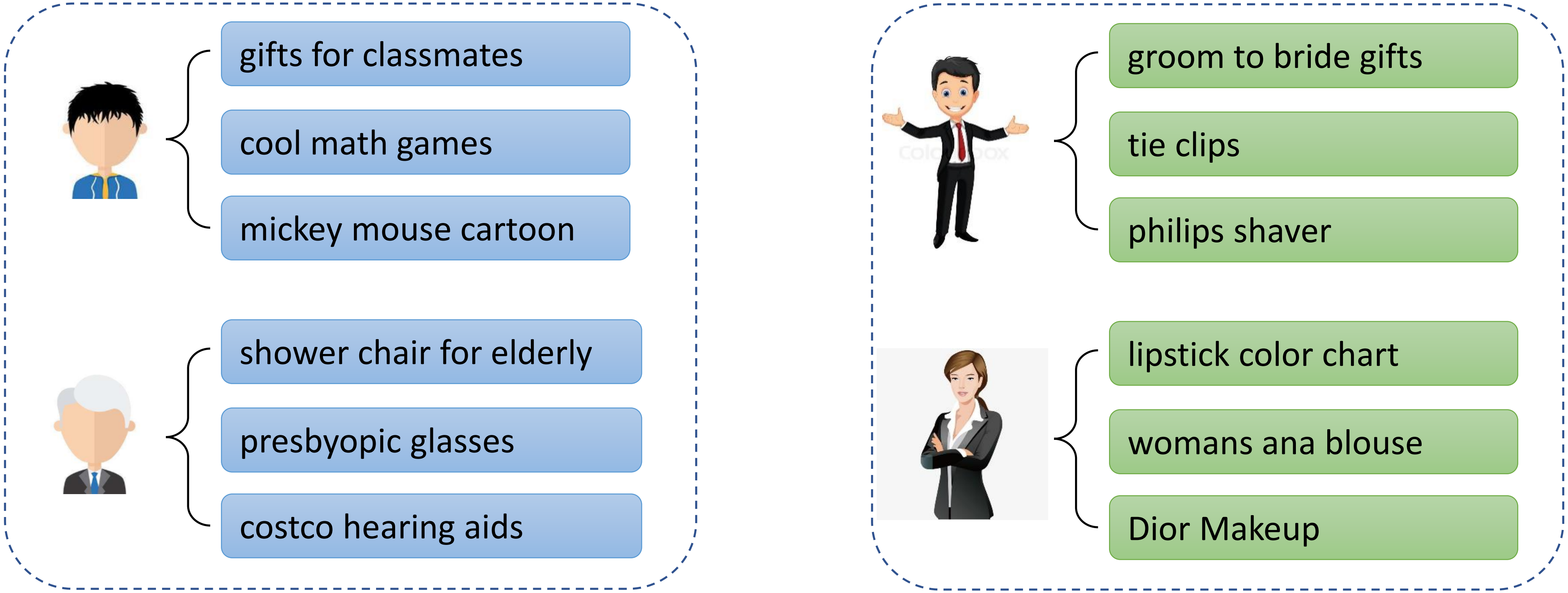
Thank you

Appendix

Algorithms from Microsoft Research



Query Log based User Modeling



Chuhan Wu, Fangzhao Wu, Junxin Liu, Shaojian He, Yongfeng Huang, Xing Xie, Neural Demographic Prediction using Search Query, WSDM 2019

Query Log based User Modeling

Different words may have different importance

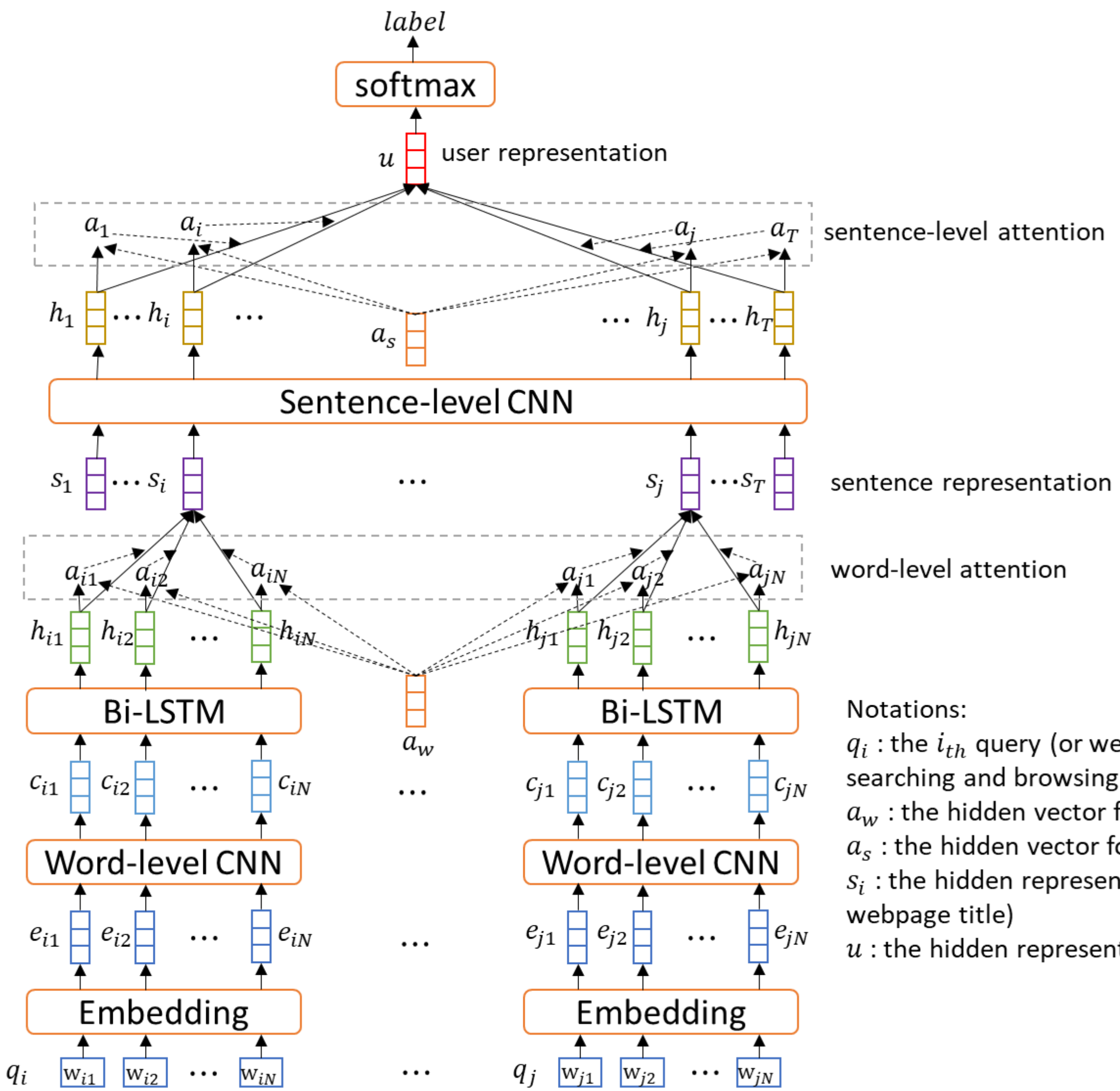
Different records have different informativeness

birthday gift for grandson
central garden street
google
my health plan
medicaid new York
medicaid for elderly in new York
alcohol treatment
amazon.com
documentary grandson
youtube

Neighboring records may have relatedness, while far ones usually not

The same word may have different importance in different contexts

Query Log based User Modeling



Notations:

q_i : the i_{th} query (or webpage title) from user's searching and browsing log

a_w : the hidden vector for word-level attention

a_s : the hidden vector for sentence-level attention

s_i : the hidden representation of the i_{th} query (or webpage title)

u : the hidden representation of the user

Explainable Recommendation Systems

Fog Harbor Fish House

4703 reviews

Their **tan tan noodles** are made of magic. The chili oil is really appetizing.

However, **prices** are on the high side.

1-800-FLOWERS.COM – Elegant Flowers for Lovers

Ad · 1800Flowers.com · 40,100+ followers on Twitter

Ratings: Product Selection 4.5/5 - Price 4/5 - Customer Service 4/5

1800flowers.com has been visited by 10K+ users in the past month

1800flowers.com is rated ★★★★★ (321,968 reviews)

Model Explainability { Transparency, Trust

Effectiveness, Persuasiveness, Readability } Presentation Quality

Feedback Aware Generative Model

- Traditional Seq2Seq model

$$\underset{\theta}{argmax} \prod_i p(y_i|x_i;\theta)$$

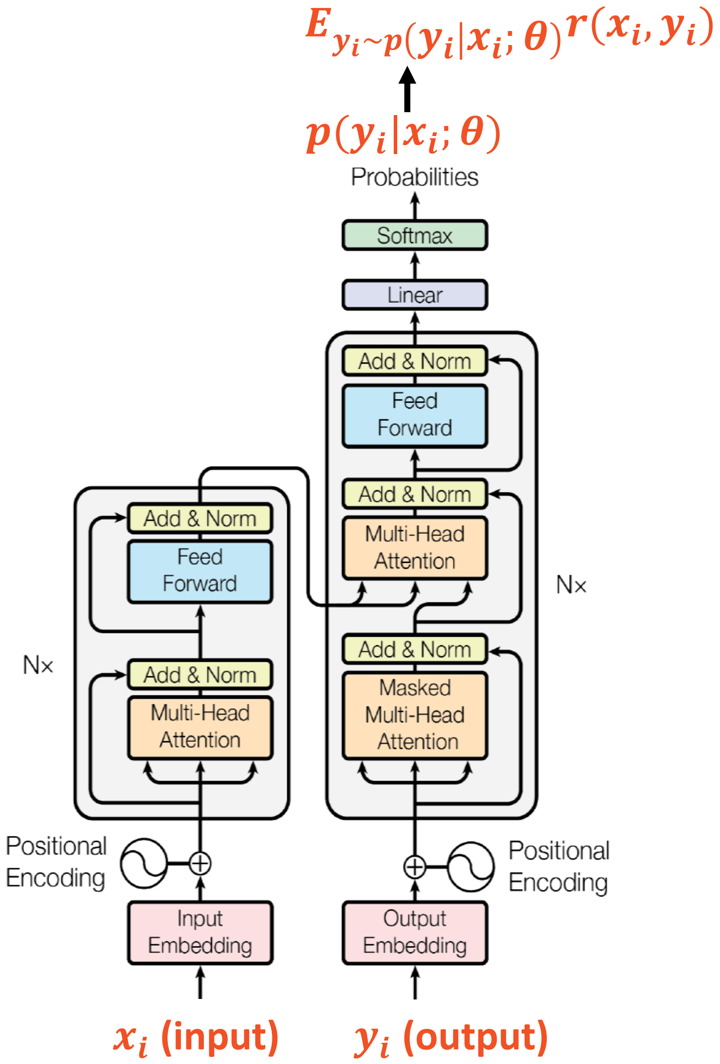
- Feedback aware model

$$\underset{\theta}{argmax} \sum_i E_{y_i \sim p(y_i|x_i;\theta)} r(x_i, y_i)$$

Input x_i	Output y_i	Reward $r(\cdot)$
Ad title, category, keyword, sitelink title	Ad title, Ad description, sitelink description	CTR

Ad title: *Flowers delivered today*
Category: *Occasions & Gifts*

Elegant flowers for any occasion.
100% smile guarantee!



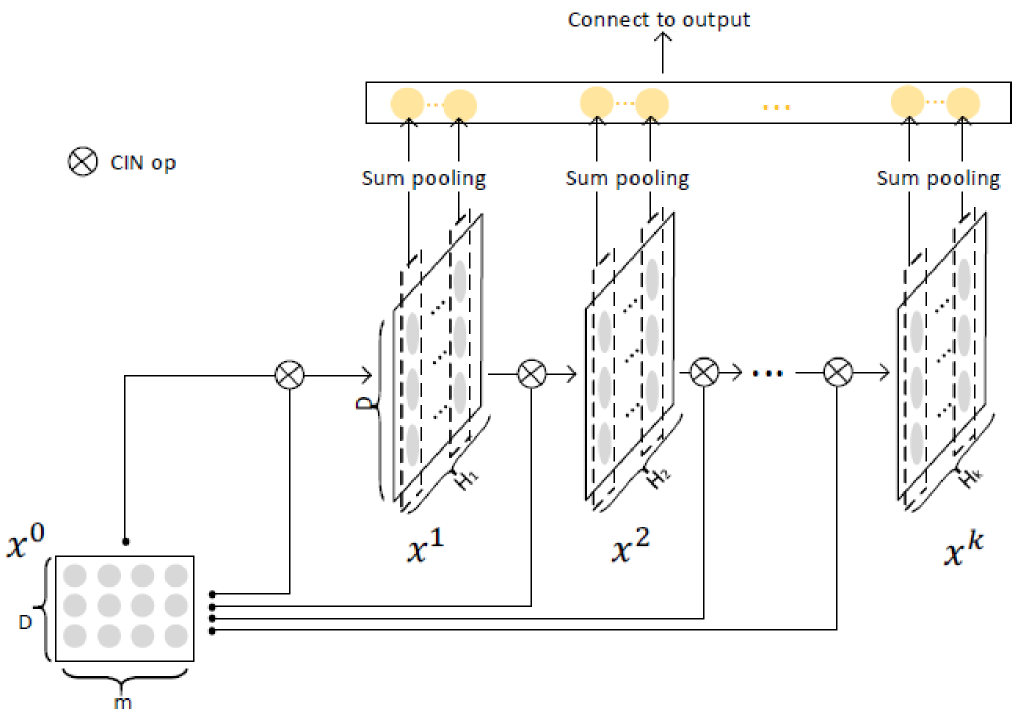
Extreme Deep Factorization Machine (xDeepFM)

➢Compressed Interaction Network (CIN)

- Hidden units at the k-th layer:

$$\mathbf{X}^k_{h,*} = \sum_{i=1}^{H_{k-1}} \sum_{j=1}^{m} \mathbf{W}^{k,h}_{ij} (\mathbf{X}^{k-1}_{i,*} \circ \mathbf{X}^0_{j,*})$$

m: # fields in raw data
D: dimension of latent space
H_k: # feature maps in the k-th hidden layer
x^0 : input data
x^k: states of the k-th hidden layer



➢Properties

- Compression: reduce interaction space from $O(mH_{k-1})$ down to $O(H_k)$
- Keep the form of vectors
  - Hidden layers are matrices, rather than vectors
- Degree of feature interactions increases with the depth of layers (explicit)

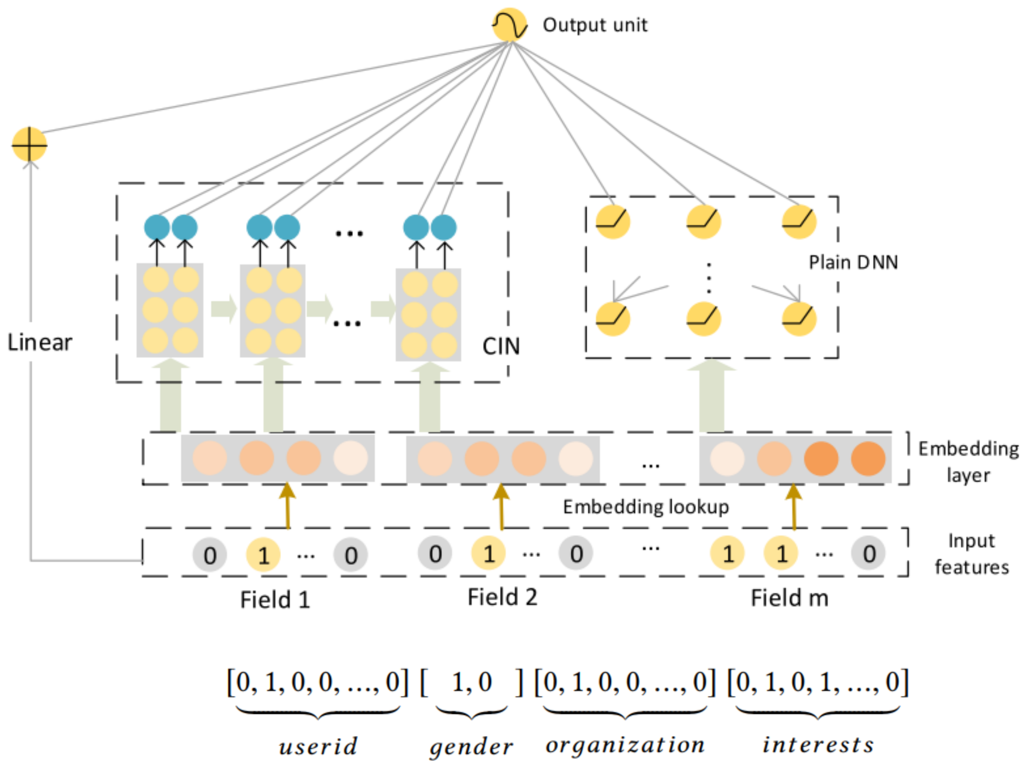
Jianxun Lian et al, Combining explicit and implicit feature interactions for recommender systems, KDD 2018

Extreme Deep Factorization Machine (xDeepFM)

- Proposed for CTR prediction

$$\hat{y} = \sigma(\mathbf{w}_{linear}^T \mathbf{a} + \mathbf{w}_{dnn}^T \mathbf{x}_{dnn}^k + \mathbf{w}_{cin}^T \mathbf{p}^+ + b)$$

  - Low-order and high-order feature interactions:
    - Linear: linear and quadratic interactions (low order)
    - DNN higher order implicit interactions (black-box, no theoretical understanding, noise effects)
    - Compressed Interaction Network (CIN)
      - Compresses embeddings
      - High-order explicit interactions
      - Vector-wise instead of bit-wise



Jianxun Lian et al, Combining explicit and implicit feature interactions for recommender systems, KDD 2018

Approval of Recommenders as a Sandbox project

Proposed Resolution:

- Recommenders as a Sandbox project of the LF AI & Data Foundation is hereby approved.

LF AI & DATA

Upcoming TAC Meetings

LF AI & DATA

Upcoming TAC Meetings

- July 13 - OPPO new sandbox project ShaderNN
- July 29 – Docarry proposal to move from Sandbox to Incubation, Tentative Project review

Please note we are always open to special topics as well.

If you have a topic idea or agenda item, please send agenda topic requests to tac-general@lists.lfaidata.foundation

Open Discussion

LF AI & DATA

TAC Meeting Details

- To subscribe to the TAC Group Calendar, visit the wiki: https://wiki.lfaidata.foundation/x/cQB2 ______
- Join from PC, Mac, Linux, iOS or Android: https://zoom.us/j/430697670
- Or iPhone one-tap:
  - US: +16465588656,,430697670# or +16699006833,,430697670#
- Or Telephone:
  - Dial(for higher quality, dial a number based on your current location):
  - US: +1 646 558 8656 or +1 669 900 6833 or +1 855 880 1246 (Toll Free) or +1 877 369 0926 (Toll Free)
- Meeting ID: 430 697 670
- International numbers available: https://zoom.us/u/achYtcw7uN

Legal Notice

- The Linux Foundation, The Linux Foundation logos, and other marks that may be used herein are owned by The Linux Foundation or its affiliated entities, and are subject to The Linux Foundation's Trademark Usage Policy at https://www.linuxfoundation.org/trademark-usage, as may be modified from time to time.
- Linux is a registered trademark of Linus Torvalds. Please see the Linux Mark Institute's trademark usage page at https://lmi.linuxfoundation.org for details regarding use of this trademark.
- Some marks that may be used herein are owned by projects operating as separately incorporated entities managed by The Linux Foundation, and have their own trademarks, policies and usage guidelines.
- TWITTER, TWEET, RETWEET and the Twitter logo are trademarks of Twitter, Inc. or its affiliates.
- Facebook and the "f" logo are trademarks of Facebook or its affiliates.
- LinkedIn, the LinkedIn logo, the IN logo and InMail are registered trademarks or trademarks of LinkedIn Corporation and its affiliates in the United States and/or other countries.
- YouTube and the YouTube icon are trademarks of YouTube or its affiliates.
- All other trademarks are the property of their respective owners. Use of such marks herein does not represent affiliation with or authorization, sponsorship or approval by such owners unless otherwise expressly specified.
- The Linux Foundation is subject to other policies, including without limitation its Privacy Policy at https://www.linuxfoundation.org/privacy and its Antitrust Policy at https://www.linuxfoundation.org/antitrust-policy. each as may be modified from time to time. More information about The Linux Foundation's policies is available at https://www.linuxfoundation.org.
- Please email legal@linuxfoundation.org with any questions about The Linux Foundation's policies or the notices set forth on this slide.

LF AI & DATA