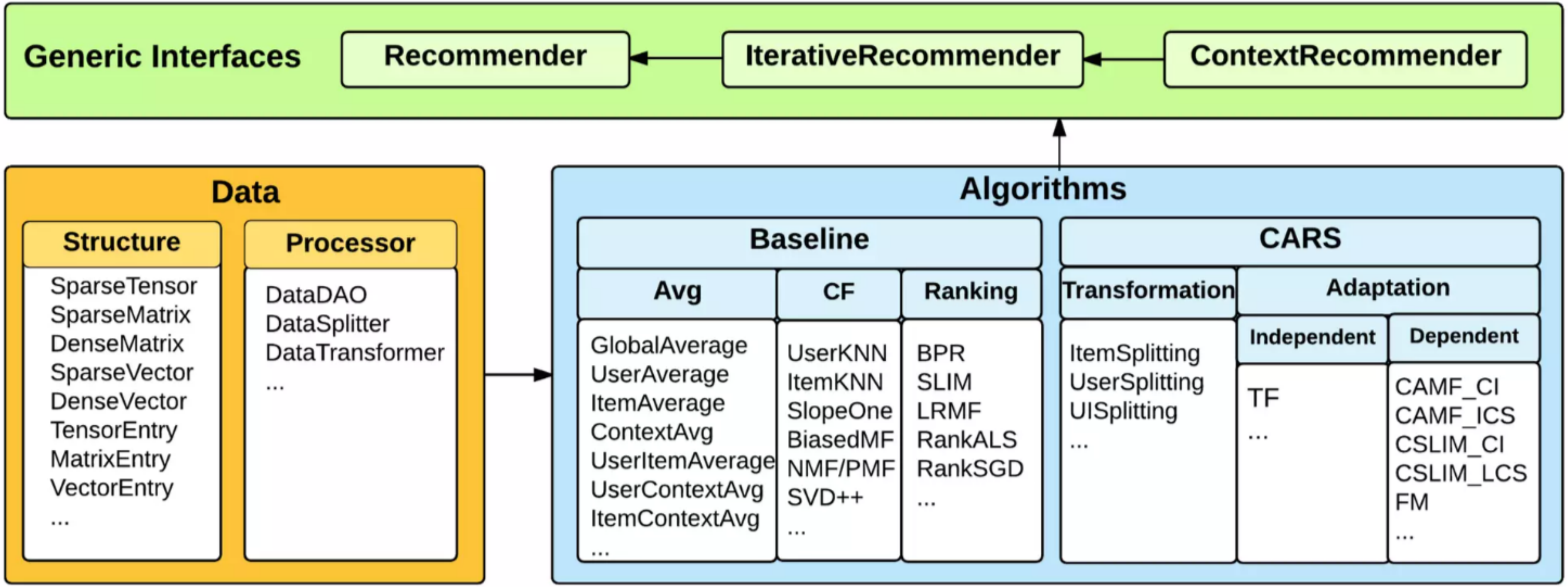
CARSKit: https://github.com/irecsys/CARSKit

Users × Items × Contexts ⟶ Ratings



Generic Interfaces

Recommender ← IterativeRecommender ← ContextRecommender

Data

Structure

SparseTensor
SparseMatrix
DenseMatrix
SparseVector
DenseVector
TensorEntry
MatrixEntry
VectorEntry
...

Processor

DataDAO
DataSplitter
DataTransformer
...

Algorithms

Baseline

Avg

GlobalAverage
UserAverage
ItemAverage
ContextAvg
UserItemAverage
UserContextAvg
ItemContextAvg
...

CF

UserKNN
ItemKNN
SlopeOne
BiasedMF
NMF/PMF
SVD++
...

Ranking

BPR
SLIM
LRMF
RankALS
RankSGD
...

CARS

Transformation

ItemSplitting
UserSplitting
UISplitting
...

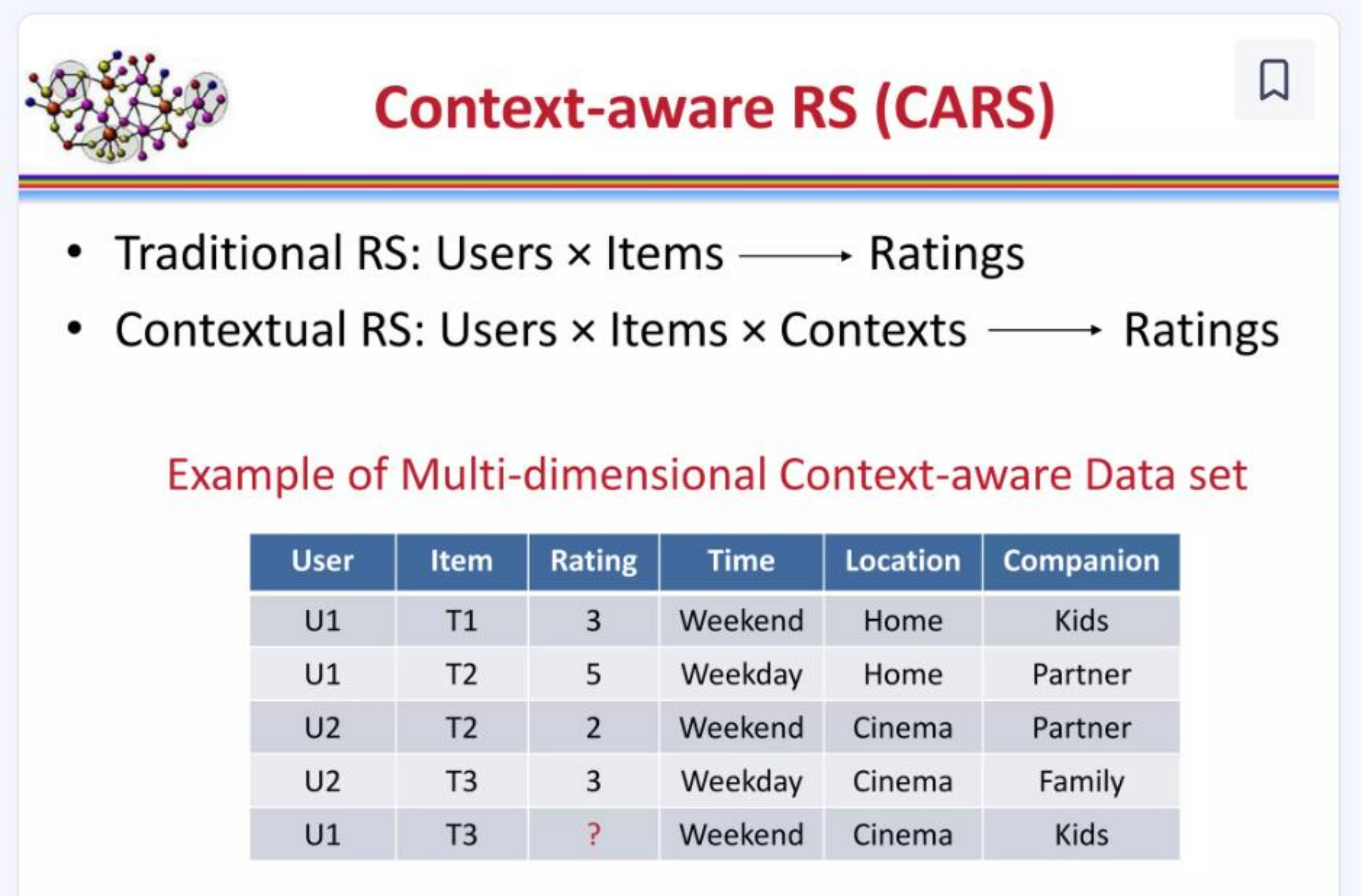
Adaptation

Independent

TF
...

Dependent

CAMF_CI
CAMF_ICS
CSLIM_CI
CSLIM_LCS
FM
...



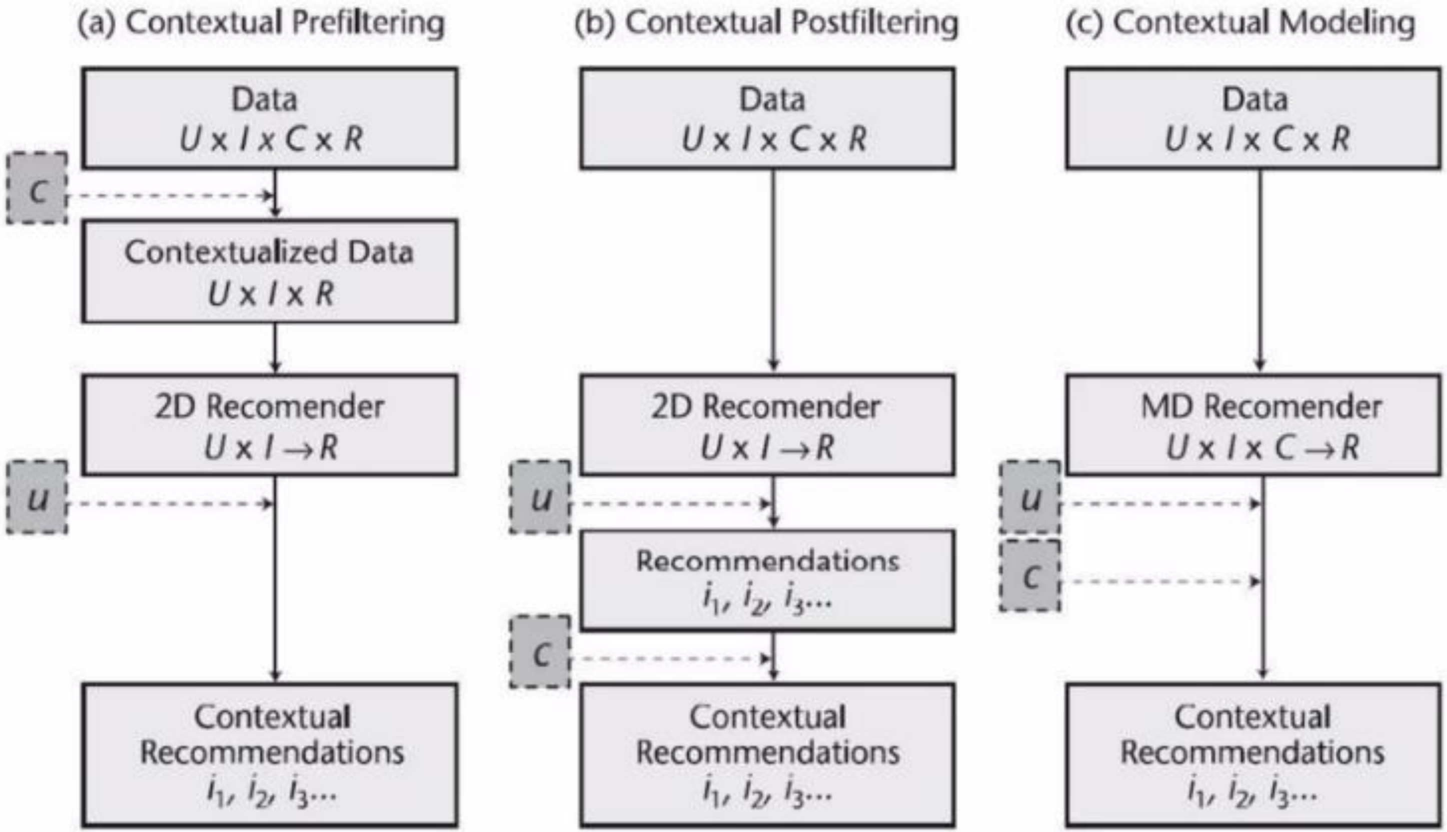
Context-aware RS (CARS)

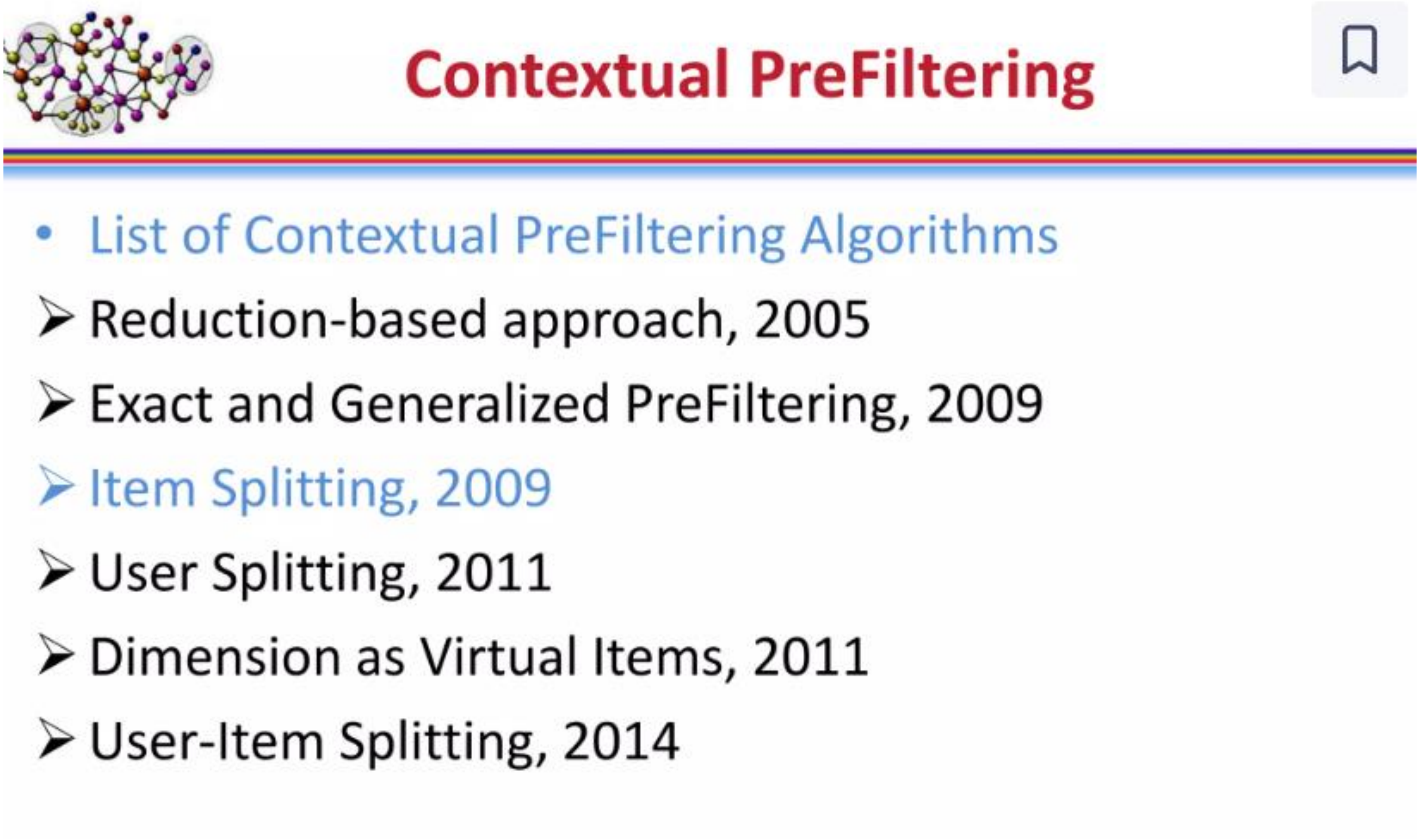
- Traditional RS: Users × Items ⟶ Ratings
- Contextual RS: Users × Items × Contexts ⟶ Ratings

Example of Multi-dimensional Context-aware Data set

User	Item	Rating	Time	Location	Companion
U1	T1	3	Weekend	Home	Kids
U1	T2	5	Weekday	Home	Partner
U2	T2	2	Weekend	Cinema	Partner
U2	T3	3	Weekday	Cinema	Family
U1	T3	?	Weekend	Cinema	Kids

- There are three ways to build algorithms for CARS





Contextual PreFiltering

- List of Contextual PreFiltering Algorithms

➢ Reduction-based approach, 2005

➢ Exact and Generalized PreFiltering, 2009

➢ Item Splitting, 2009

➢ User Splitting, 2011

➢ Dimension as Virtual Items, 2011

➢ User-Item Splitting, 2014

Contextual Modeling

- List of Contextual Modeling Algorithms

➢ Tensor Factorization, 2010

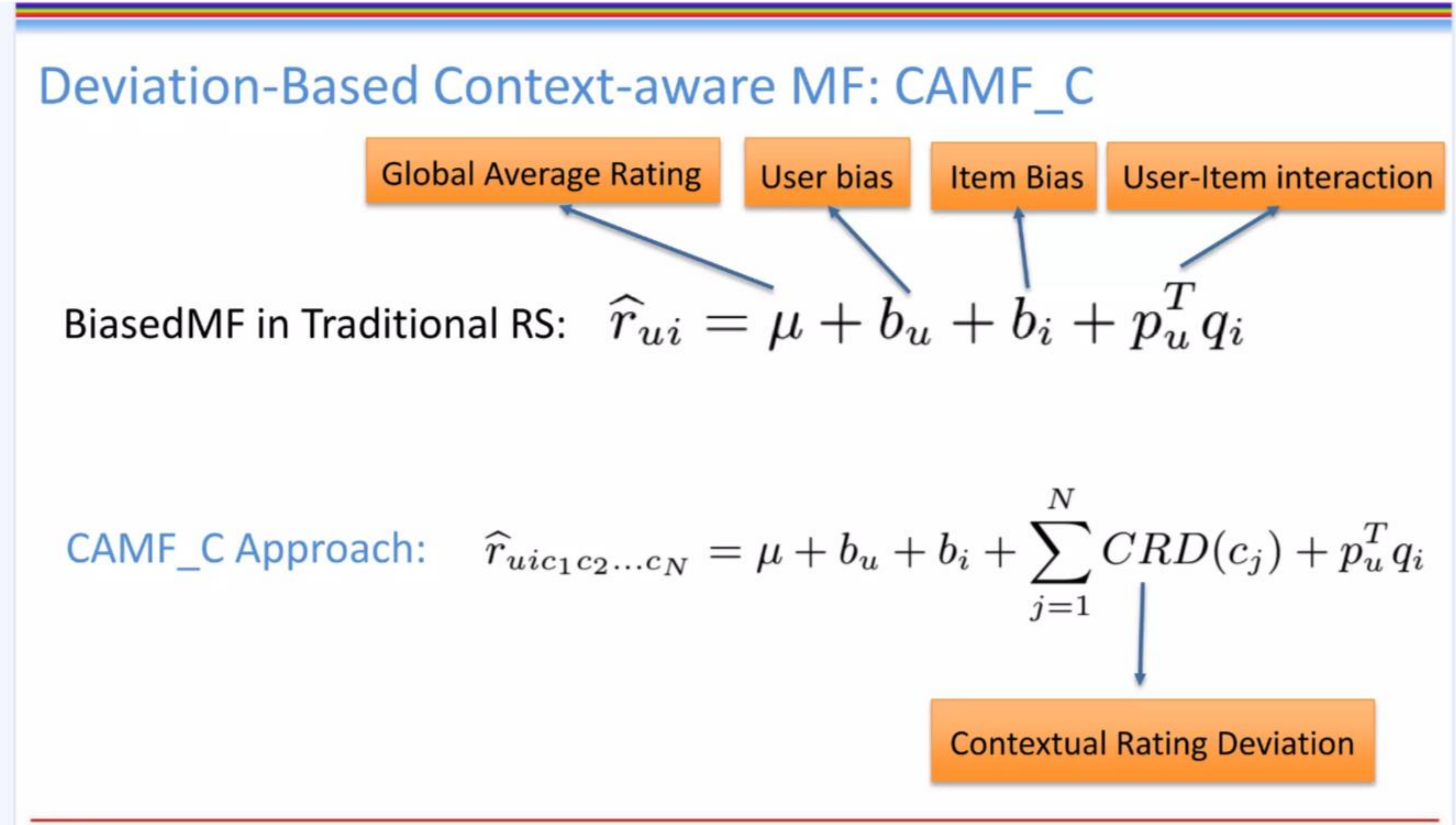
➢ Factorization Machines, 2011

➢ Deviation-Based Context-aware Matrix Factorization, 2011

➢ Deviation-Based Contextual Sparse Linear Method, 2014

➢ Similarity-Based Context-aware Matrix Factorization, 2015

➢ Similarity-Based Contextual Sparse Linear Method, 2015



Deviation-Based Context-aware MF: CAMF_C

Global Average Rating

User bias

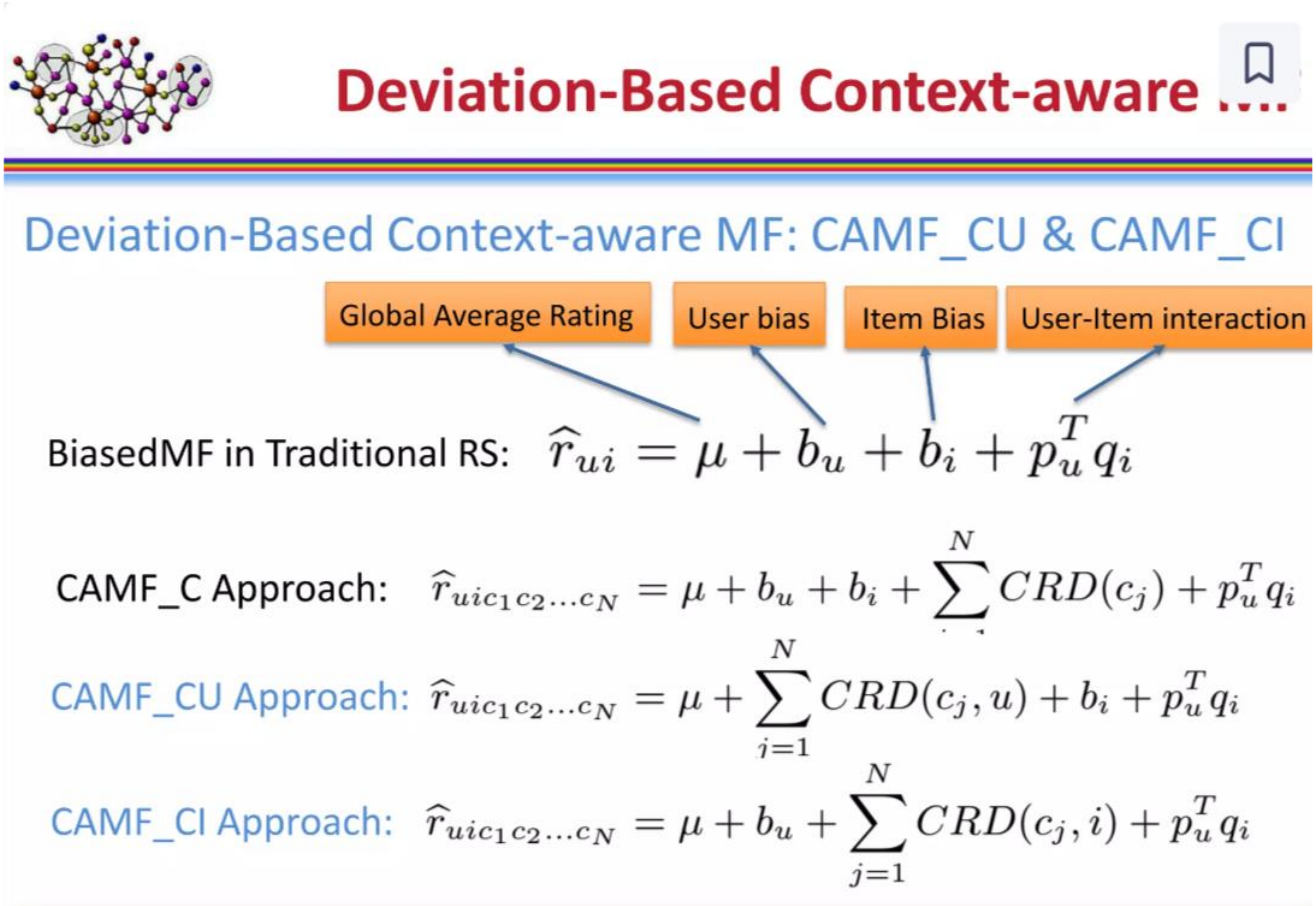
Item Bias

User-Item interaction

BiasedMF in Traditional RS: $\widehat{r}_{ui} = \mu + b_u + b_i + p_u^T q_i$

CAMF_C Approach: $\widehat{r}_{uic_1 c_2 \ldots c_N} = \mu + b_u + b_i + \sum_{j=1}^{N} CRD(c_j) + p_u^T q_i$

Contextual Rating Deviation



Deviation-Based Context-aware ...

Deviation-Based Context-aware MF: CAMF_CU & CAMF_CI

Global Average Rating

User bias

Item Bias

User-Item interaction

BiasedMF in Traditional RS: $\widehat{r}_{ui} = \mu + b_u + b_i + p_u^T q_i$

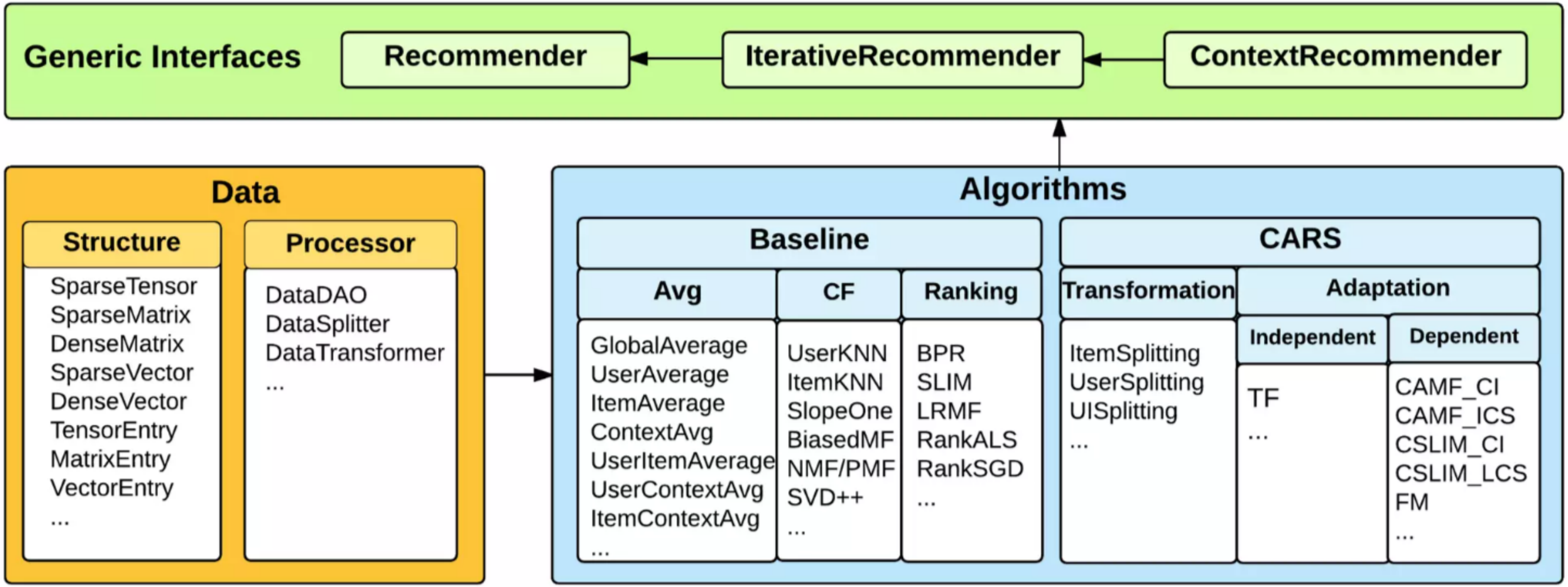
CAMF_C Approach: $\widehat{r}_{uic_1c_2\ldots c_N} = \mu + b_u + b_i + \sum_{j=1}^{N} CRD(c_j) + p_u^T q_i$

CAMF_CU Approach: $\widehat{r}_{uic_1c_2\ldots c_N} = \mu + \sum_{j=1}^{N} CRD(c_j, u) + b_i + p_u^T q_i$

CAMF_CI Approach: $\widehat{r}_{uic_1c_2\ldots c_N} = \mu + b_u + \sum_{j=1}^{N} CRD(c_j, i) + p_u^T q_i$

CARSKit: https://github.com/irecsys/CARSKit

Users × Items × Contexts ⟶ Ratings



Generic Interfaces

Recommender ← IterativeRecommender ← ContextRecommender

Data

Structure
SparseTensor
SparseMatrix
DenseMatrix
SparseVector
DenseVector
TensorEntry
MatrixEntry
VectorEntry
...

Processor
DataDAO
DataSplitter
DataTransformer
...

Algorithms

Baseline

Avg
GlobalAverage
UserAverage
ItemAverage
ContextAvg
UserItemAverage
UserContextAvg
ItemContextAvg
...

CF
UserKNN
ItemKNN
SlopeOne
BiasedMF
NMF/PMF
SVD++
...

Ranking
BPR
SLIM
LRMF
RankALS
RankSGD
...

CARS

Transformation
ItemSplitting
UserSplitting
UISplitting
...

Adaptation

Independent
TF
...

Dependent
CAMF_CI
CAMF_ICS
CSLIM_CI
CSLIM_LCS
FM
...

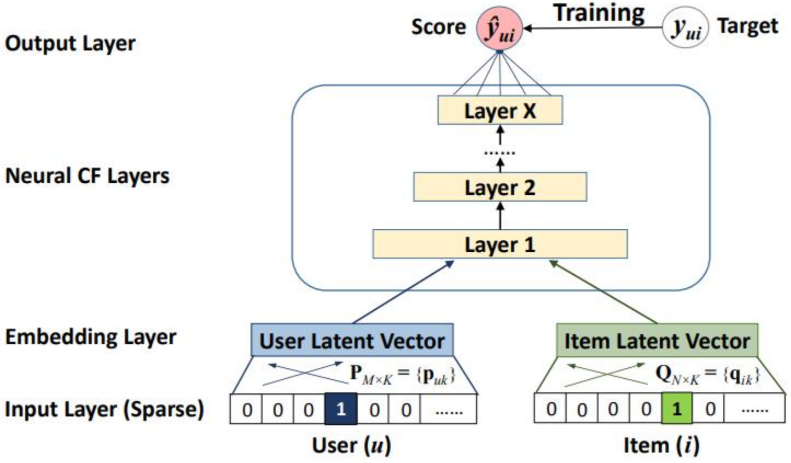


Figure 2: Neural collaborative filtering framework

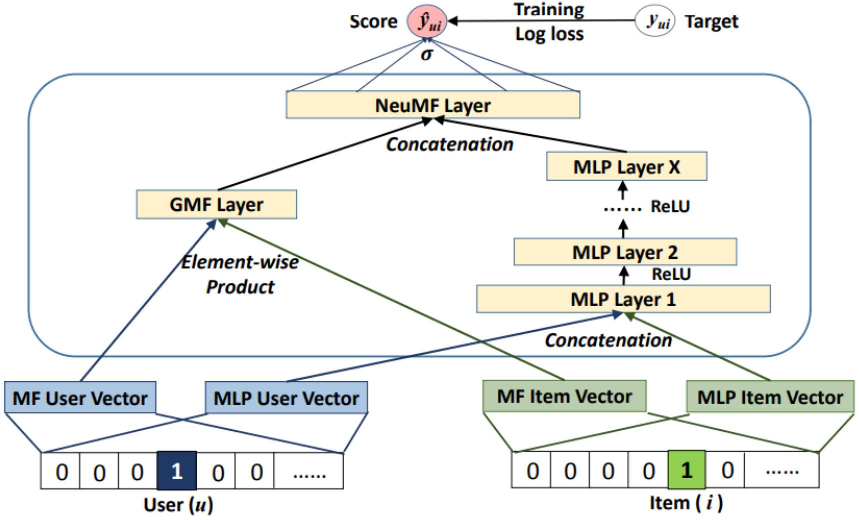


Figure 3: Neural matrix factorization model

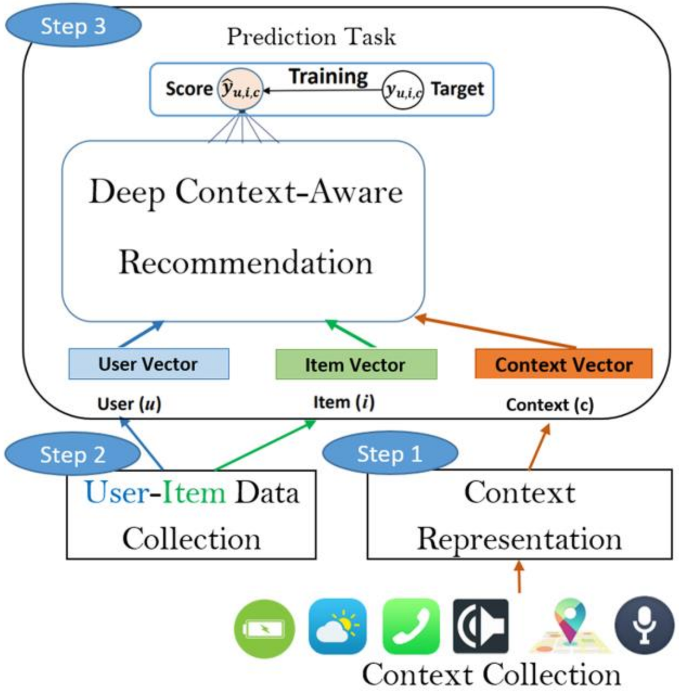


Fig. 1. Deep context-aware recommendation framework.

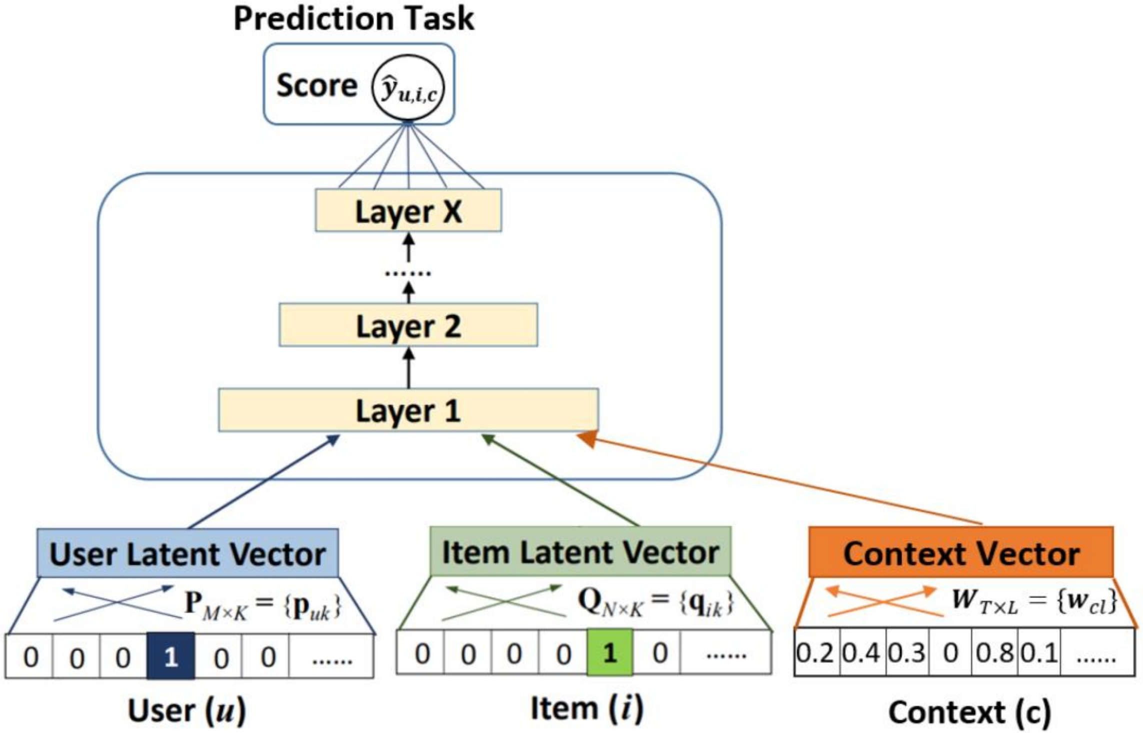


Fig. 3. Neural-CF (NCF) extension with contextual information.

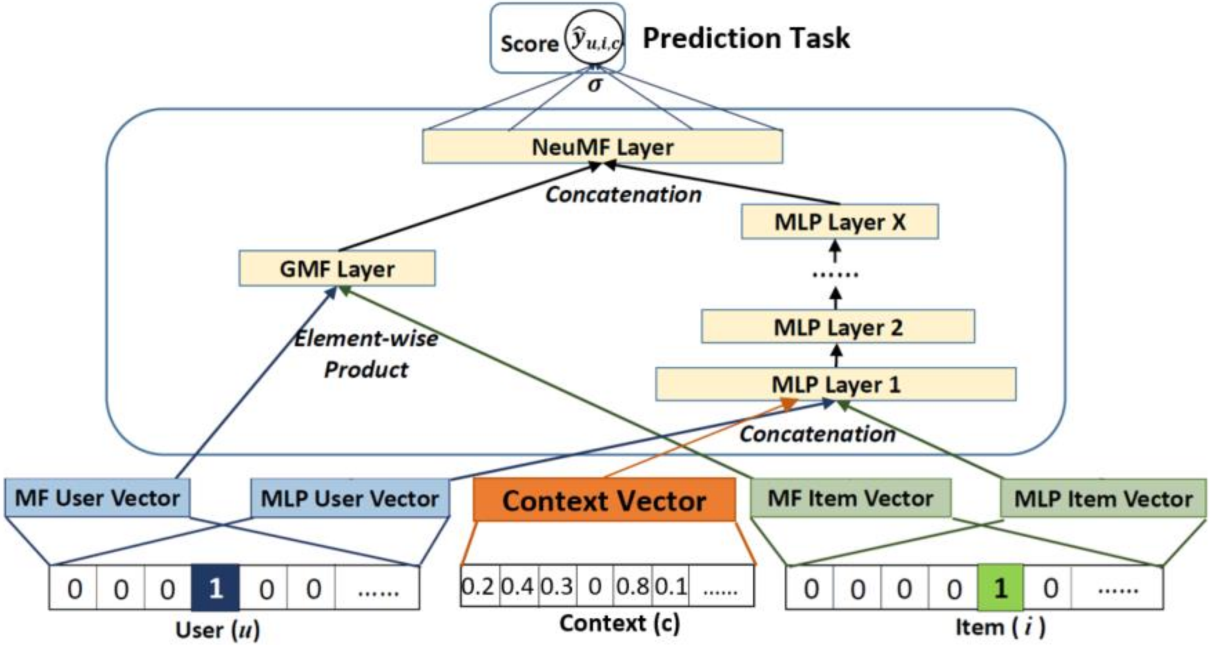
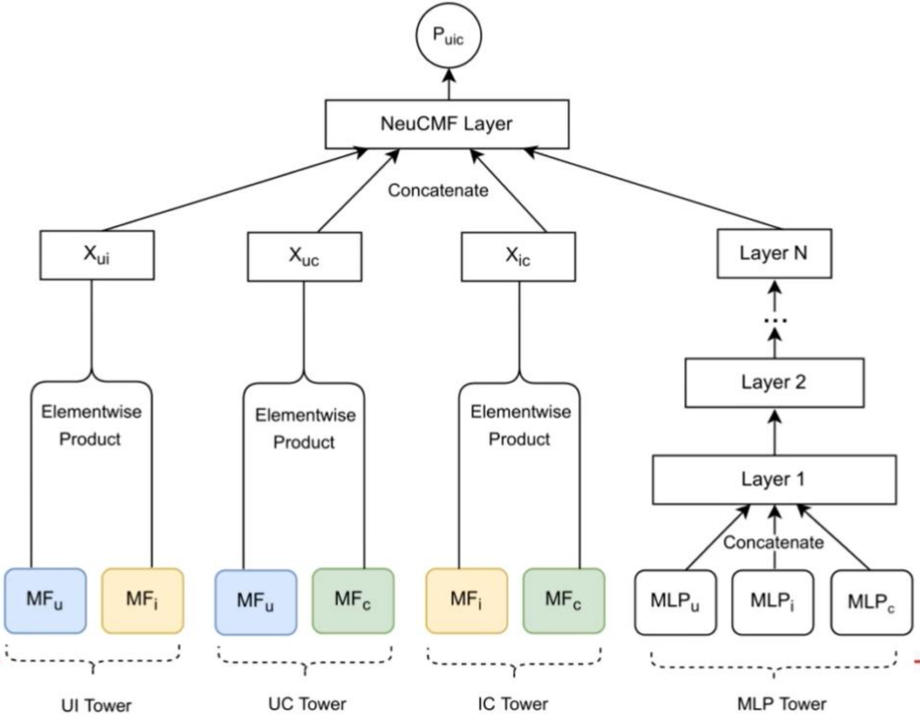
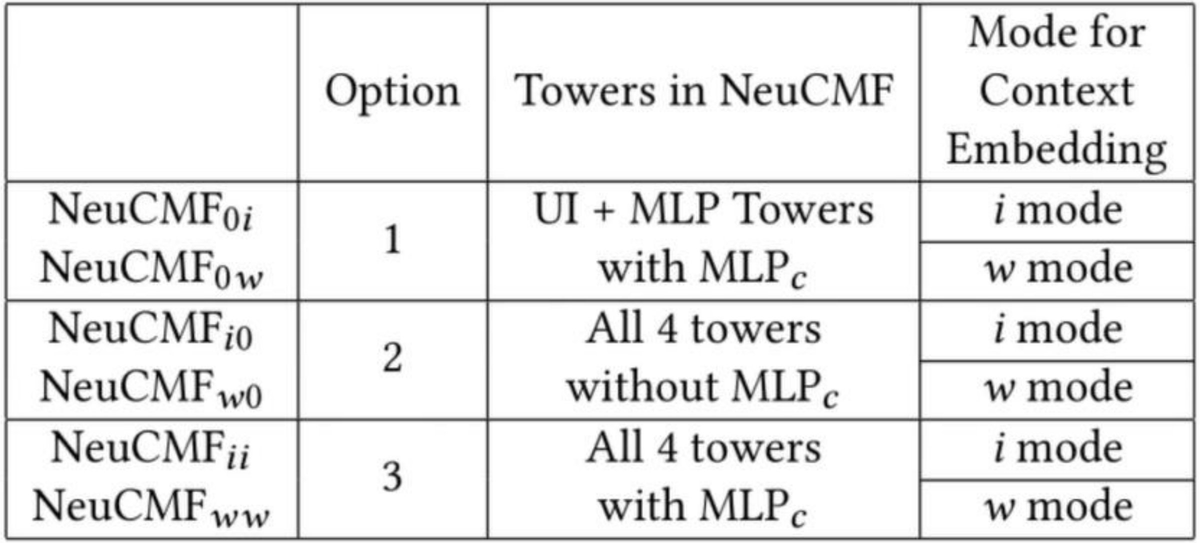


Fig. 4. Neural-MF (NeuMF) extension with contextual information.

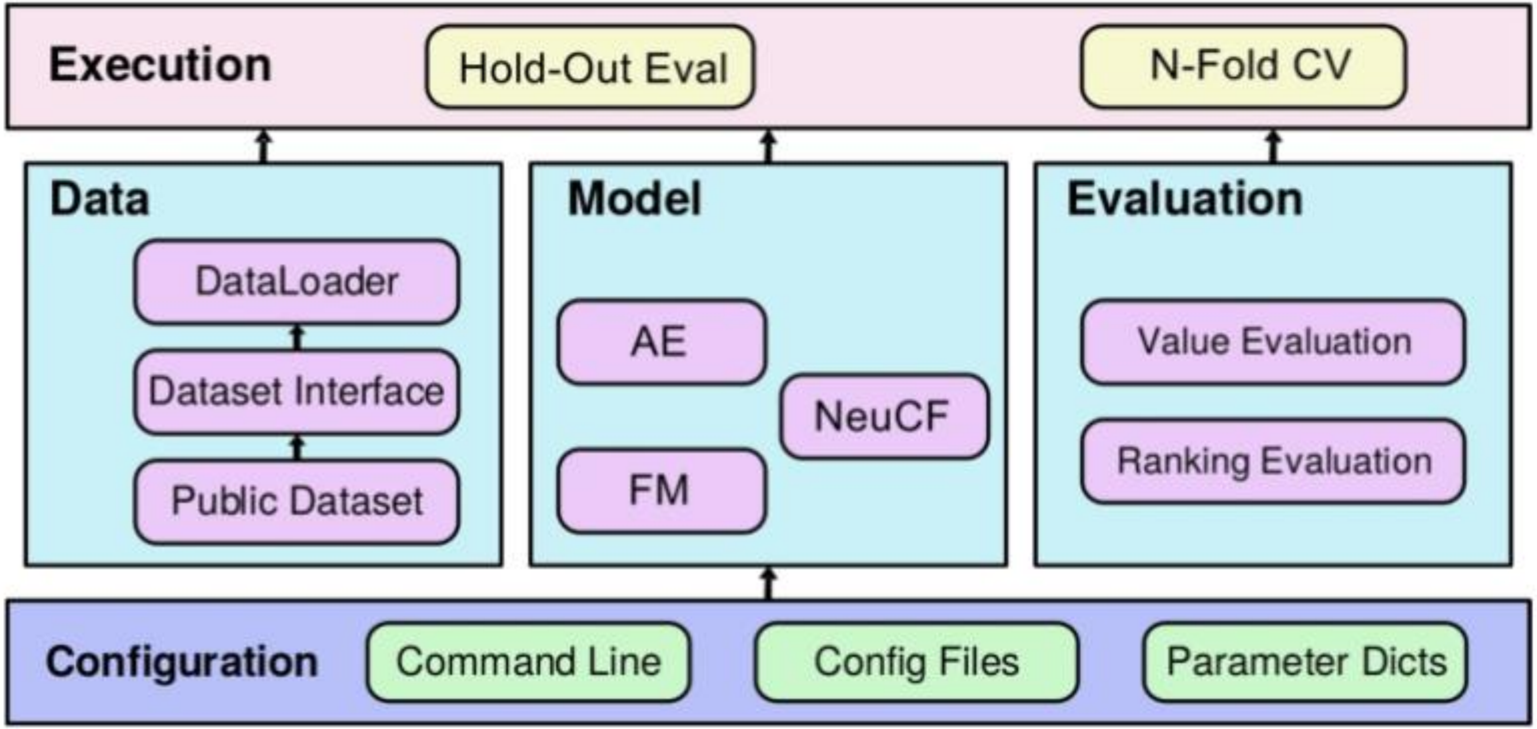
- Incorporating contexts into MLP & MF tower

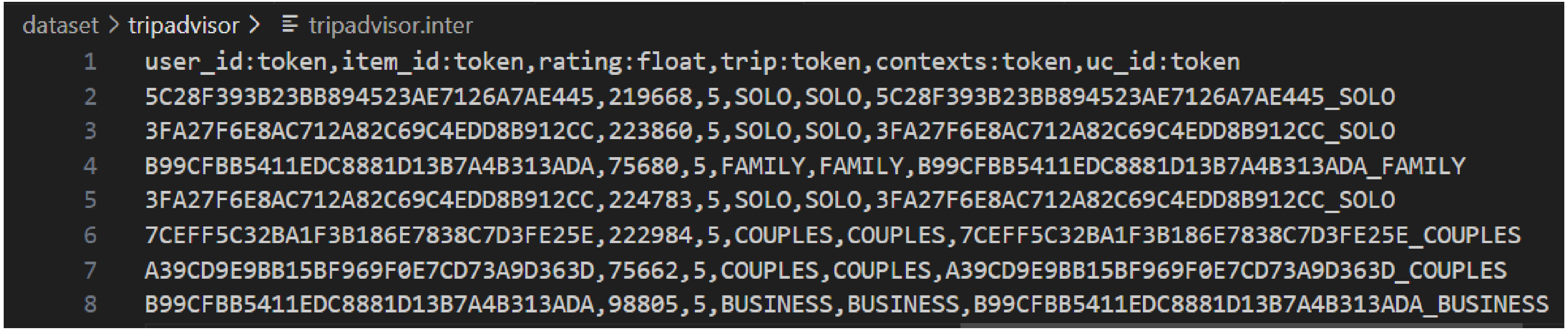


- Incorporating contexts into MLP & MF tower



	Option	Towers in NeuCMF	Mode for Context Embedding
NeuCMF$_{0i}$	1	UI + MLP Towers with MLP$_c$	i mode
NeuCMF$_{0w}$			w mode
NeuCMF$_{i0}$	2	All 4 towers without MLP$_c$	i mode
NeuCMF$_{w0}$			w mode
NeuCMF$_{ii}$	3	All 4 towers with MLP$_c$	i mode
NeuCMF$_{ww}$			w mode

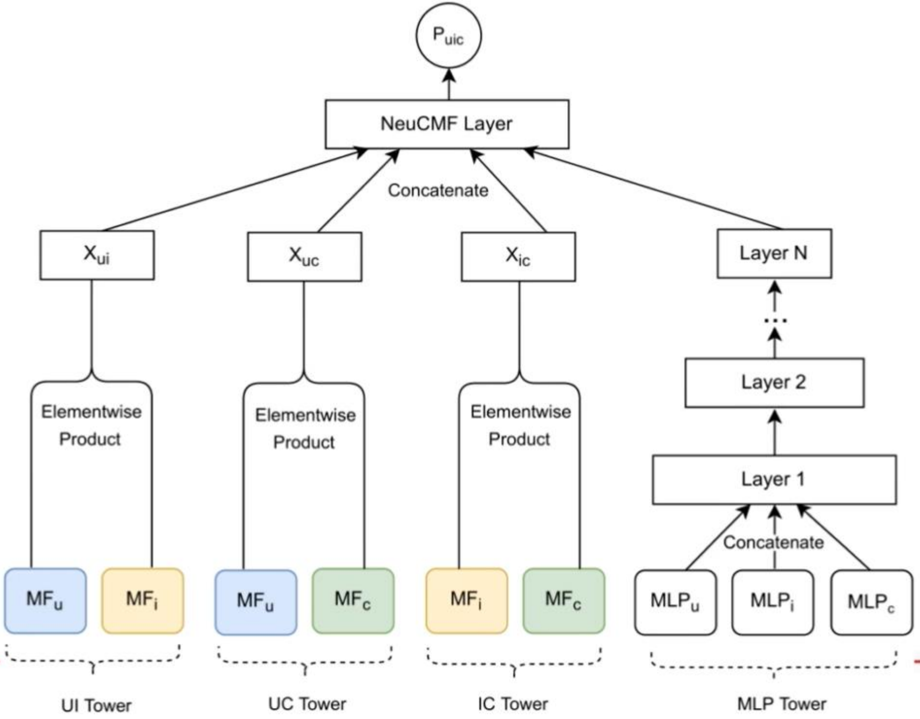

Execution
Hold-Out Eval
N-Fold CV
Data
DataLoader
Dataset Interface
Public Dataset
Model
AE
NeuCF
FM
Evaluation
Value Evaluation
Ranking Evaluation
Configuration
Command Line
Config Files
Parameter Dicts



```
user_id:token,item_id:token,rating:float,trip:token,contexts:token,uc_id:token
5C28F393B23BB894523AE7126A7AE445,219668,5,SOLO,SOLO,5C28F393B23BB894523AE7126A7AE445_SOLO
3FA27F6E8AC712A82C69C4EDD8B912CC,223860,5,SOLO,SOLO,3FA27F6E8AC712A82C69C4EDD8B912CC_SOLO
B99CFBB5411EDC8881D13B7A4B313ADA,75680,5,FAMILY,FAMILY,B99CFBB5411EDC8881D13B7A4B313ADA_FAMILY
3FA27F6E8AC712A82C69C4EDD8B912CC,224783,5,SOLO,SOLO,3FA27F6E8AC712A82C69C4EDD8B912CC_SOLO
7CEFF5C32BA1F3B186E7838C7D3FE25E,222984,5,COUPLES,COUPLES,7CEFF5C32BA1F3B186E7838C7D3FE25E_COUPLES
A39CD9E9BB15BF969F0E7CD73A9D363D,75662,5,COUPLES,COUPLES,A39CD9E9BB15BF969F0E7CD73A9D363D_COUPLES
B99CFBB5411EDC8881D13B7A4B313ADA,98805,5,BUSINESS,BUSINESS,B99CFBB5411EDC8881D13B7A4B313ADA_BUSINESS
```

```
26 Aug 14:57    INFO  tripadvisor
The number of users: 2372
Average actions of users: 5.978490088570224
The number of items: 2270
Average actions of items: 6.24724548259145
The number of inters: 14175
The sparsity of the dataset: 99.73674142529214%
Remain Fields: ['user_id', 'item_id', 'rating', 'trip', 'contexts', 'uc_id']
Context dimension - trip: 6 values: : ['BUSINESS' 'COUPLES' 'FAMILY' 'FRIENDS' 'SOLO'
'[PAD]']
```

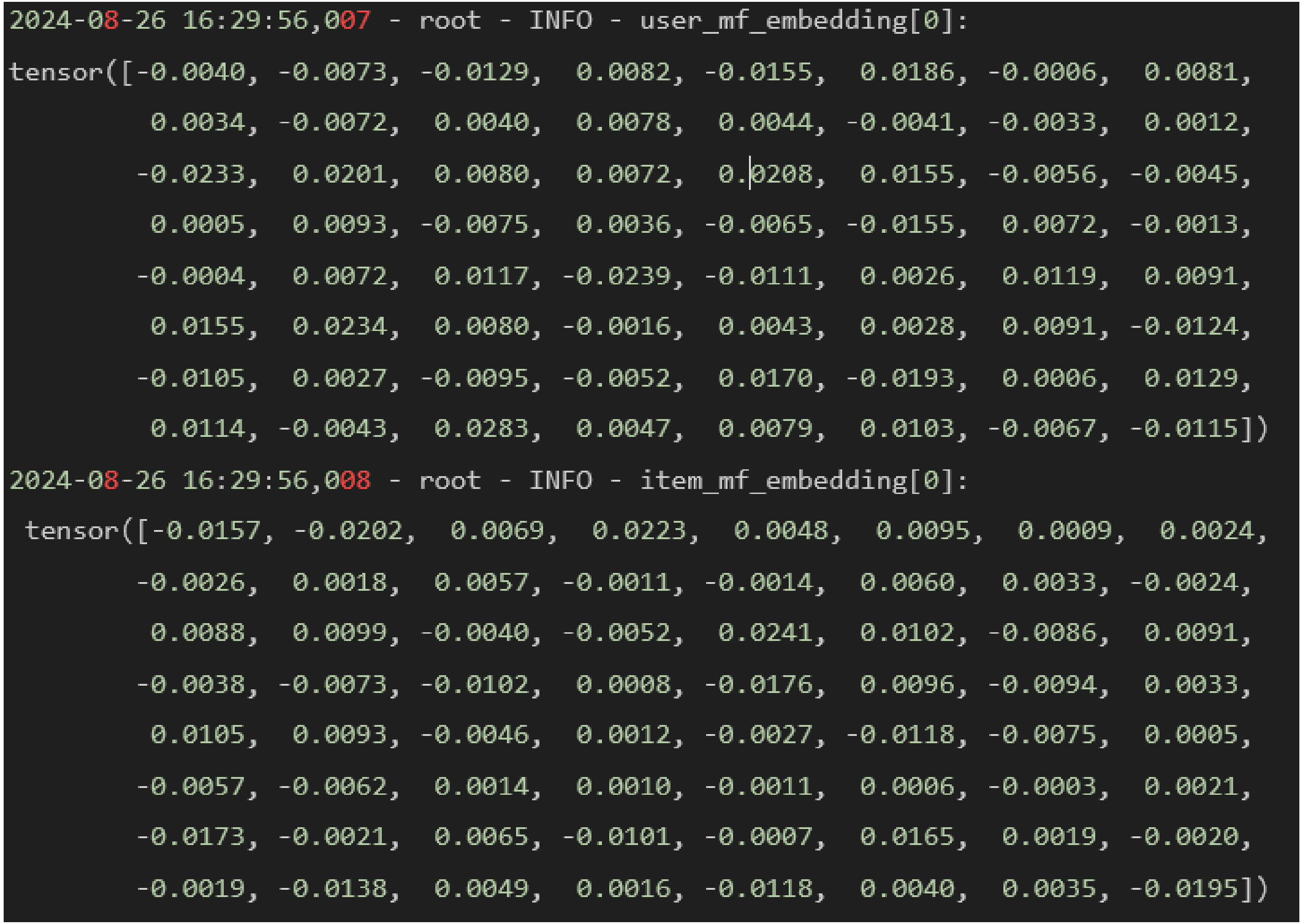
- Incorporating contexts into MLP & MF tower



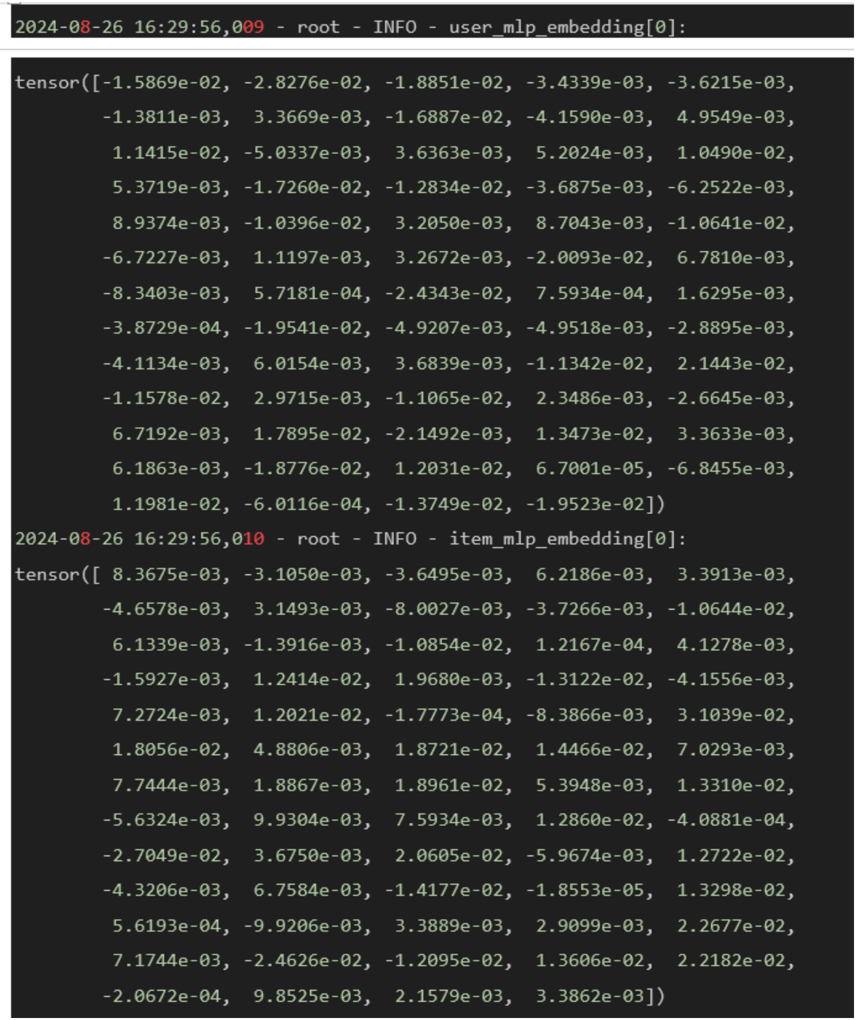
```
2024-08-26 16:08:57,052 - root - INFO - mf_embedding_size: 64
2024-08-26 16:08:57,052 - root - INFO - mlp_embedding_size: 64
2024-08-26 16:08:57,052 - root - INFO - mlp_hidden_size: [128, 64, 32]
```

```
2024-08-26 16:08:57,052 - root - INFO - user_mf_embedding: Embedding(2372, 64)
2024-08-26 16:08:57,052 - root - INFO - item_mf_embedding: Embedding(2270, 64)
2024-08-26 16:08:57,052 - root - INFO - context_situation_mf_embedding: [Embedding(6,
64)]

2024-08-26 16:08:57,052 - root - INFO - user_mlp_embedding: Embedding(2372, 64)
2024-08-26 16:08:57,052 - root - INFO - item_mlp_embedding: Embedding(2270, 64)
2024-08-26 16:08:57,052 - root - INFO - context_dimensions_mlp_embedding:
[Embedding(6, 64)]
```

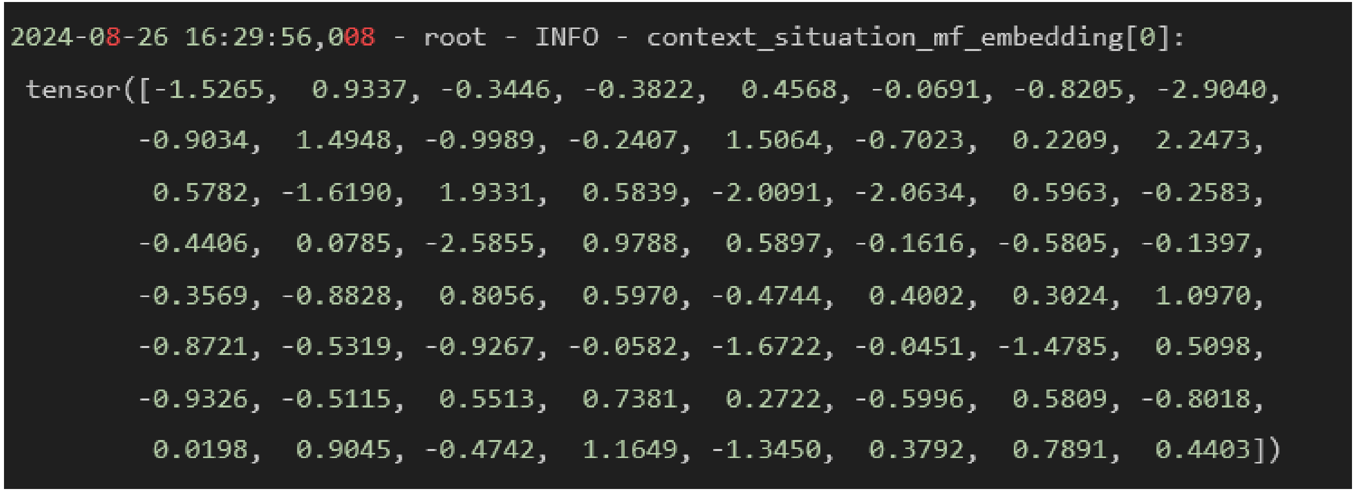


```
2024-08-26 16:29:56,007 - root - INFO - user_mf_embedding[0]:
tensor([-0.0040, -0.0073, -0.0129,  0.0082, -0.0155,  0.0186, -0.0006,  0.0081,
         0.0034, -0.0072,  0.0040,  0.0078,  0.0044, -0.0041, -0.0033,  0.0012,
        -0.0233,  0.0201,  0.0080,  0.0072,  0.0208,  0.0155, -0.0056, -0.0045,
         0.0005,  0.0093, -0.0075,  0.0036, -0.0065, -0.0155,  0.0072, -0.0013,
        -0.0004,  0.0072,  0.0117, -0.0239, -0.0111,  0.0026,  0.0119,  0.0091,
         0.0155,  0.0234,  0.0080, -0.0016,  0.0043,  0.0028,  0.0091, -0.0124,
        -0.0105,  0.0027, -0.0095, -0.0052,  0.0170, -0.0193,  0.0006,  0.0129,
         0.0114, -0.0043,  0.0283,  0.0047,  0.0079,  0.0103, -0.0067, -0.0115])
2024-08-26 16:29:56,008 - root - INFO - item_mf_embedding[0]:
 tensor([-0.0157, -0.0202,  0.0069,  0.0223,  0.0048,  0.0095,  0.0009,  0.0024,
        -0.0026,  0.0018,  0.0057, -0.0011, -0.0014,  0.0060,  0.0033, -0.0024,
         0.0088,  0.0099, -0.0040, -0.0052,  0.0241,  0.0102, -0.0086,  0.0091,
        -0.0038, -0.0073, -0.0102,  0.0008, -0.0176,  0.0096, -0.0094,  0.0033,
         0.0105,  0.0093, -0.0046,  0.0012, -0.0027, -0.0118, -0.0075,  0.0005,
        -0.0057, -0.0062,  0.0014,  0.0010, -0.0011,  0.0006, -0.0003,  0.0021,
        -0.0173, -0.0021,  0.0065, -0.0101, -0.0007,  0.0165,  0.0019, -0.0020,
        -0.0019, -0.0138,  0.0049,  0.0016, -0.0118,  0.0040,  0.0035, -0.0195])
```

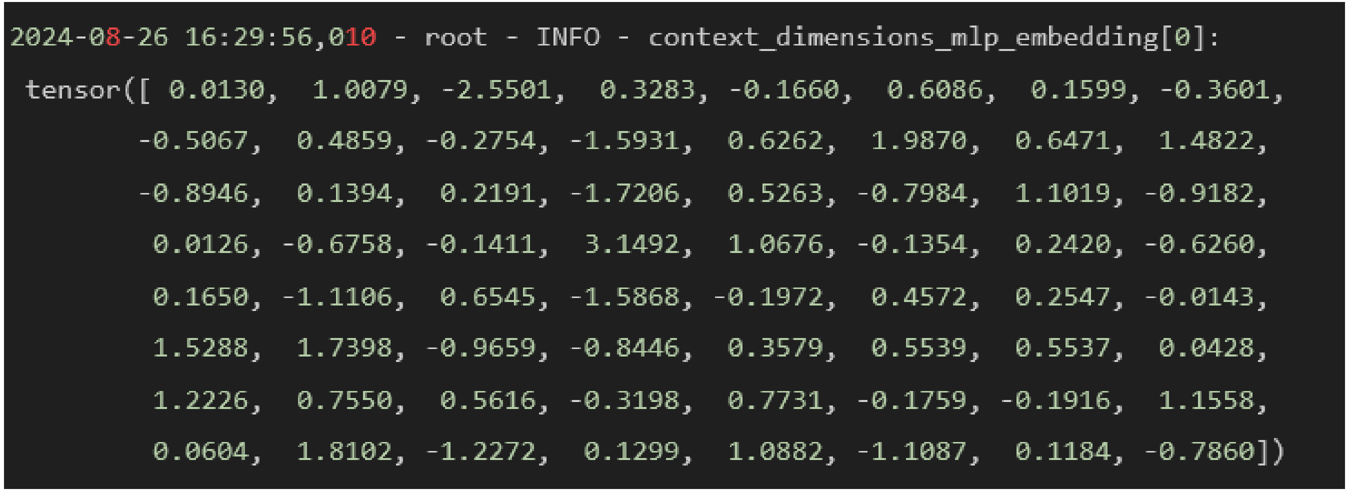


```
2024-08-26 16:29:56,009 - root - INFO - user_mlp_embedding[0]:
```

```
tensor([-1.5869e-02, -2.8276e-02, -1.8851e-02, -3.4339e-03, -3.6215e-03,
        -1.3811e-03,  3.3669e-03, -1.6887e-02, -4.1590e-03,  4.9549e-03,
         1.1415e-02, -5.0337e-03,  3.6363e-03,  5.2024e-03,  1.0490e-02,
         5.3719e-03, -1.7260e-02, -1.2834e-02, -3.6875e-03, -6.2522e-03,
         8.9374e-03, -1.0396e-02,  3.2050e-03,  8.7043e-03, -1.0641e-02,
        -6.7227e-03,  1.1197e-03,  3.2672e-03, -2.0093e-02,  6.7810e-03,
        -8.3403e-03,  5.7181e-04, -2.4343e-02,  7.5934e-04,  1.6295e-03,
        -3.8729e-04, -1.9541e-02, -4.9207e-03, -4.9518e-03, -2.8895e-03,
        -4.1134e-03,  6.0154e-03,  3.6839e-03, -1.1342e-02,  2.1443e-02,
        -1.1578e-02,  2.9715e-03, -1.1065e-02,  2.3486e-03, -2.6645e-03,
         6.7192e-03,  1.7895e-02, -2.1492e-03,  1.3473e-02,  3.3633e-03,
         6.1863e-03, -1.8776e-02,  1.2031e-02,  6.7001e-05, -6.8455e-03,
         1.1981e-02, -6.0116e-04, -1.3749e-02, -1.9523e-02])
2024-08-26 16:29:56,010 - root - INFO - item_mlp_embedding[0]:
tensor([ 8.3675e-03, -3.1050e-03, -3.6495e-03,  6.2186e-03,  3.3913e-03,
        -4.6578e-03,  3.1493e-03, -8.0027e-03, -3.7266e-03, -1.0644e-02,
         6.1339e-03, -1.3916e-03, -1.0854e-02,  1.2167e-04,  4.1278e-03,
        -1.5927e-03,  1.2414e-02,  1.9680e-03, -1.3122e-02, -4.1556e-03,
         7.2724e-03,  1.2021e-02, -1.7773e-04, -8.3866e-03,  3.1039e-02,
         1.8056e-02,  4.8806e-03,  1.8721e-02,  1.4466e-02,  7.0293e-03,
         7.7444e-03,  1.8867e-03,  1.8961e-02,  5.3948e-03,  1.3310e-02,
        -5.6324e-03,  9.9304e-03,  7.5934e-03,  1.2860e-02, -4.0881e-04,
        -2.7049e-02,  3.6750e-03,  2.0605e-02, -5.9674e-03,  1.2722e-02,
        -4.3206e-03,  6.7584e-03, -1.4177e-02, -1.8553e-05,  1.3298e-02,
         5.6193e-04, -9.9206e-03,  3.3889e-03,  2.9099e-03,  2.2677e-02,
         7.1744e-03, -2.4626e-02, -1.2095e-02,  1.3606e-02,  2.2182e-02,
        -2.0672e-04,  9.8525e-03,  2.1579e-03,  3.3862e-03])
```



```
2024-08-26 16:29:56,008 - root - INFO - context_situation_mf_embedding[0]:
 tensor([-1.5265,  0.9337, -0.3446, -0.3822,  0.4568, -0.0691, -0.8205, -2.9040,
        -0.9034,  1.4948, -0.9989, -0.2407,  1.5064, -0.7023,  0.2209,  2.2473,
         0.5782, -1.6190,  1.9331,  0.5839, -2.0091, -2.0634,  0.5963, -0.2583,
        -0.4406,  0.0785, -2.5855,  0.9788,  0.5897, -0.1616, -0.5805, -0.1397,
        -0.3569, -0.8828,  0.8056,  0.5970, -0.4744,  0.4002,  0.3024,  1.0970,
        -0.8721, -0.5319, -0.9267, -0.0582, -1.6722, -0.0451, -1.4785,  0.5098,
        -0.9326, -0.5115,  0.5513,  0.7381,  0.2722, -0.5996,  0.5809, -0.8018,
         0.0198,  0.9045, -0.4742,  1.1649, -1.3450,  0.3792,  0.7891,  0.4403])
```



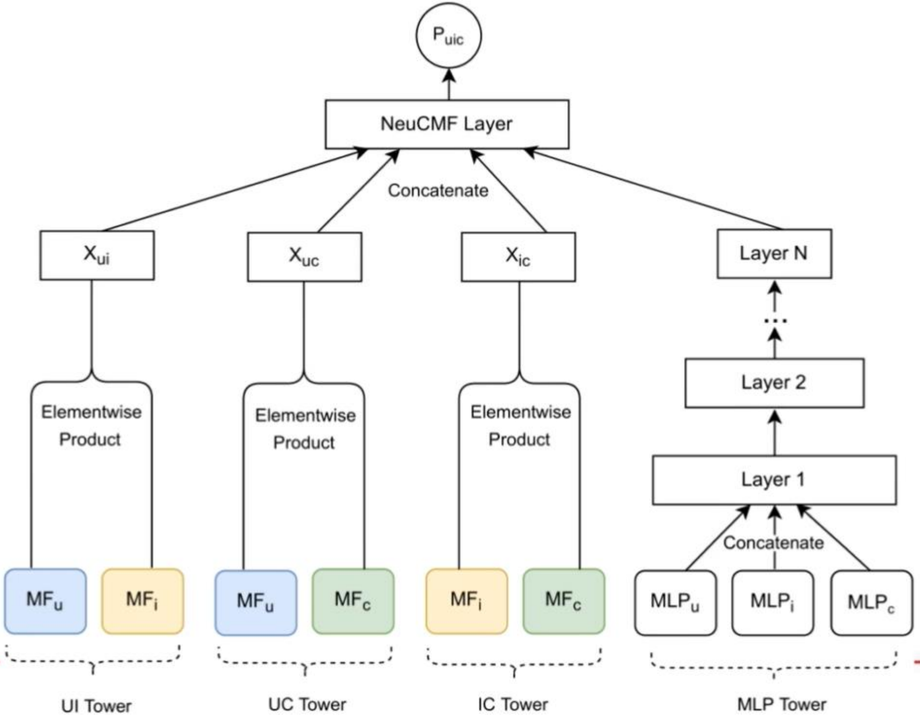
```
2024-08-26 16:29:56,010 - root - INFO - context_dimensions_mlp_embedding[0]:
 tensor([ 0.0130,  1.0079, -2.5501,  0.3283, -0.1660,  0.6086,  0.1599, -0.3601,
        -0.5067,  0.4859, -0.2754, -1.5931,  0.6262,  1.9870,  0.6471,  1.4822,
        -0.8946,  0.1394,  0.2191, -1.7206,  0.5263, -0.7984,  1.1019, -0.9182,
         0.0126, -0.6758, -0.1411,  3.1492,  1.0676, -0.1354,  0.2420, -0.6260,
         0.1650, -1.1106,  0.6545, -1.5868, -0.1972,  0.4572,  0.2547, -0.0143,
         1.5288,  1.7398, -0.9659, -0.8446,  0.3579,  0.5539,  0.5537,  0.0428,
         1.2226,  0.7550,  0.5616, -0.3198,  0.7731, -0.1759, -0.1916,  1.1558,
         0.0604,  1.8102, -1.2272,  0.1299,  1.0882, -1.1087,  0.1184, -0.7860])
```

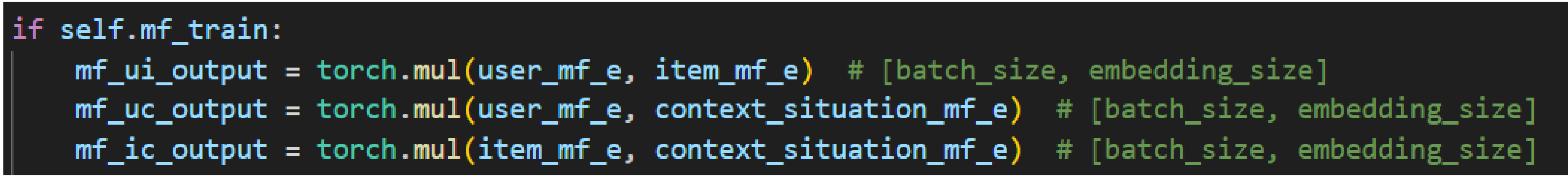
```
26 Aug 14:57    INFO  [Training]: train_batch_size = [500] negative sampling: [None]
26 Aug 14:57    INFO  [Evaluation]: eval_batch_size = [409600] eval_args: [{'split':
{'CV': 5}, 'order': 'RO', 'group_by': 'user', 'mode': {'valid': 'labeled', 'test': 'labeled'}}]
```

```
Thu 29 Aug 2024 16:57:02 INFO  user_mf_e size: torch.Size([500, 64])
Thu 29 Aug 2024 16:57:02 INFO  item_mf_e size: torch.Size([500, 64])
Thu 29 Aug 2024 16:57:02 INFO  context_situation_e size: torch.Size([500, 64])

Thu 29 Aug 2024 16:57:02 INFO  user_mlp_e size: torch.Size([500, 64])
Thu 29 Aug 2024 16:57:02 INFO  item_mlp_e size: torch.Size([500, 64])
Thu 29 Aug 2024 16:57:02 INFO  context_situation_e size: torch.Size([500, 64])
```

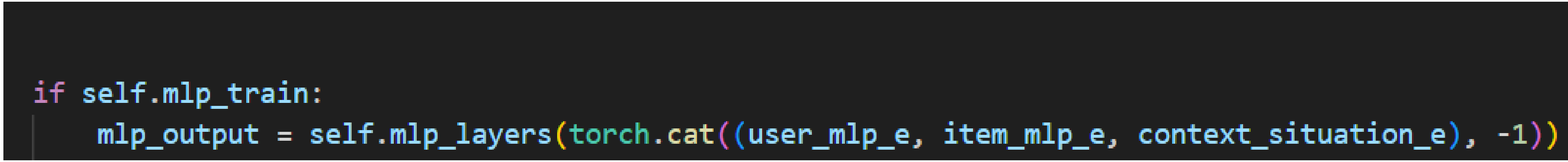
- Incorporating contexts into MLP & MF tower





```
if self.mf_train:
    mf_ui_output = torch.mul(user_mf_e, item_mf_e)  # [batch_size, embedding_size]
    mf_uc_output = torch.mul(user_mf_e, context_situation_mf_e)  # [batch_size, embedding_size]
    mf_ic_output = torch.mul(item_mf_e, context_situation_mf_e)  # [batch_size, embedding_size]
```

```
2024-08-26 17:02:56,894 - root - INFO - mf_uc_output:  - shape: torch.Size([500, 64])
2024-08-26 17:02:56,894 - root - INFO - mf_ic_output:  - shape: torch.Size([500, 64])
2024-08-26 17:02:56,895 - root - INFO - mf_ui_output:  - shape: torch.Size([500, 64])
```



```
if self.mlp_train:
    mlp_output = self.mlp_layers(torch.cat((user_mlp_e, item_mlp_e, context_situation_e), -1))
```

```
Thu 29 Aug 2024 16:38:11 INFO  mlp_output size: torch.Size([500, 32])
```

```
2024-08-26 16:42:37,954 - root - INFO - MLP input sizes: [192, 128, 64, 32]
2024-08-26 16:42:37,955 - root - INFO - MLP dropout probability: 0.1
2024-08-26 16:42:37,955 - root - INFO - MLP layers structure: MLPLayers(
  (mlp_layers): Sequential(
    (0): Dropout(p=0.1, inplace=False)
    (1): Linear(in_features=192, out_features=128, bias=True)
    (2): ReLU()
    (3): Dropout(p=0.1, inplace=False)
    (4): Linear(in_features=128, out_features=64, bias=True)
    (5): ReLU()
    (6): Dropout(p=0.1, inplace=False)
    (7): Linear(in_features=64, out_features=32, bias=True)
    (8): ReLU()
  )
)
```

```
self.mlp_layers = MLPLayers(
[(2 + self.n_contexts_dim) *
self.mlp_embedding_size] +
self.mlp_hidden_size,
self.dropout_prob)
```

```
Thu 29 Aug 2024 16:38:11 INFO  mf_ui_output size: torch.Size([500, 64])
Thu 29 Aug 2024 16:38:11 INFO  mf_ic_output size: torch.Size([500, 64])
Thu 29 Aug 2024 16:38:11 INFO  mf_uc_output size: torch.Size([500, 64])
Thu 29 Aug 2024 16:38:11 INFO  mlp_output size: torch.Size([500, 32])
Thu 29 Aug 2024 16:38:11 INFO  concatenated_output size: torch.Size([500, 224])
Thu 29 Aug 2024 16:38:11 INFO  output size: torch.Size([500, 1])
```

```
2024-08-26 17:02:56,905 - root - INFO - output.squeeze(-1) .shape: torch.Size([500])
```

```
self.actfun = nn.LeakyReLU()
if self.mf_train and self.mlp_train:
            self.predict_layer = nn.Linear(num_mf_towers * self.mf_embedding_size * self.n_contexts_dim
+ self.mlp_hidden_size[-1], 1)

        if self.mf_train and self.mlp_train:
            output = self.actfun(
                self.predict_layer(torch.cat((mf_ui_output, mf_uc_output, mf_ic_output, mlp_output), -1)))
```

Yong Zheng. 2022. A Family of Neural Contextual Matrix Factorization Models for Context-Aware Recommendations. In Adjunct Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization. ACM. https://doi.org/10.1145/3511047.3536404

BaseFactorizationMachine:
BaseFactorizationMachine()
Embedding Size: 64
Number of Feature Fields: 7
MLP Hidden Size: [128, 64, 32]
Size list for MLP layers: [448, 128, 64, 32]