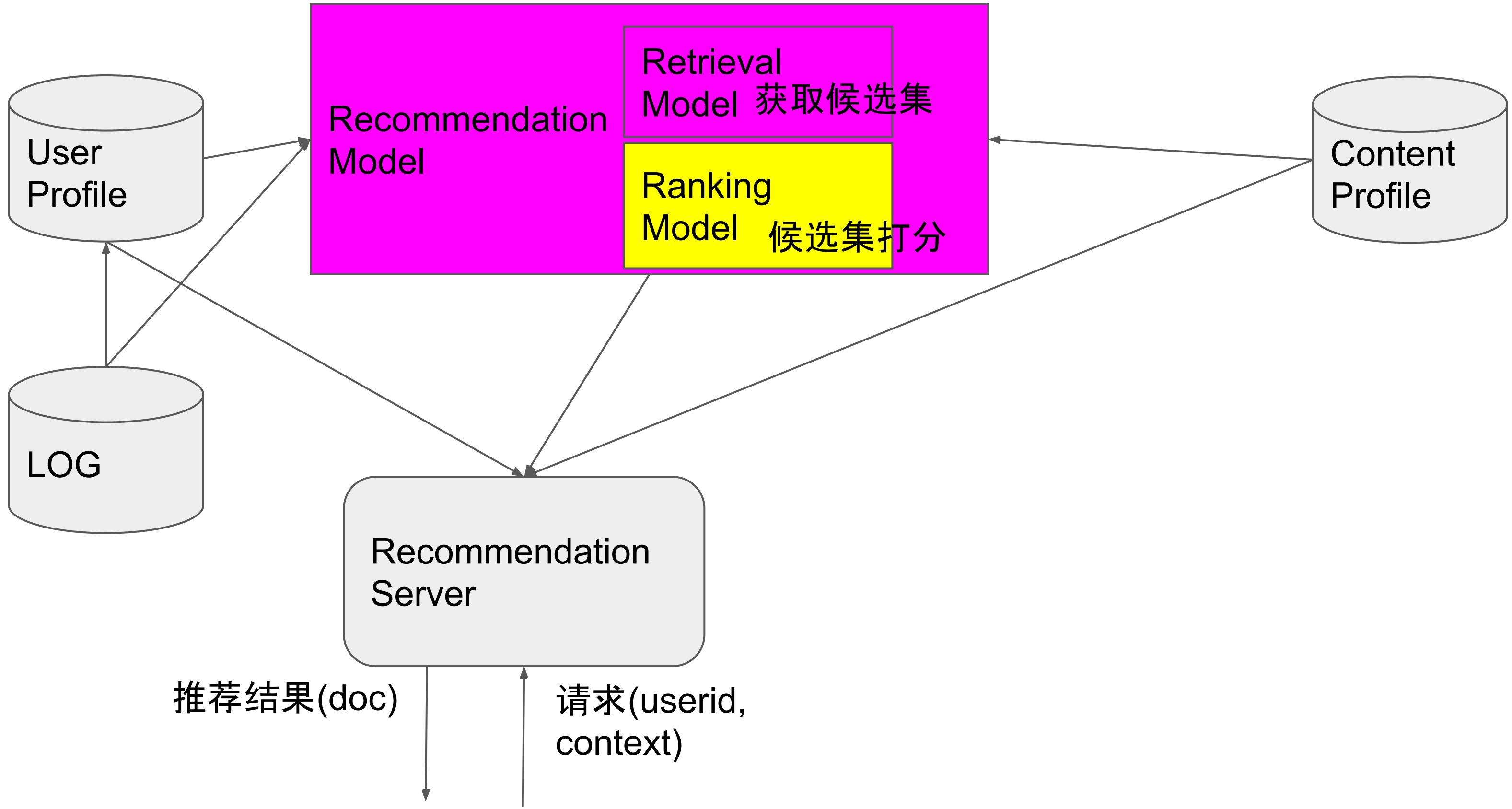
提纲

(1) 推荐领域的尝试,包括NN
(2) tensorflow wide and deep learning model 介绍
(3) 我们是如何使用的
(4) 未来工作

Recommendation Framework



协同过滤

Item-based collaborative filtering

Matrix Factorization: Latent factor vector of user U and item I

核心思想：P(U, I) = f(U,I,C),

Minimize Loss(U, I) = Loss_func(P(U, I), R(U, I))

WALS, SVD++, TimeSVD++, BPMF, BPTF, LDA

Factorization machine:

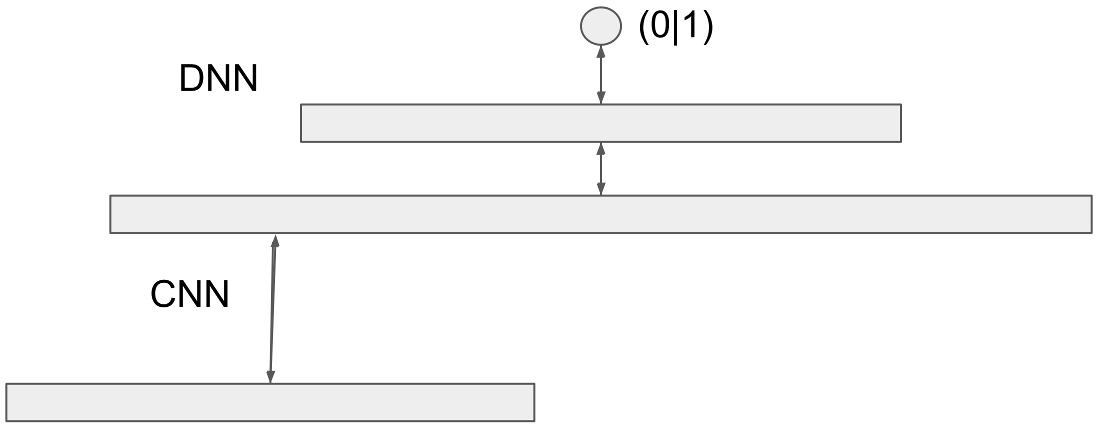
f(y|x) = w0 + SUM{w(i) * x(i)} + SUM{x(i) *x(j) * <v(i), v<j>>}

神经网络

User and Item Embedding

- 基于行为
  - Item2Vec, AutoRec
- 基于内容
  - CNN, AutoEncoder,RNN

DNN as the final classifier



工业界做了哪些

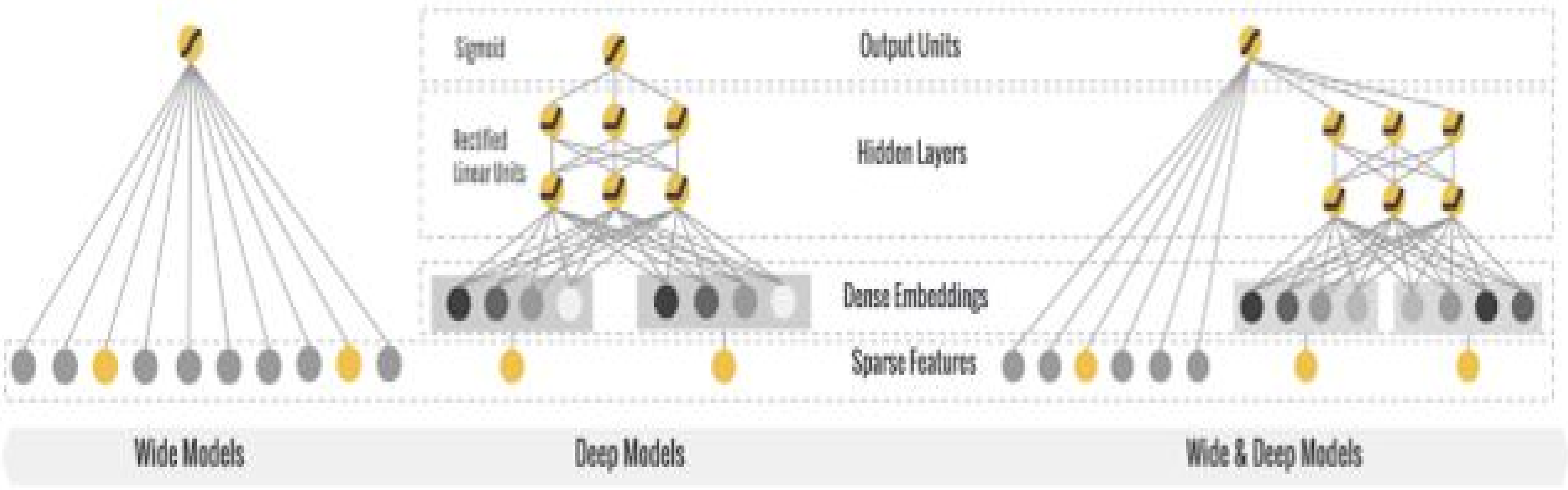
（1）Youtube: DNN for video retrieval and ranking

（2）Spotify: CNN for music content based recommendation

(3) Google: Wide and Deep Learning For Recomender Systems

Wide and Deep

(1) Model
  (a) Wide: Linear
  (b) Deep: DNN
(2) Feature Column
  (a) Categorical
  (b) Continuous
(3) Classification and regression



Wide and deep: How to build a model

1) Define Features

```
gender = sparse_column_with_keys('gender', ['Female', 'Male'])
occupation = sparse_column_with_hash_bucket(column_name = 'occupation', hash_bucket_size=1000)
age = real_valued_column("age")
cross_column = crossed_column(columns = [gender, occupation], hash_bucket_size=1000)
embedding_column =embedding_column(age, dimension=8)
```

2) Define Input

input_fn() -> (features, labels)

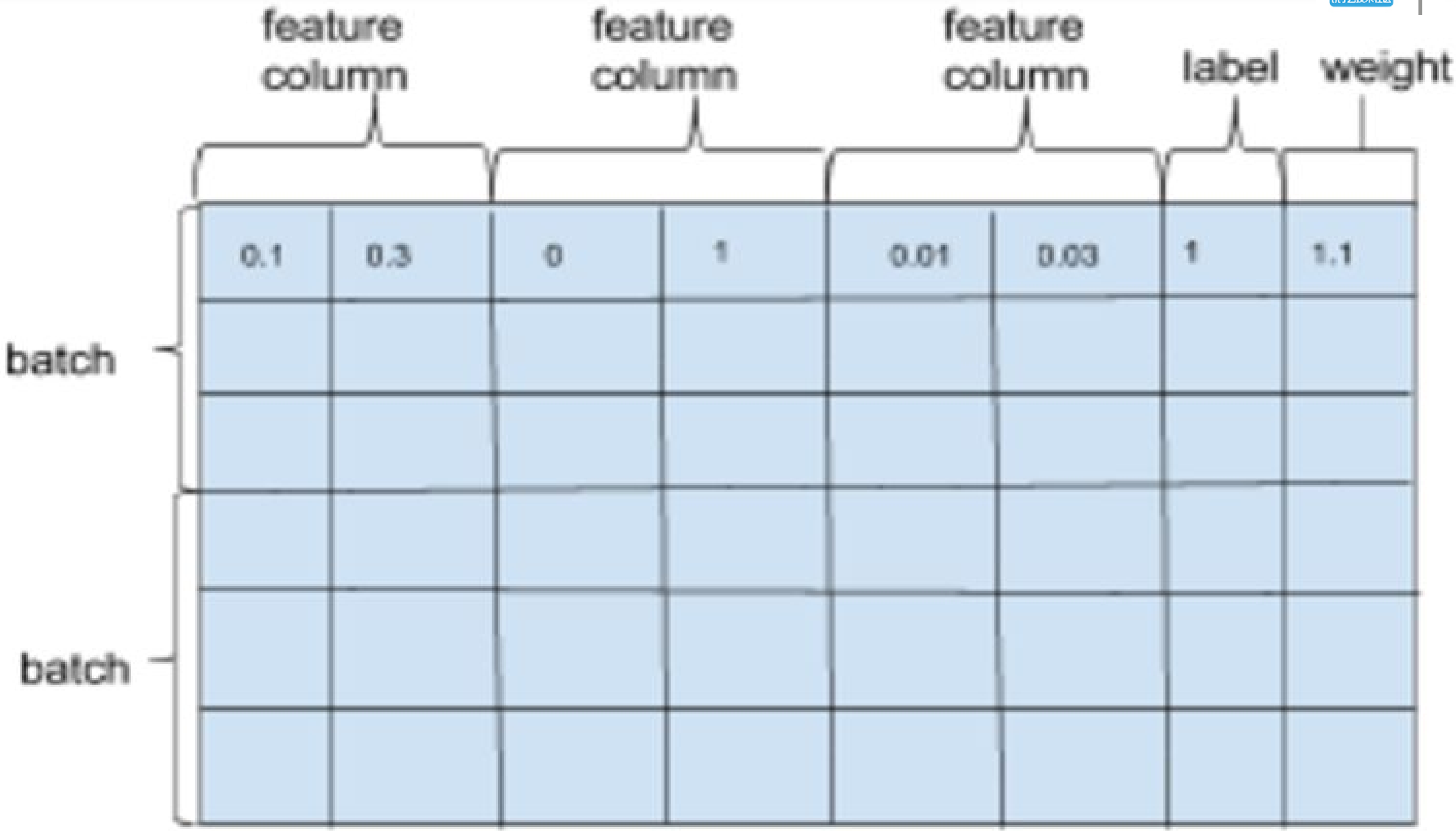
3) Define Model

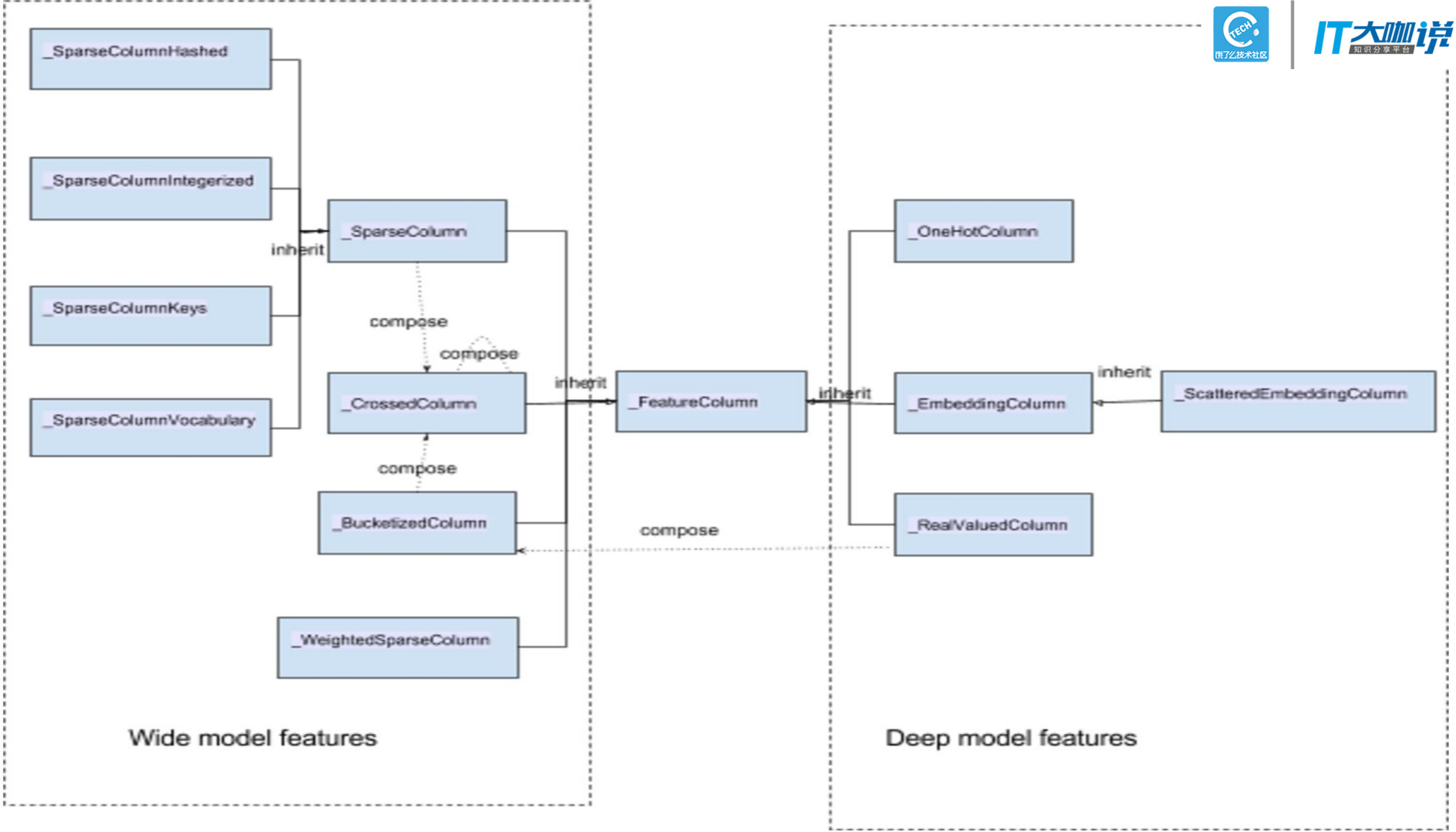
```
deep_columns = [age, embedding_column]
linear_feature_columns = [cross_column]
m = tf.estimator.DNNLinearCombinedClassifier(model_dir=model_dir,
   linear_feature_columns=crossed_columns,
   dnn_feature_columns=deep_columns,
   dnn_hidden_units=[100, 50])
```

4) Train and Evaluate

```
m.train(input_fn=input_fn,steps=train_steps)
m.evaluate(input_fn=input_fn, steps = None)
```

Wide and deep: Features




_SparseColumnHashed
_SparseColumnIntegerized
_SparseColumnKeys
_SparseColumnVocabulary
inherit
_SparseColumn
compose
compose
_CrossedColumn
compose
_BucketizedColumn
_WeightedSparseColumn
inherit
_FeatureColumn
inherit
_OneHotColumn
_EmbeddingColumn
inherit
_ScatteredEmbeddingColumn
_RealValuedColumn
compose
Wide model features
Deep model features

Wide and deep: 模型

1) Wide:

y = w * **x** + b

2) Deep:

hidden(1) = f(w(0) * x’ + b(0))

y’ = f (w(L) * hidden(L) + b(L))

3) Wide and Deep:

prediction = y + y’ + b

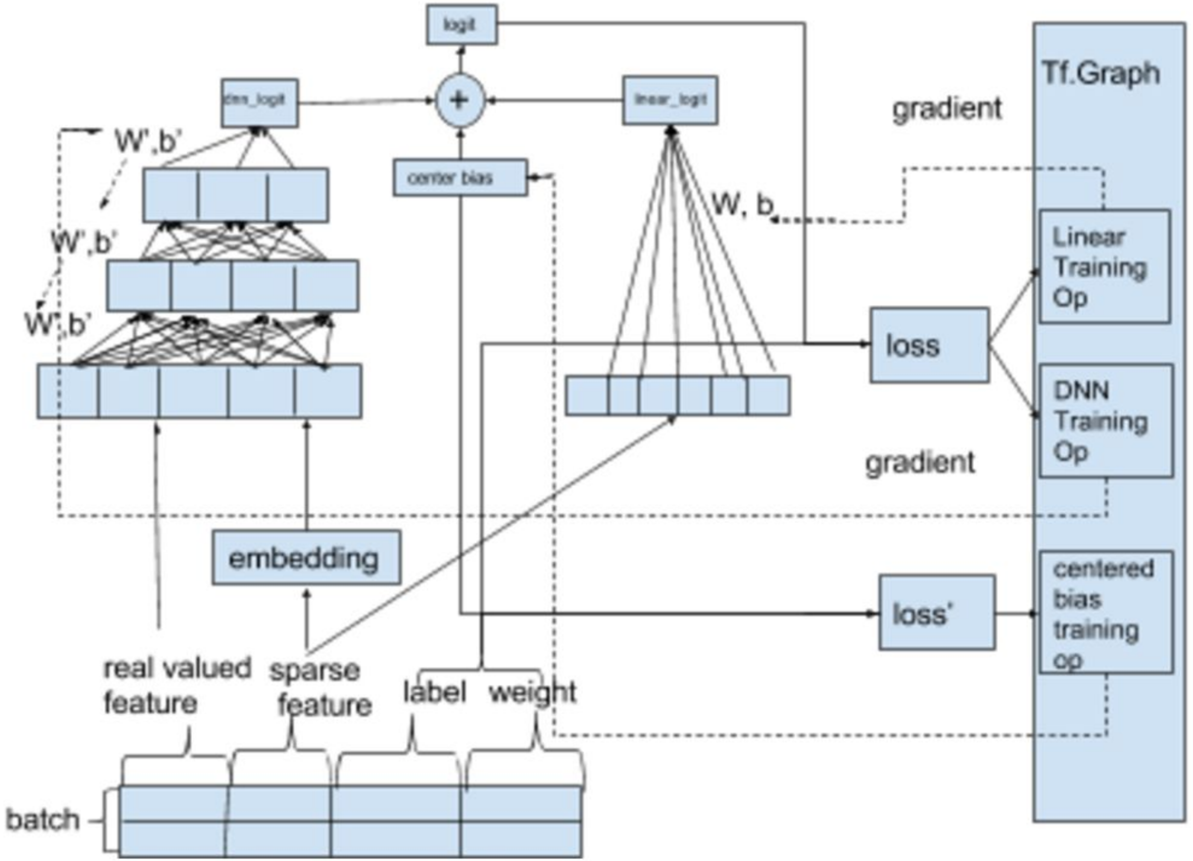
loss = soft_max_cross_entropy (label, prediction)

4) Joint training:

loss -> wide (FTRL)

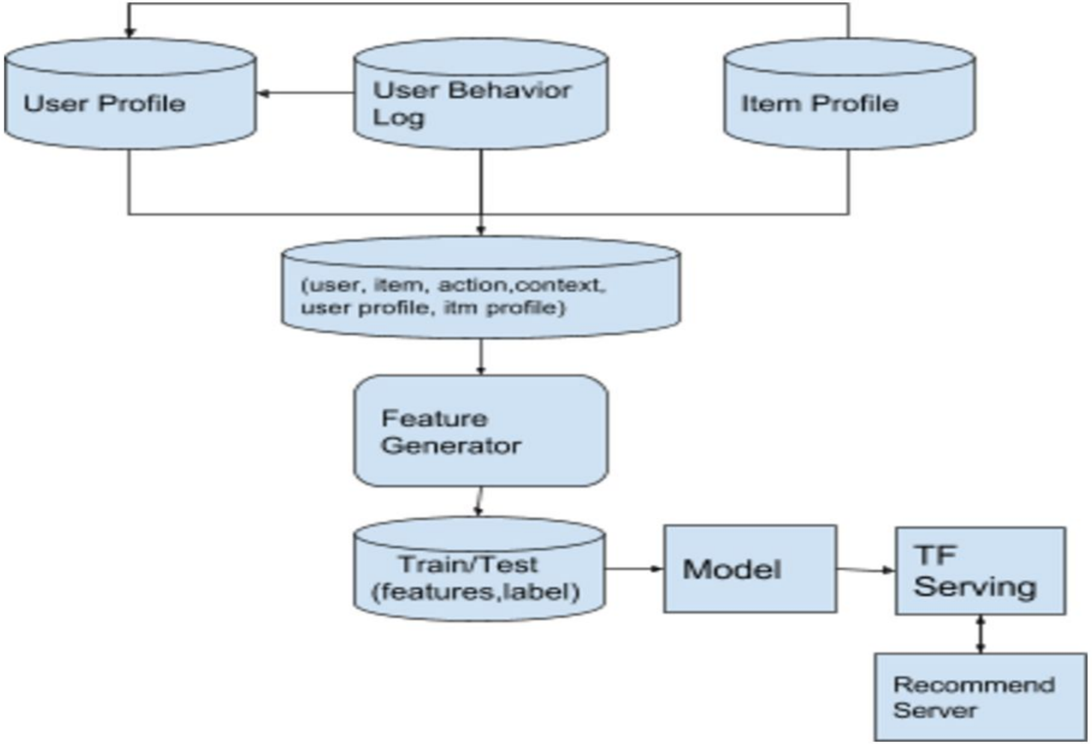
-> deep (Adagrad)

-> bias (Adagrad)

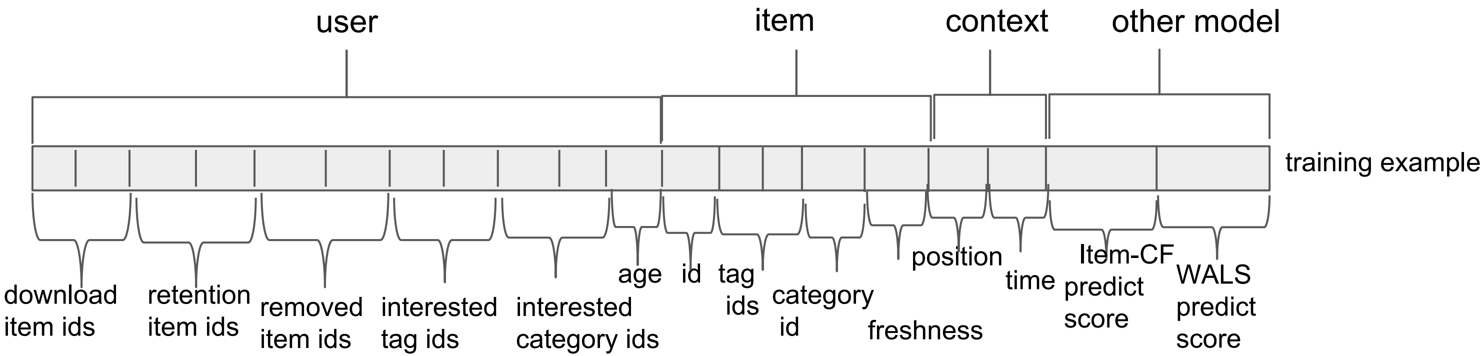


Wide and deep: How do we use it

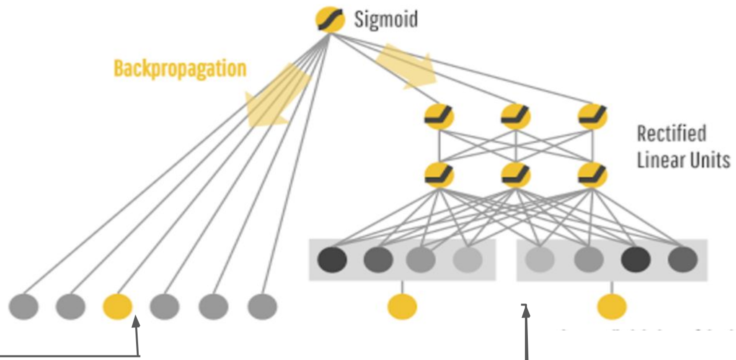
Problem: Given a list of retrieved apps , use wide and deep model to rerank the list to maximize user's app download rate and at the same time increase user retention on these downloaded apps



Wide and Deep: define features



1) Cross Features
   a) user download item id X impression item id
   b) user interested tag id X impression item tag id
   c) user interested category id X impression item tag id
2) Real-Valued Features
   a) item-CF predict score on impressioned item
   b) impression item's recent downloads
   c) impression item freshness
3) Embedding Features
   a) items, tags and category in user profile
   b) impressioned item id, tags and category
   c) click position



Wide and deep: define label, weight

Label: 0|1

Training example weight:

positive sample:

1. 0 + (download_score + retention score) * delete score / sqrt (1 + scale’ * item downloads)

download score = position normalized COEC (deeper download in the list gain higher score)

retention score = LOG(scale * total moving average use time)

delete score = tanh(LOG((delete timestamp - download timestamp) * scale))

negative sample:

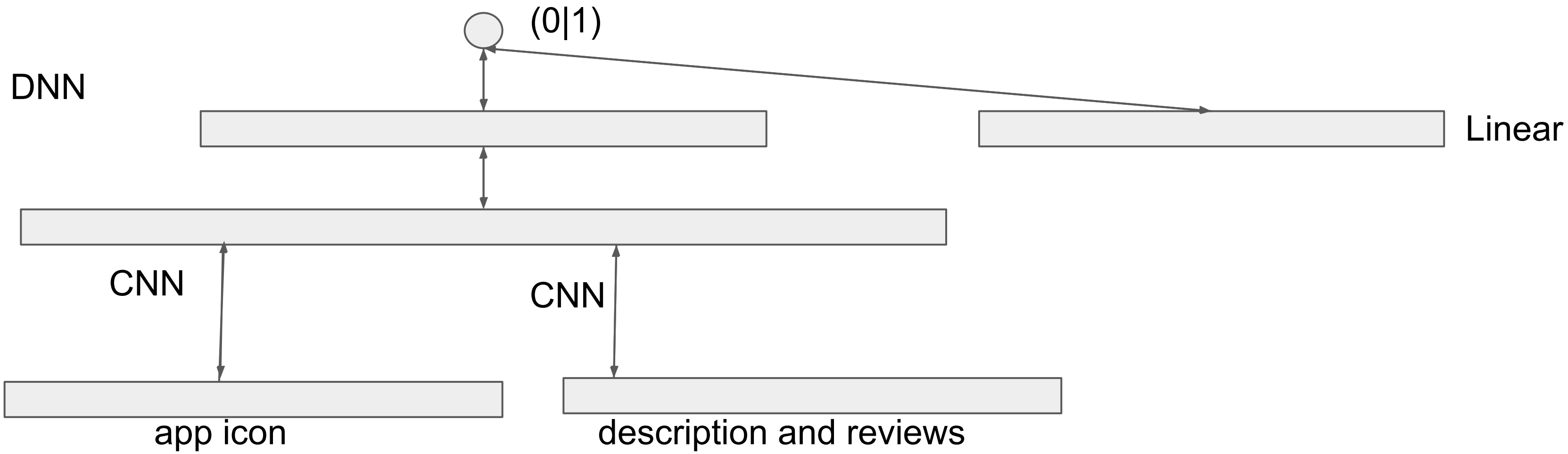
1.0

Wide and deep: train and evaluate model

Data pre-processing:
(1) train / eval split by timestamp (7:3)
(2) remove “extremely hot” users
(3) remove “extremely hot” items
Wide: higher order of feature cross, use user id
Deep:
    other gradient optimizations: eg. Adam
    # of training steps
    # of hidden layers and hidden units
    use batch normalization
    use weight decay
Wide + Deep:
    wide * w1 + deep * w2 + bias
Evaluation Metric:
    AUC (0.94)

Wide and deep: future work

1) Use pre-trained embeddings in the wide and deep framework
   (a) CNN to extract app image and raw textual features as input to the deep model



2) Learning to rank problem