TensorFlow在推荐系统中的应用和实现

李嘉璇

《TensorFlow技术解析与实战》作者
前百度研发工程师

WOTD

World Of Tech
2017年12月1-2日

全球软件开发技术峰会

［深圳站］

报名咨询： 010-68478816
议题提交： wot@51cto.com
市场合作： yangxh@51cto.com
商务合作： songjc@51cto.com
媒体合作： yankk@51cto.com
在线咨询（微信）：18401576051

团·购·享·受·更·多·优·惠

5折 优惠（截止8月31日）
现在报名，立省1400元 / 张

主题大纲

01 推荐引擎架构及策略演进

02 深度学习引入推荐系统的架构和实现

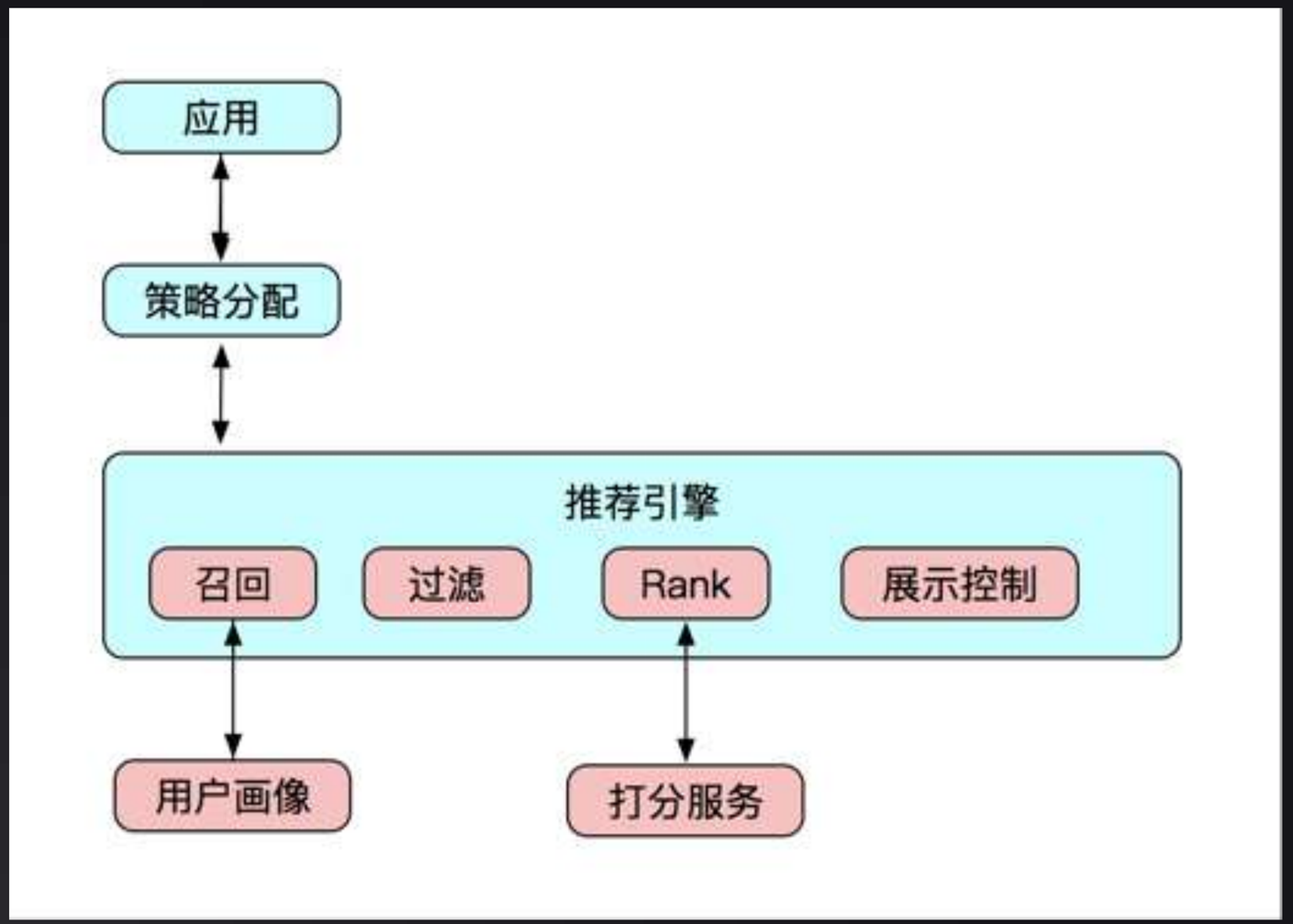
03 强化学习的分类

04 强化学习引入推荐系统的实现

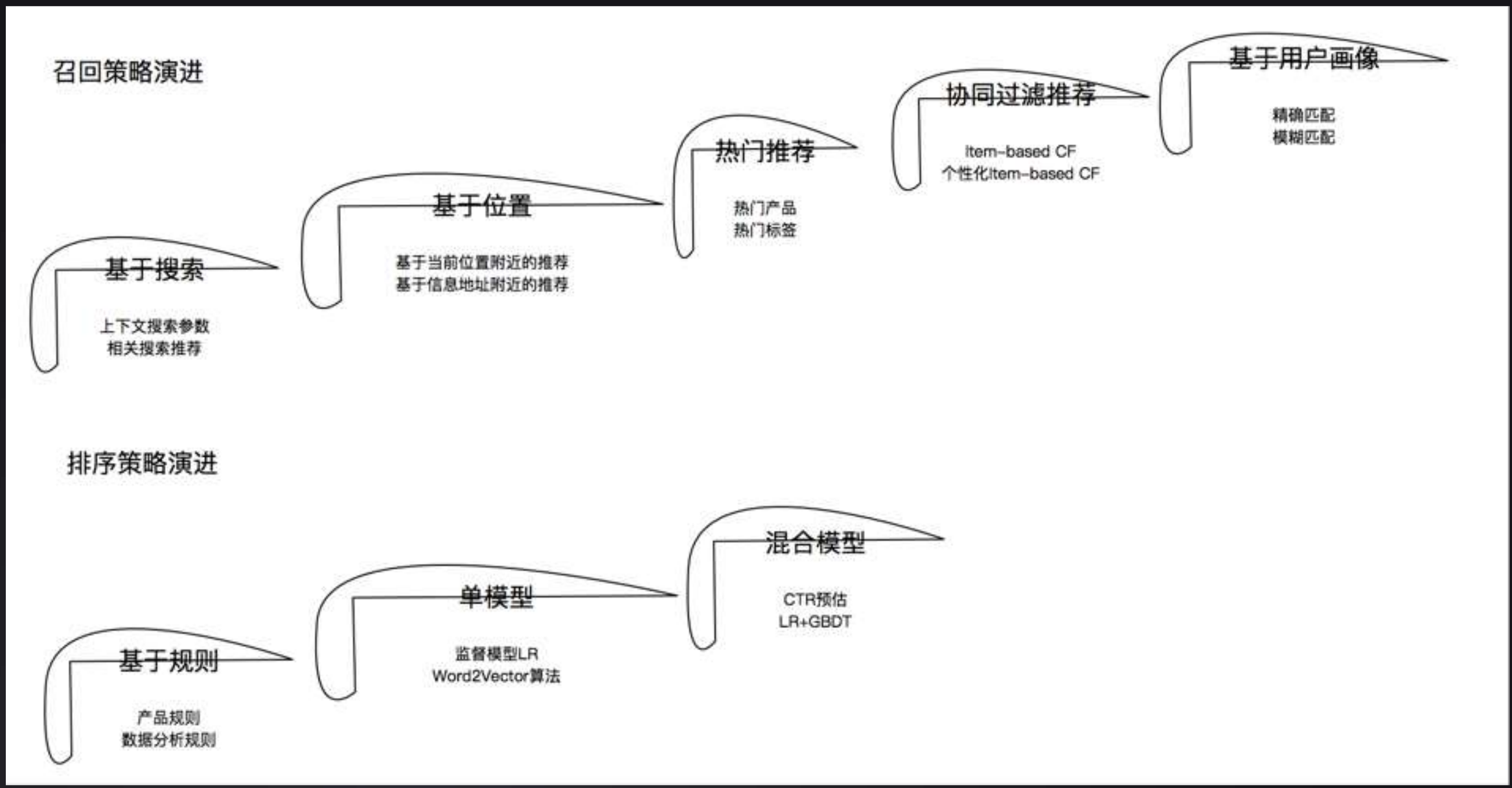
个人微信

推荐引擎架构

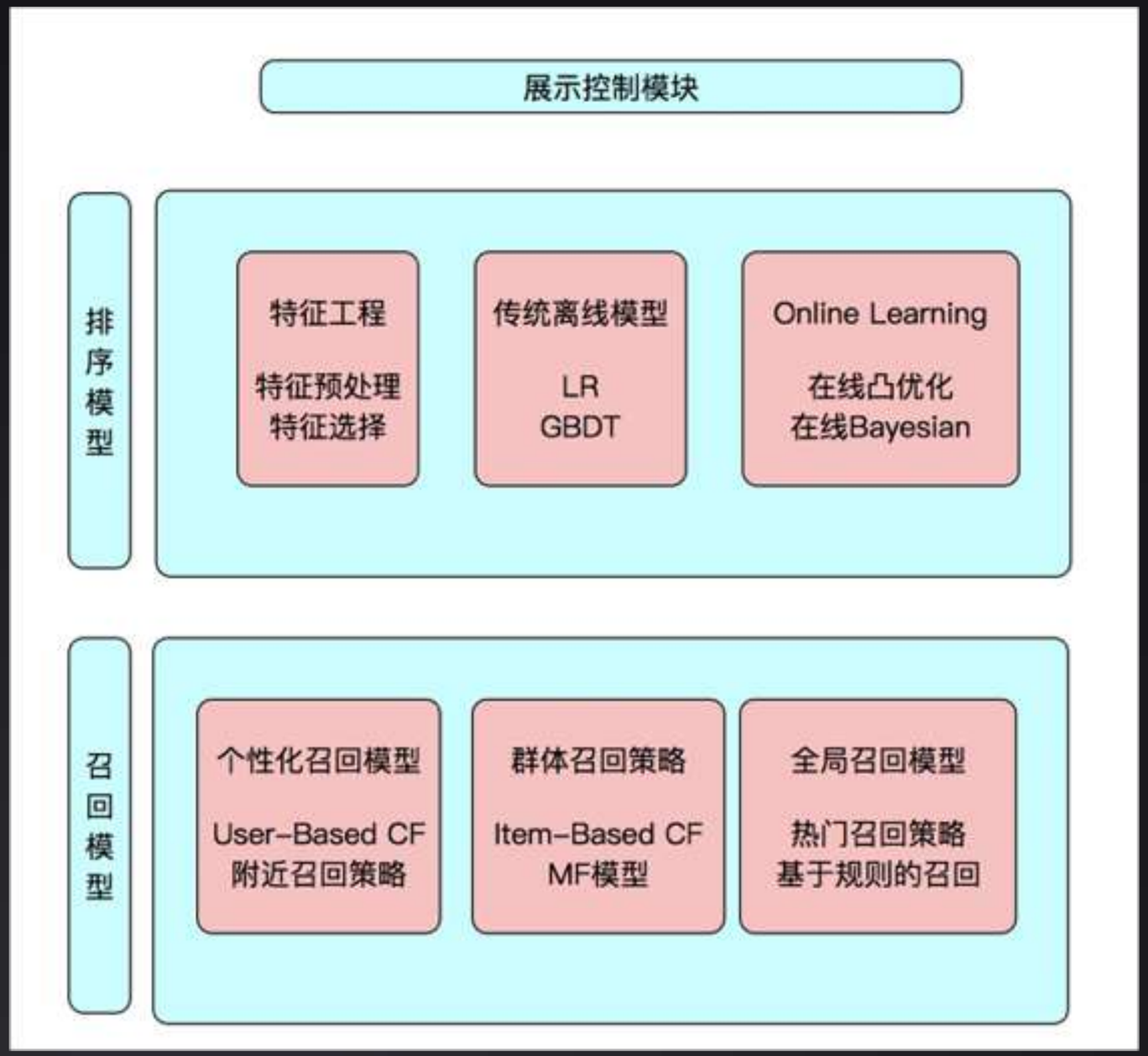
- 召回
  - 场景区分
  - 多策略融合
  - 策略降级
- 过滤
  - 低质量信息/黑名单/已查看信息
- Rank
  - 产品规则/个性化排序
- 展示控制
  - 多样性控制/推荐理由



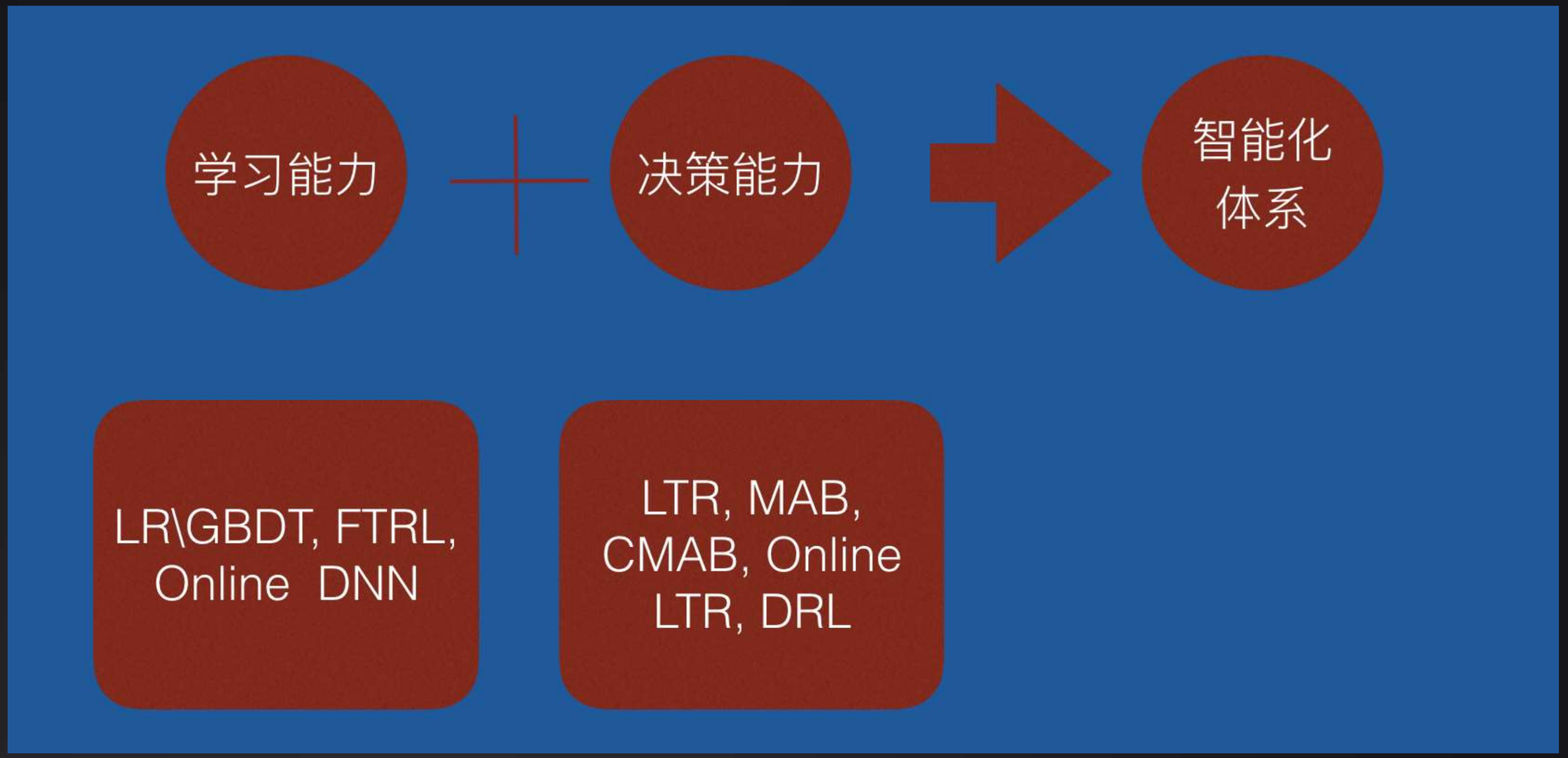
推荐引擎策略演进


召回策略演进
基于搜索
上下文搜索参数
相关搜索推荐
基于位置
基于当前位置附近的推荐
基于信息地址附近的推荐
热门推荐
热门产品
热门标签
协同过滤推荐
Item-based CF
个性化Item-based CF
基于用户画像
精确匹配
模糊匹配
排序策略演进
基于规则
产品规则
数据分析规则
单模型
监督模型LR
Word2Vector算法
混合模型
CTR预估
LR+GBDT

推荐引擎算法模型

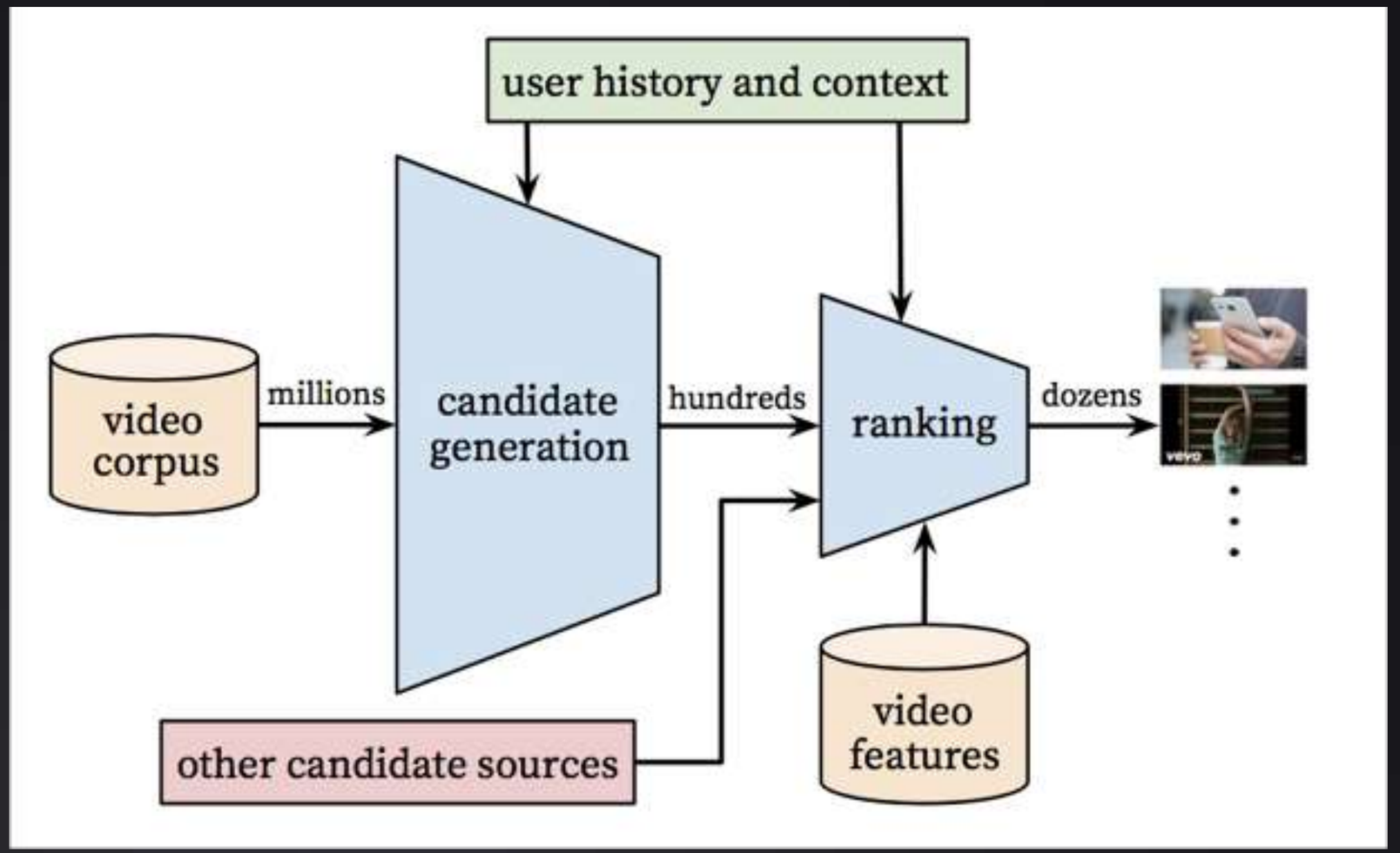

展示控制模块
排序模型
特征工程
特征预处理
特征选择
传统离线模型
LR
GBDT
Online Learning
在线凸优化
在线Bayesian
召回模型
个性化召回模型
User-Based CF
附近召回策略
群体召回策略
Item-Based CF
MF模型
全局召回模型
热门召回策略
基于规则的召回

智能化时代的搜索和推荐



Deep Neural Networks for YouTube Recommendations

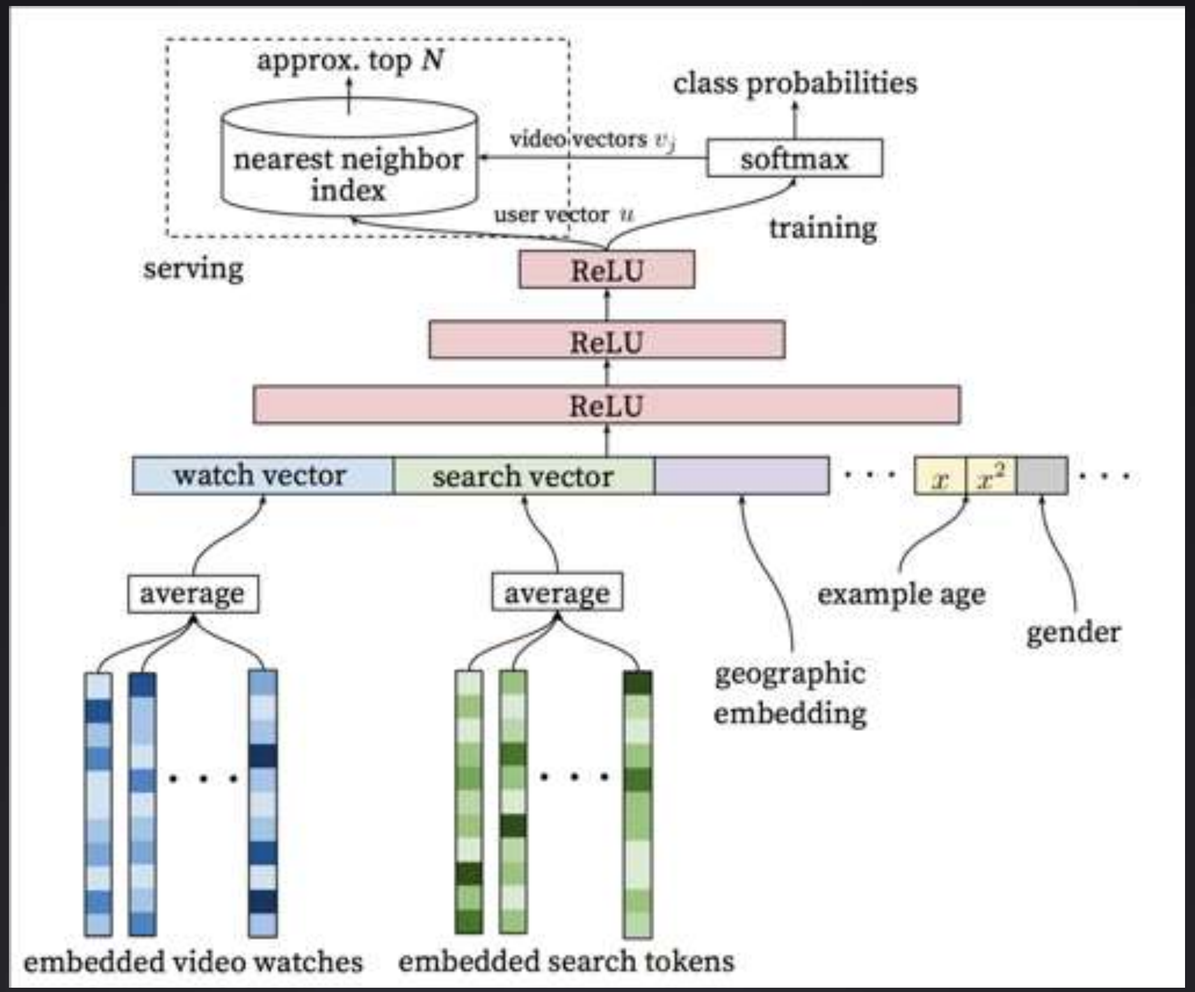
推荐引擎架构



Deep Neural Networks for YouTube Recommendations

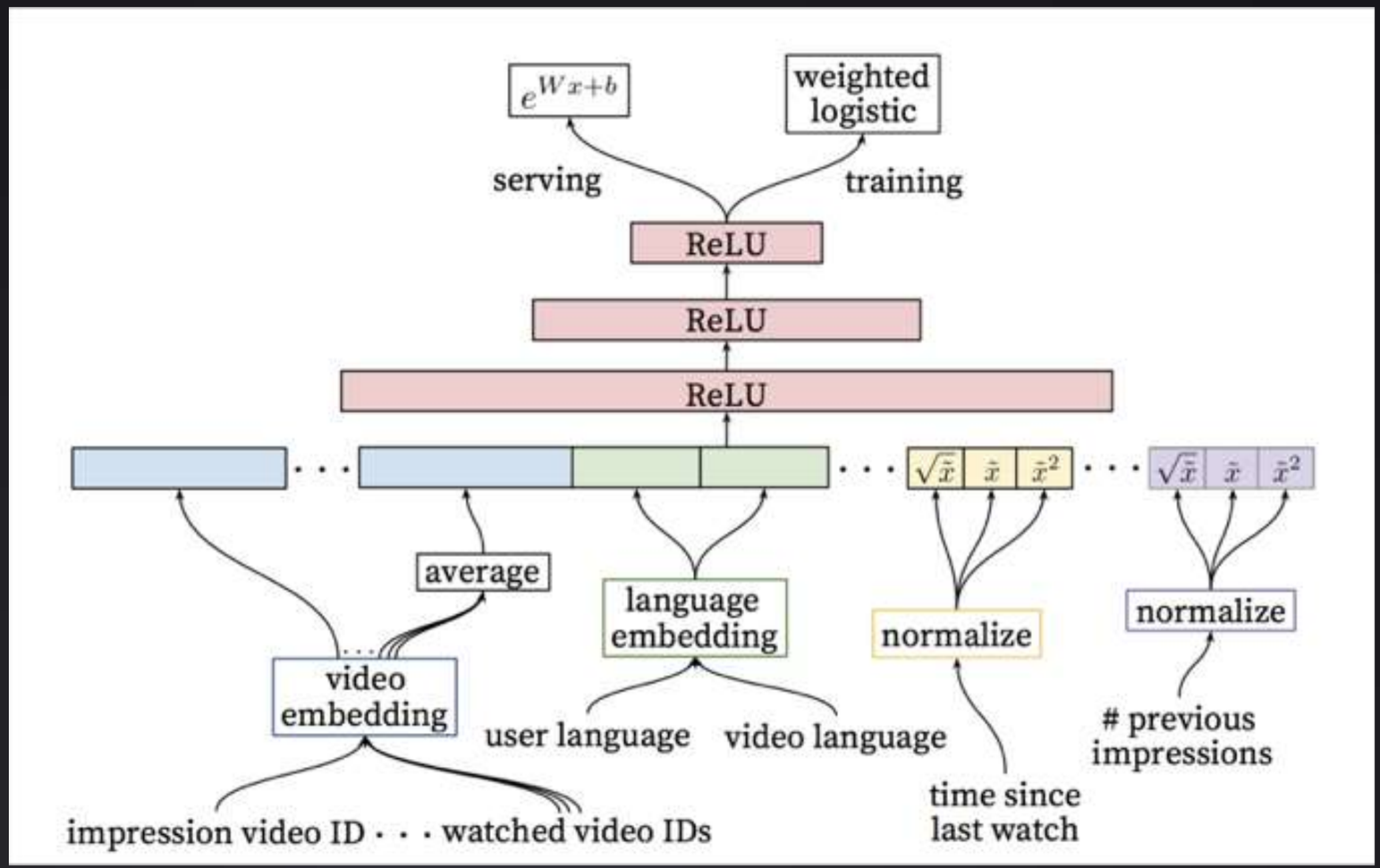
CANDIDATE GENERATION

$$P(w_t = i|U, C) = \frac{e^{v_i u}}{\sum_{j \in V} e^{v_j u}}$$



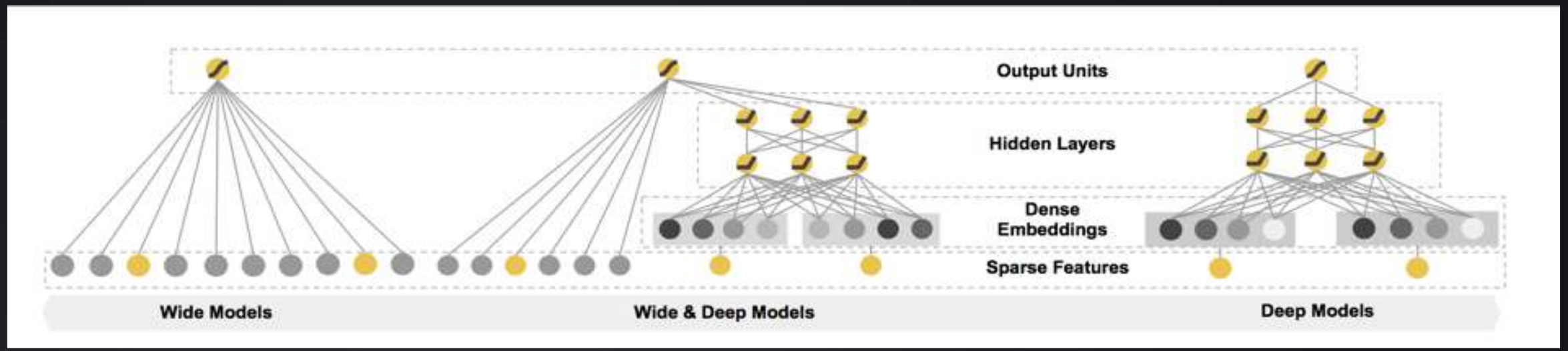
Deep Neural Networks for YouTube Recommendations

RANKING

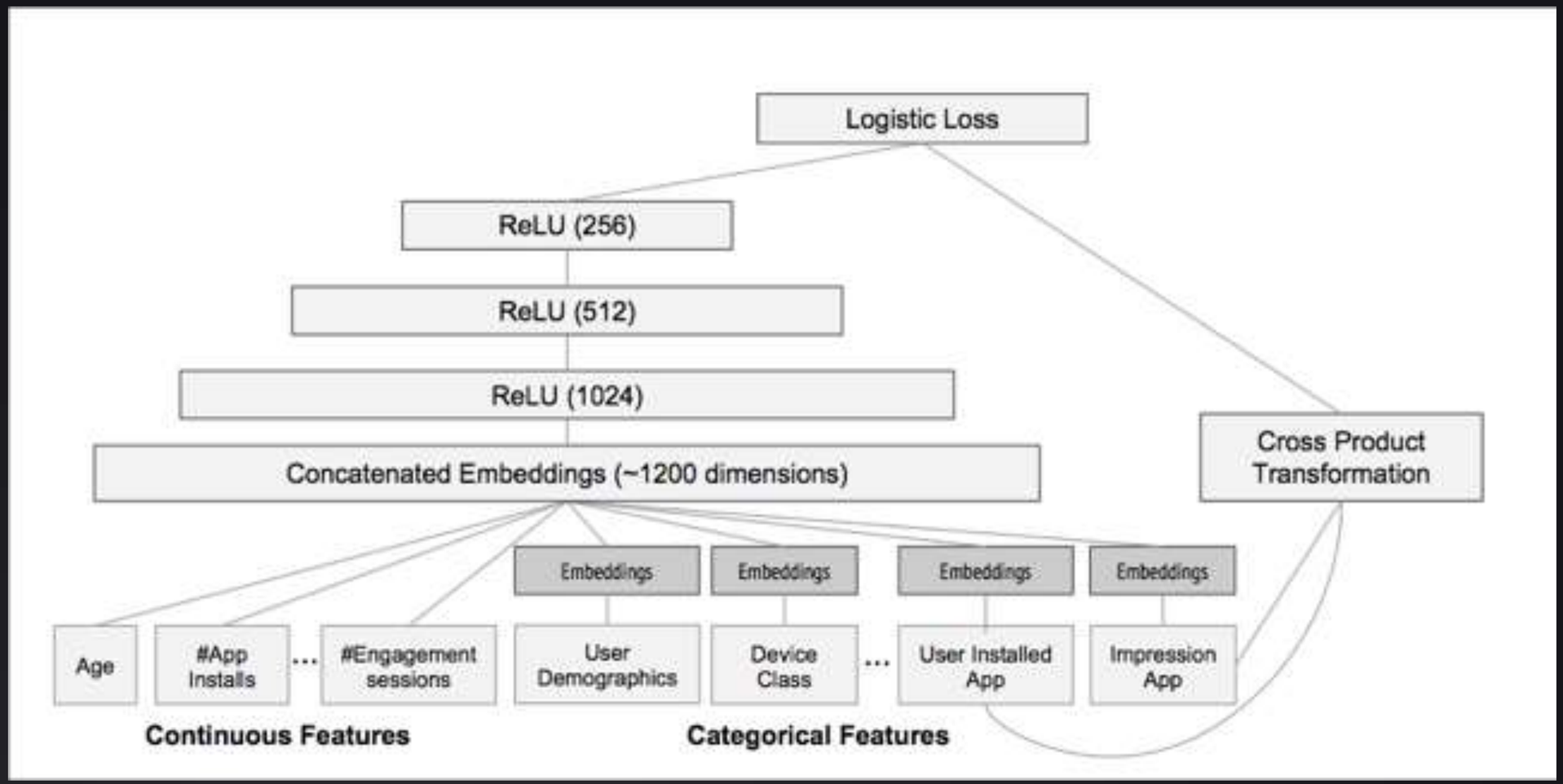


Wide & Deep Learning模型

- 将深度神经网络(DNN)和逻辑回归(Logistic Regression)模型并置在同一个网络中
- 将离散型特征(Categorical Feature)和连续型特征(Continuous Feature)有机地结合在一起



Wide & Deep Learning模型



$$P(Y=1|\mathbf{x}) = \sigma(\mathbf{w}_{wide}^T[\mathbf{x}, \phi(\mathbf{x})] + \mathbf{w}_{deep}^T a^{(l_f)} + b)$$

Wide & Deep 模型的TensorFlow实现

- **Wide Model: Linear Model with Crossed Feature Columns**

```
wide_columns = [gender, native_country, education, occupation, workclass,
                relationship, age_buckets,
                tf.contrib.layers.crossed_column([education, occupation],
                                                 hash_bucket_size=int(1e4)),
                tf.contrib.layers.crossed_column(
                    [age_buckets, education, occupation],
                    hash_bucket_size=int(1e6)),
                tf.contrib.layers.crossed_column([native_country, occupation],
                                                 hash_bucket_size=int(1e4))]
```

Wide & Deep 模型的 TensorFlow实现

- Deep Model: Neural Network with Embeddings

```
deep_columns = [
    tf.contrib.layers.embedding_column(workclass, dimension=8),
    tf.contrib.layers.embedding_column(education, dimension=8),
    tf.contrib.layers.embedding_column(gender, dimension=8),
    tf.contrib.layers.embedding_column(relationship, dimension=8),
    tf.contrib.layers.embedding_column(native_country,
                                       dimension=8),
    tf.contrib.layers.embedding_column(occupation, dimension=8),
    age,
    education_num,
    capital_gain,
    capital_loss,
    hours_per_week,
]
```

Wide & Deep 模型的 TensorFlow实现

- Combining Wide and Deep Models into One

```
if model_type == "wide":
  m = tf.contrib.learn.LinearClassifier(model_dir=model_dir,
                                        feature_columns=wide_columns)
elif model_type == "deep":
  m = tf.contrib.learn.DNNClassifier(model_dir=model_dir,
                                     feature_columns=deep_columns,
                                     hidden_units=[100, 50])
else:
  m = tf.contrib.learn.DNNLinearCombinedClassifier(
      model_dir=model_dir,
      linear_feature_columns=wide_columns,
      dnn_feature_columns=deep_columns,
      dnn_hidden_units=[100, 50],
      fix_global_step_increment_bug=True)
return m
```

排序方法

- Learning to Rank（LTR）
  - 在商品维度进行学习，根据商品的点击、成交数据构造学习样本，回归出排序权重
  - 有大量的样本是不可见的，所以LTR模型从某种意义上说是解释了过去现象，并不一定真正全局最优的

两种解决方案

- 在离线训练中解决 online 和 offline 不一致的问题
  - Counterfactual Machine Learning
- 在线 trial-and-error 进行学习
  - Bandit Learning 和 Reinforcement Learning

强化学习简介

- 把推荐系统看作智能体（Agent）、把用户看做环境（Environment），则商品的推荐问题可以被视为典型的顺序决策问题。Agent每一次排序策略的选择可以看成一次试错（Trial-and-Error），把用户的反馈，点击成交等作为从环境获得的奖赏。

强化学习分类

- Model-free 和 Model-based

Q-learning, Sarsa,
Policy Gradients

- Policy-Based 和 Value-Based

policy gradients

Q-learning, sarsa

强化学习分类

- 回合更新和单步更新

Monte-carlo learning 和基础版的 policy gradients

Qlearning, Sarsa, 升级版的 policy gradients

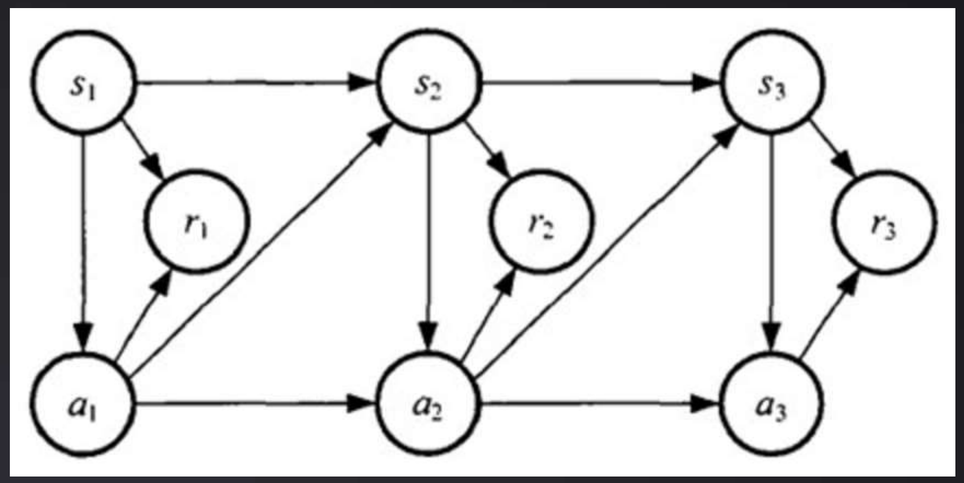
- 在线学习 和 离线学习

- sarsa, sarsa lambda

Q learning, Deep-Q-Network

问题建模

MDP可以由一个四元组<S, A, R, T>表示：
（1）S为状态空间（State Space）；
（2）A为动作空间（Action Space）；
（3）R为奖赏函数；
（4）T为环境状态转移函数（State Transition Function）



问题建模

- 状态定义
  - 在每一个PV请求发生时，把在最近一段时间内点击的商品的特征作为当前Agent感知到的状态
- 奖赏函数定义
  - 用户根据排序的结果进行的浏览、商品点击或购买等行为都可以看成对Agent的排序策略的直接反馈

问题建模

- 算法设计
  - Q-learning（Tabular）

	left	right
0	0.0	0.0
1	0.0	0.0
2	0.0	0.0
3	0.0	0.0
4	0.0	0.0
5	0.0	0.0

Q-learning的算法实现

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q (e.g., ε-greedy)
    Take action a, observe r, s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
    $s \leftarrow s'$;
  until s is terminal

```python
def rl():
    q_table = build_q_table(N_STATES, ACTIONS)  # 初始 q table
    for episode in range(MAX_EPISODES):     # 回合
        step_counter = 0
        S = 0   # 回合初始位置
        is_terminated = False   # 是否回合结束
        update_env(S, episode, step_counter)    # 环境更新
        while not is_terminated:

            A = choose_action(S, q_table)   # 选行为
            S_, R = get_env_feedback(S, A)  # 实施行为并得到环境的反馈
            q_predict = q_table.ix[S, A]    # 估算的(状态-行为)值
            if S_ != 'terminal':
                q_target = R + GAMMA * q_table.iloc[S_, :].max()   #  实际的(状态-行为)
            else:
                q_target = R     #  实际的(状态-行为)值 (回合结束)
                is_terminated = True    # terminate this episode

            q_table.ix[S, A] += ALPHA * (q_target - q_predict)  #  q_table 更新
            S = S_  # 探索者移动到下一个 state

            update_env(S, episode, step_counter+1)  # 环境更新

            step_counter += 1
    return q_table
```

问题建模优化

- 奖赏塑形（Reward Shaping）

$$R(s,a,s') = R_0(s,a,s') + \Phi(s)$$

- 势函数（Potential Function）：学习过程中的子目标（Local Objective）

- 把每个状态对应PV的商品信息纳入Reward的定义中，将势函数定义为

$$\Phi(s) = \sum_{i=1}^{K} \mathbb{L}(i|\mu_\theta(s))$$

问题建模

- 在单商品的推荐场景，a对应的是单个商品。我们的目标是学习在状态s下采取动作a所能获得的累积奖励（的期望值）

$$Q(s,a) = \mathbb{E}[R|s,a]$$

- 多商品推荐场景：假设用户是否会点击单商品的决策是独立的

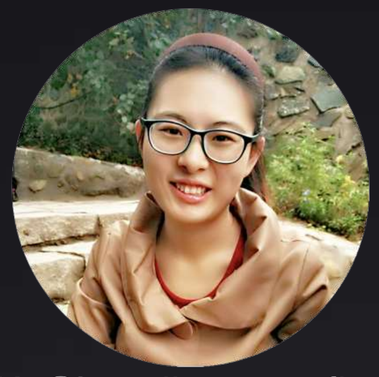
$$f(s,i) = I(s_i)[r_i + \gamma \sum_{j \in a_i} f(s_i,j)]$$

Actor-Critic

- 结合了 Policy Gradient (Actor) 和 Function Approximation (Critic) 的方法. Actor 基于概率选行为, Critic 基于 Actor 的行为评判行为的得分, Actor 根据 Critic 的评分修改选行为的概率
- Actor-model 在某种意义上，我们看成是一个从 state生成 action的 Generative Model
- Critic-model 看成是基于状态-策略输入下的 Q 值回归网络的 Discriminative Model
- 整个数据理解和建模过程就通过这样的系统新运作方式来使得生成式模型更好地去发现「未知」世界中的True Positive 样本

更多强化学习方法

- Deep Deterministic Policy Gradient（DDPG）
- Asynchronous Advantage Actor-Critic (A3C)

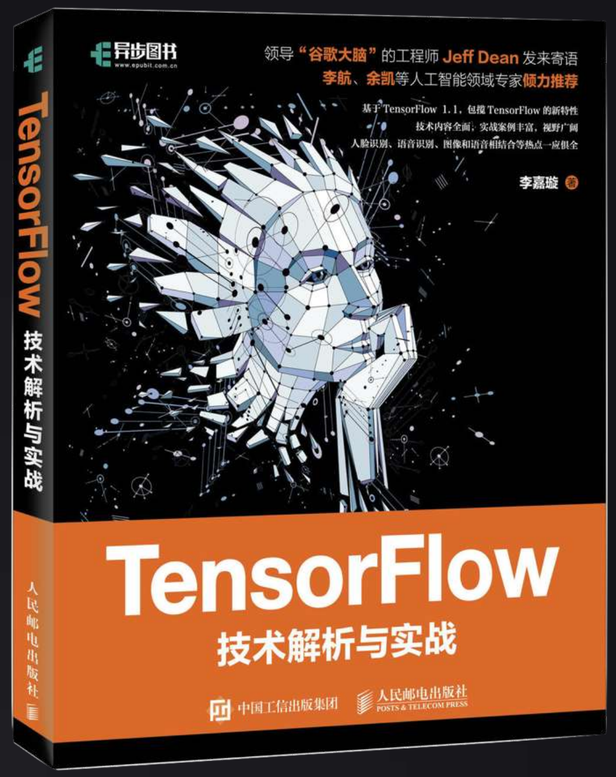


WeChat: ljxtensorflow

Email:qiyueli_2013@gmail.com

微博：李嘉璇的微博

个人微信



签售

THANK YOU