A 1st introduction to

Large Scale Computing

Concepts & Techniques

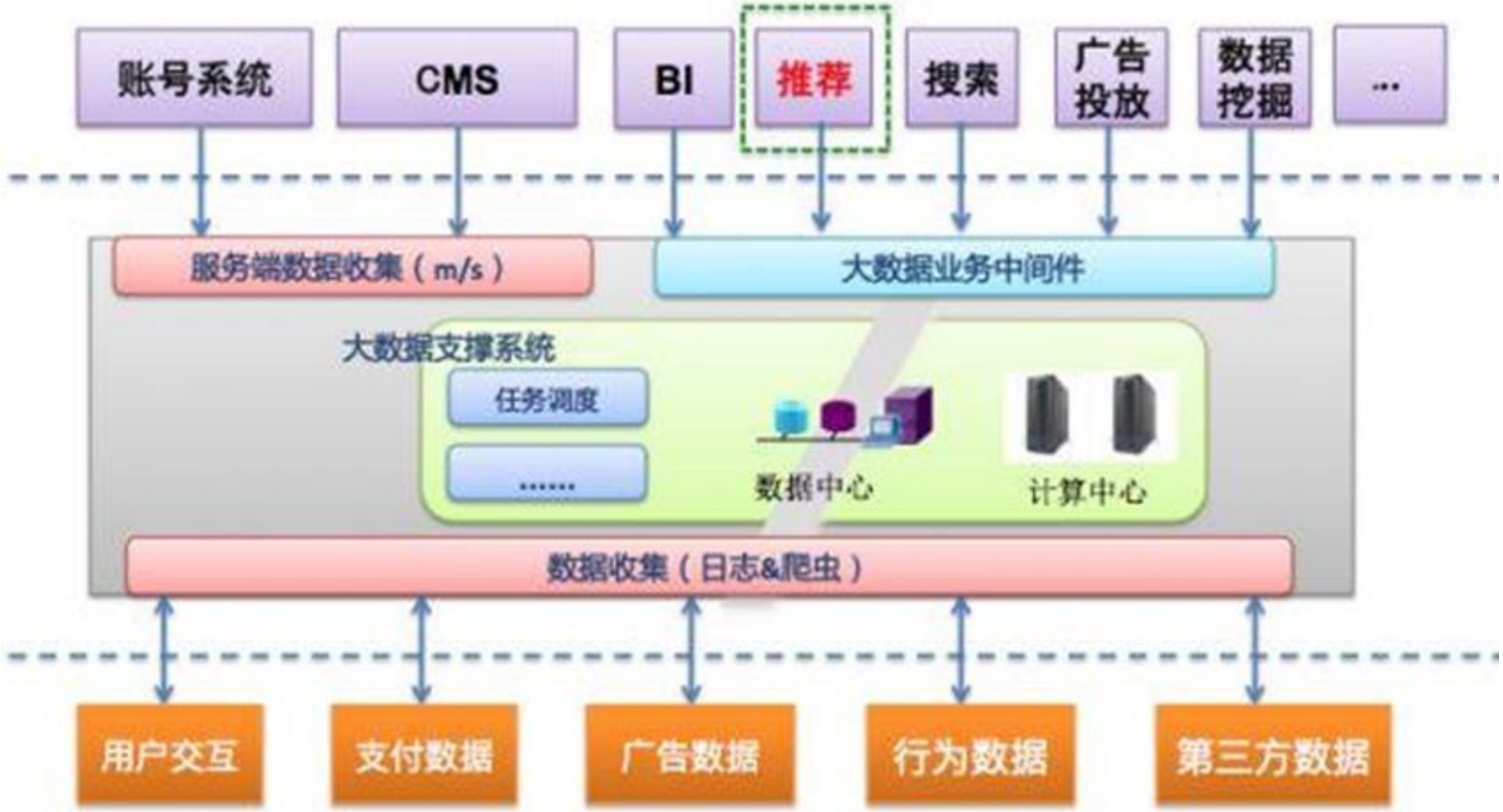
Chapter 6: Recommending System

孔令波
mlinking@126.com
+86 15010255486

大纲

- **前言**
  - 为什么需要“大规模计算” [HPC, DL, Business platform system, Cloud已经合流]
    - 导入 – 科学计算(天气预报)，DL, 互联网平台(Google, Amazon, Alibaba, MeiTuan, …)
- **基础篇**
  - 并发程序的样子 – Divide & Conquer, Model & Challenges, PCAM, Data/Task, …
    - 天气预报的计算
  - 运行环境
    - 硬件 – 自己梳理的3个方案 – Shared/Unshared Memory, Hybrid
    - 系统软件 – 协议栈, Modern OS, Distributed Job Scheduler, GTM等
- **算法级篇**
  - OpenMP, MPI, CUDA (DL的实现), Big Data 中的MR/Spark等 (只涉及在Big Data SDK之上的编程；大数据本身的介绍放到后一部分)
- **系统级篇 – 互联网平台的实现**
  - “秒杀”的技术架构
  - 计算广告
  - 系统架构 (HTAP等)
    - Flink, ClickHouse, MaxCompute, ELK …

Two valuable challenges – Targeting/Precision Advertising

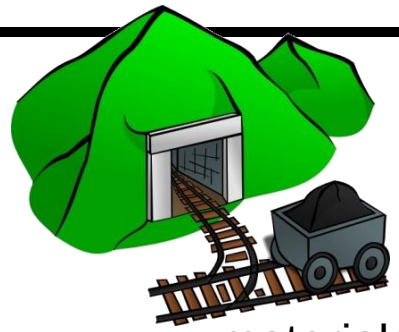


https://www.infoq.cn/article/V0wtItKL7BAIyJ-WKoz8

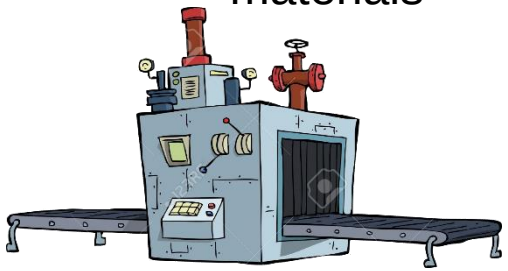
“精准广告”

- **About Recommendation and computing adv [关于计算广告]**
  - From Recommendation to Computing Advs
  - You should know how to earn money from ads in Google! – **RTB** (Real Time Bargain)
- **Algorithms**
  - CF, FM, LR, GBDT-LR (2014), Embedding, LDA (Latent Dirichlet Analysis) with EM, DL
    - Feature engineering is critical
- **Big Data way**

Old ways



materials



machines

workers

How to target the customers efficiently

Advertising/Recommending

The key is To recommend something to some one

Modern ways Data driven modeling to figure out the target customers – **Computing Adv. & Recommendation**

Google

Products/Services

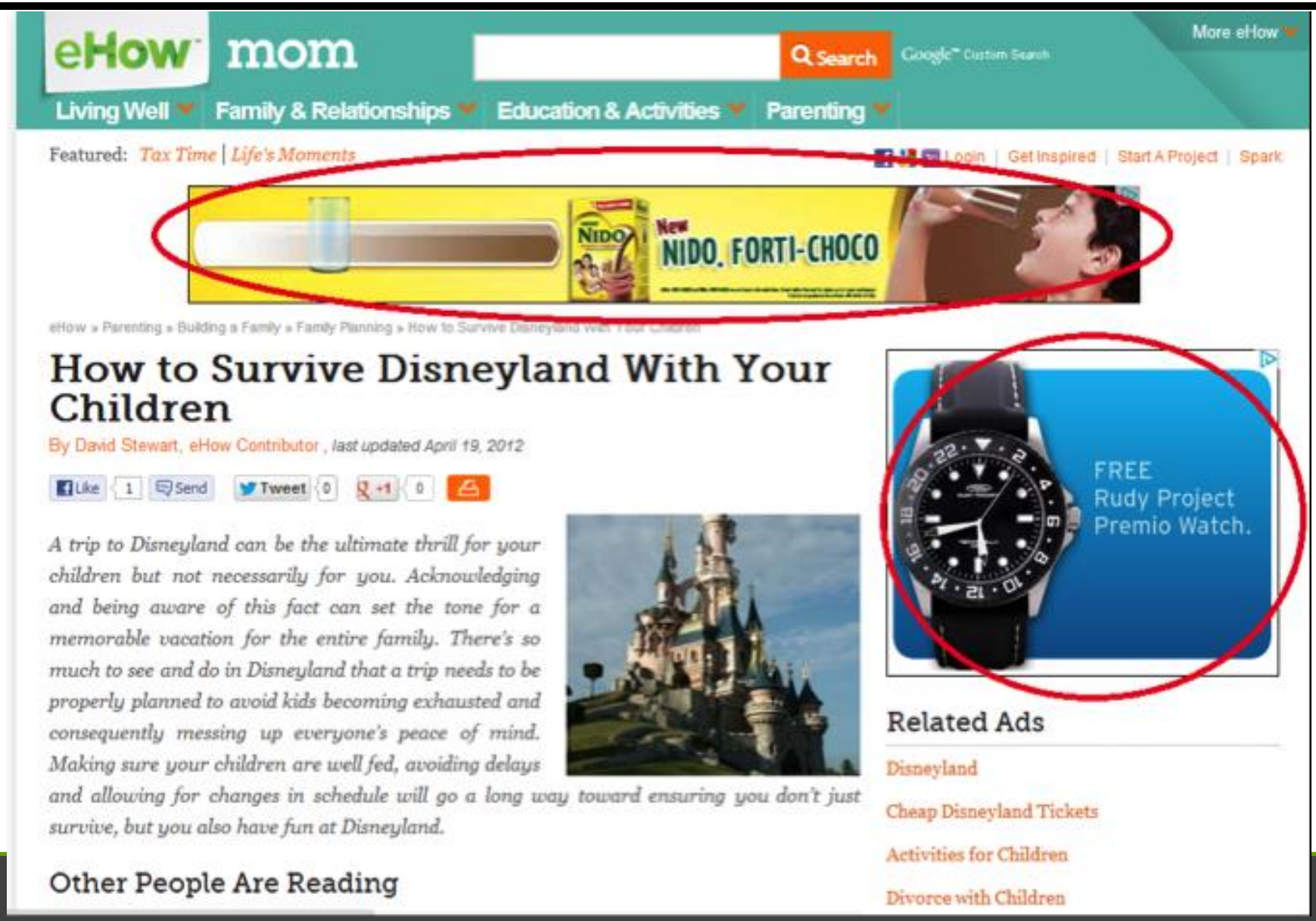
Customers

Short history about online recommendation

- **1978, Gary Thuerk with Digital Equipment Corporation (DEC) sent the 1st email recommendation letter to introduce the DEC computers**
- **1991, the Internet starts**
- **Oct 1994，Hot Wired sold AT&T a 468*60 Banner adv，the 1st online banner adv.**
  - This is a milestone in adv history，representing the start of “banner advertising”
- **1997, Reed Hastings and Marc Randolph founded Netflix (Kibble)，a DVD renting company**
- **1998, Amazon used item based CF (基于物品的协同过滤算法: ItemCF算法)**
- **Oct 2000, Google released Adwords – by using keyword bidding and CPM as Settlement method [结算方式]**

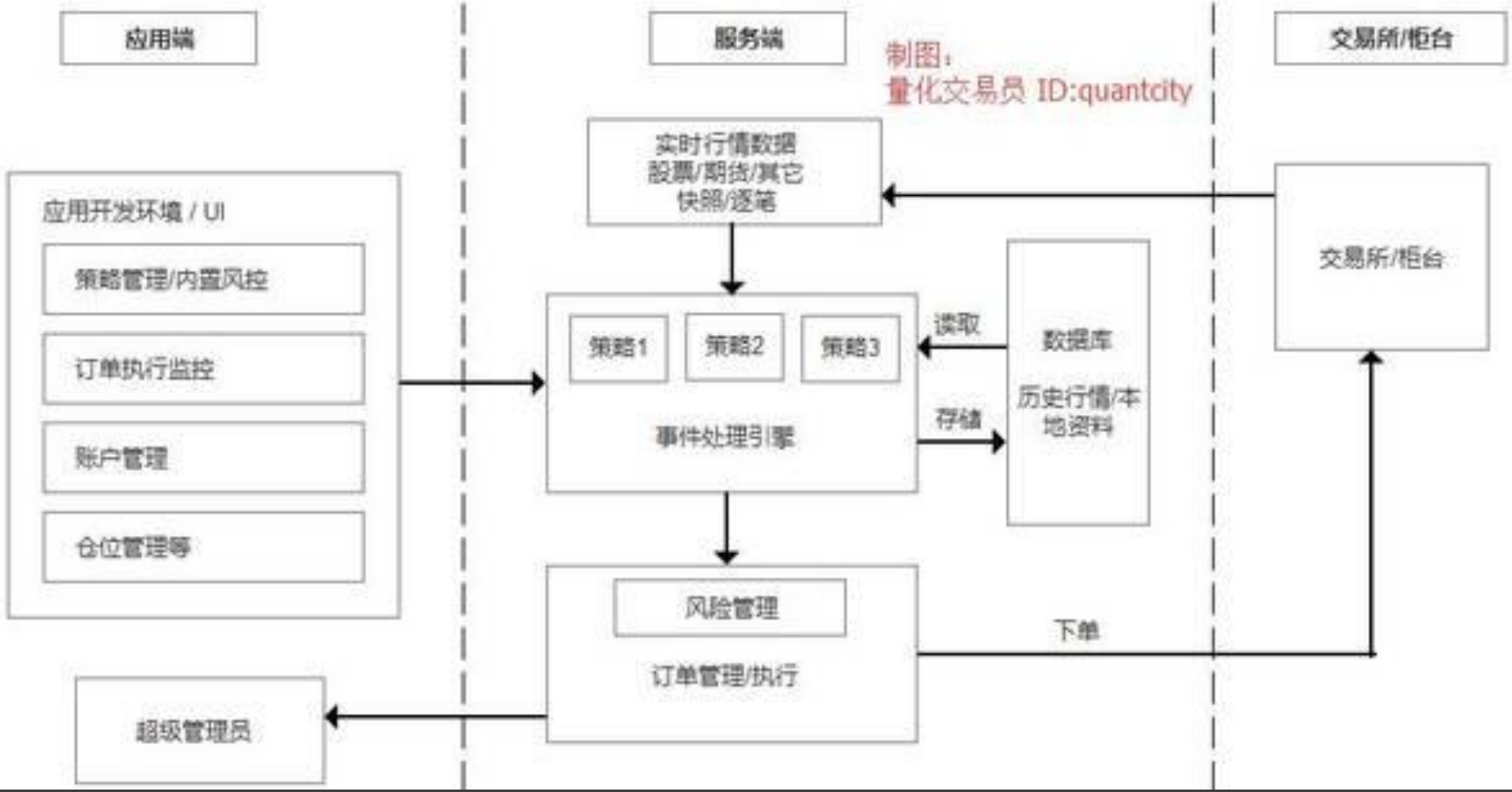
- 2003-2006, Netflix proposed Cinematch ranking algorithm to recommend movies for users – Collaborative Filtering [CF: 协同过滤]
- 2006, Jason Knapp proposed Real Time Bidding(RTB: 实时竞价)
- June 2011, Google proposed Ad Exchange
- Sept 2011, Taobao [淘宝] released 1st Ad Exchange - ADX Taobao Ad Exchange (TANX)
- Oct 2011, Google proposed LBS (Location based Serving) adv
- 2013, Twitter released real-time feeding (实时信息流)
- May 2018, Euro passed (GDPR:通用数据保护条例) to protect privacy

online banner advertising – as a new media for ADV



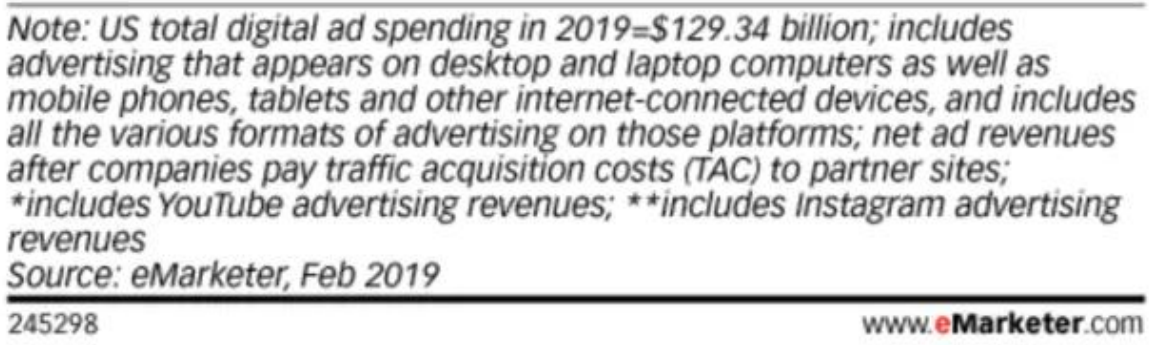
Google makes ADV professional

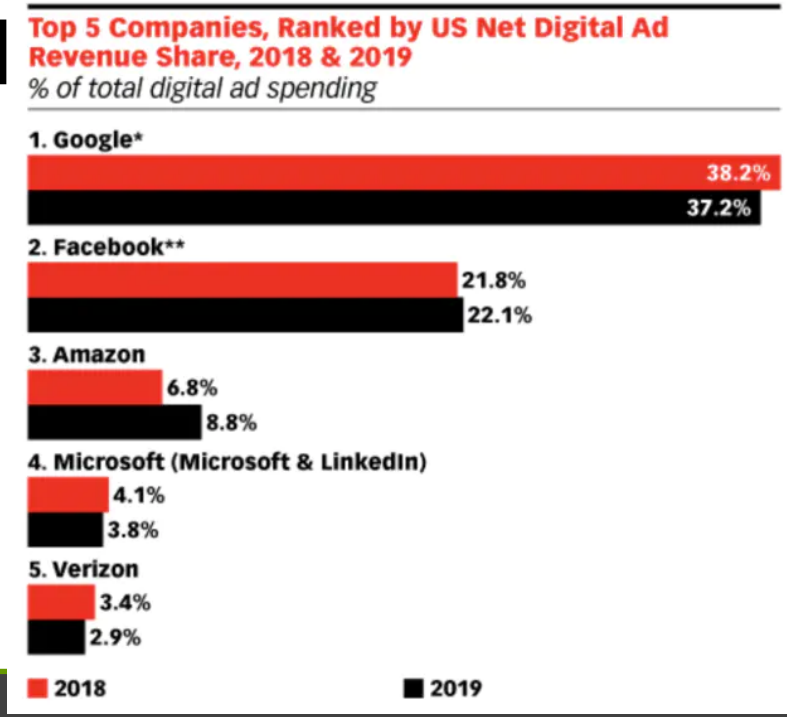
- Taking advantages from “Online Stock Bidding system” [在线股票竞价系统]



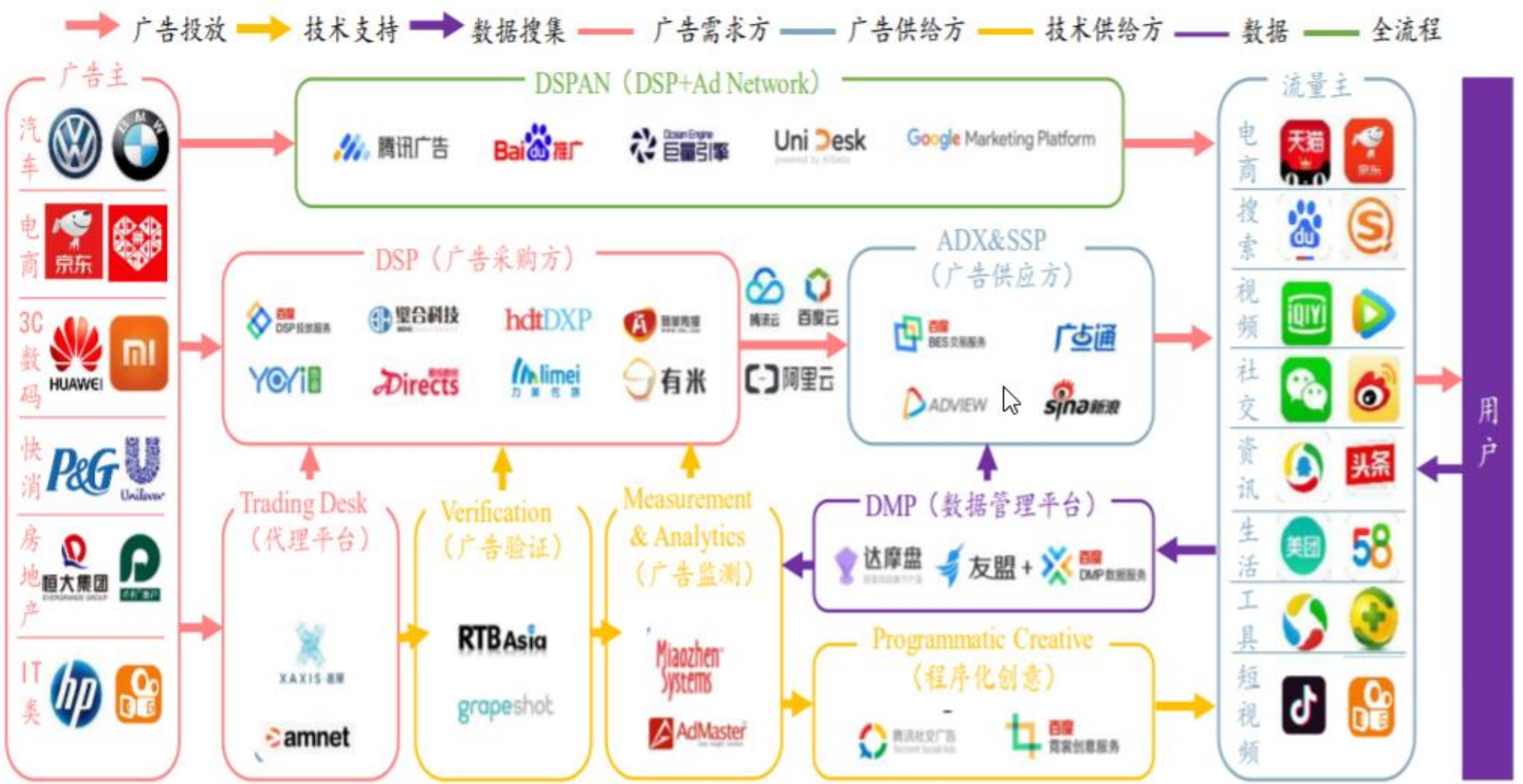
2 major types of recommendation scenarios

- **Recommendation used by itself [自卖自销]**
  - Like Amazon originally used item based CF (基于物品的协同过滤算法: ItemCF算法) in 1998
  - Like Netflix proposed Cinematch ranking algorithm to recommend movies for users – Collaborative Filtering [CF: 协同过滤] in 2003-2006
- **Recommendation used as Broker [广告平台]**
  - When a company maintains a lot of users, it's usual to use recommendation to earn money from the advertisements





图表7： 互联网广告产业链

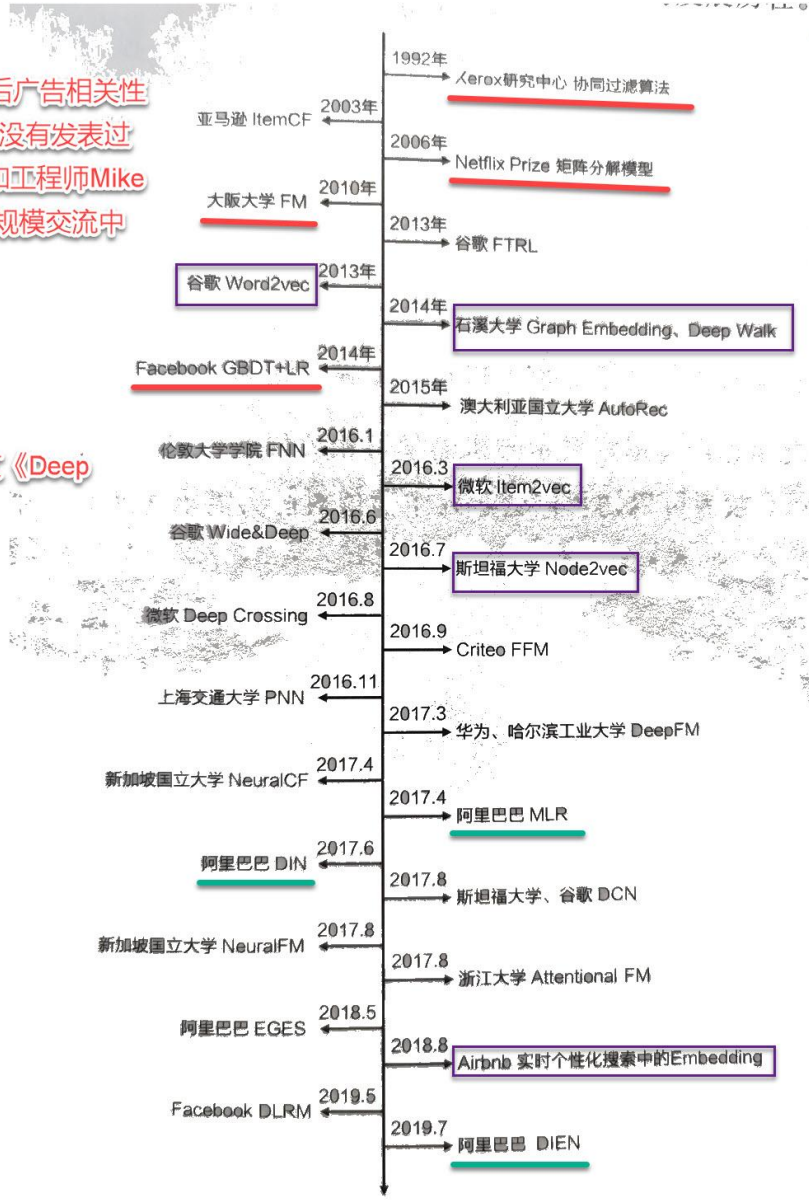


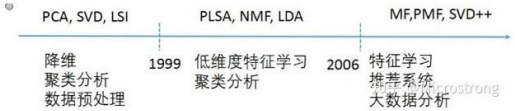
资料来源：中信建投研究发展部

So many algorithms

Google Rephil是Google AdSense背后广告相关性计算的头号秘密武器。但是这个系统没有发表过论文。只是其作者（博士Uri Lerner和工程师Mike Yar）在2002年在湾区举办的几次小规模交流中简要介绍过。

2016 - Youtube的深度推荐系统论文《Deep Neural Networks for YouTube Recommendations》





PCA

SVD - 奇异值分解的应用有很多，比如：用SVD解PCA、潜在语言索引也依赖于SVD算法。可以说，SVD是矩阵分解、降维、压缩、特征学习的一个基础的工具，所以SVD在机器学习领域相当的重要

LSA/LSI - SVD分解用在语义分析（称为LSA，latent semantic analysis）

pLSA/pLSI - 1999年Hofmann提出

NMF - Lee和Seung于1999年在自然杂志上提出的一种矩阵分解方法（Non-negative Matrix Factorization）

LDA - 2003年提出，NLP领域做Topic Model算法

- 从2007年至今，国内外很多团队都尝试过并行化pLSA和LDA - SparseLDA, AliasLDA, LightLDA, WarpLDA

- LDA一直没有大规模的火起来。这是因为不幸遇到了深度学习这股风：google在2013年提出了word2vec

[FM] Factorization Machines，由Steffen Rendle于2010提出

MF - Matrix Factorization

PMF（Probabilistic Matrix Factorization）

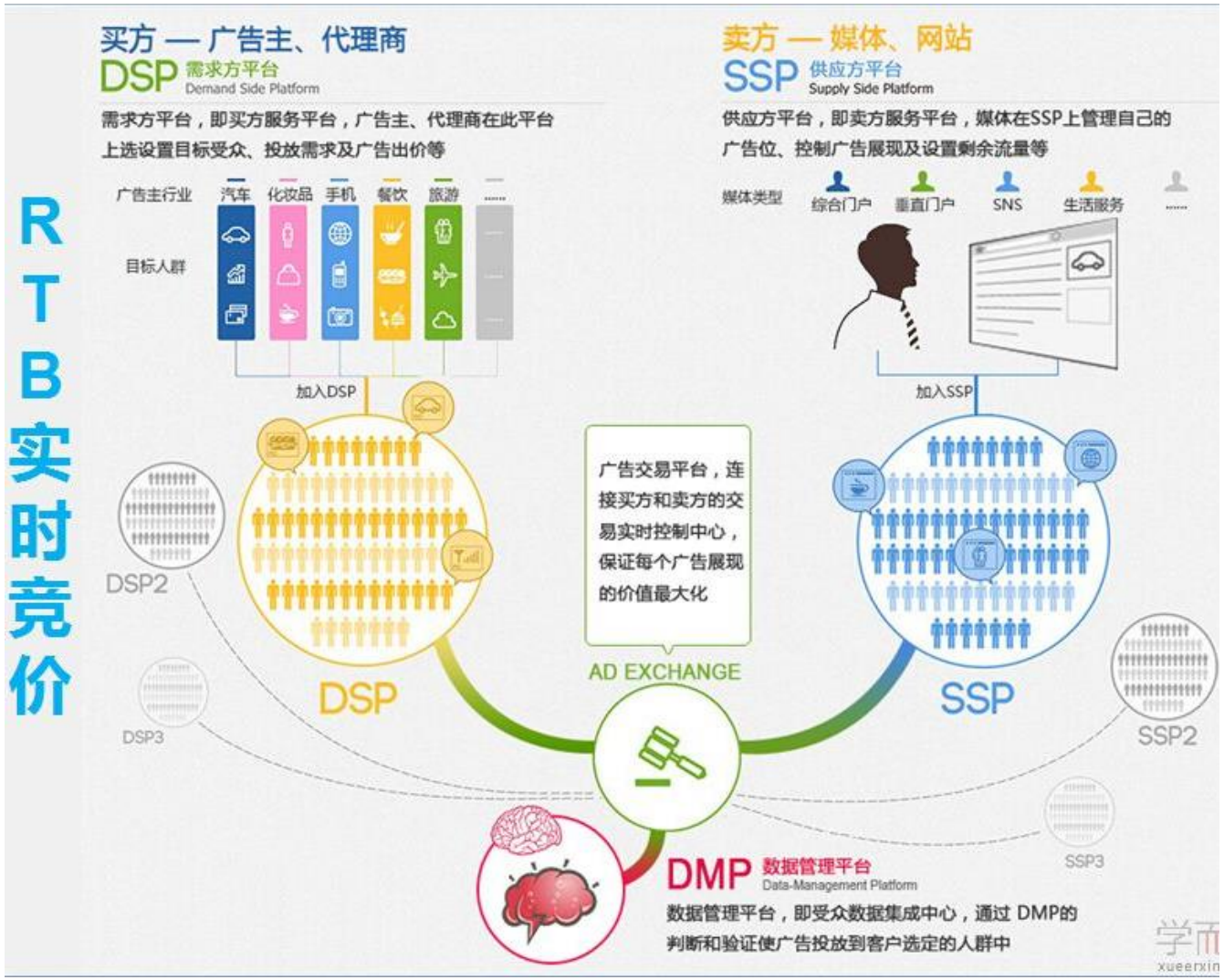
SVD++

Peacock - 2014, Yi WANG, ACM Trans on Intelligent Systems and Technology

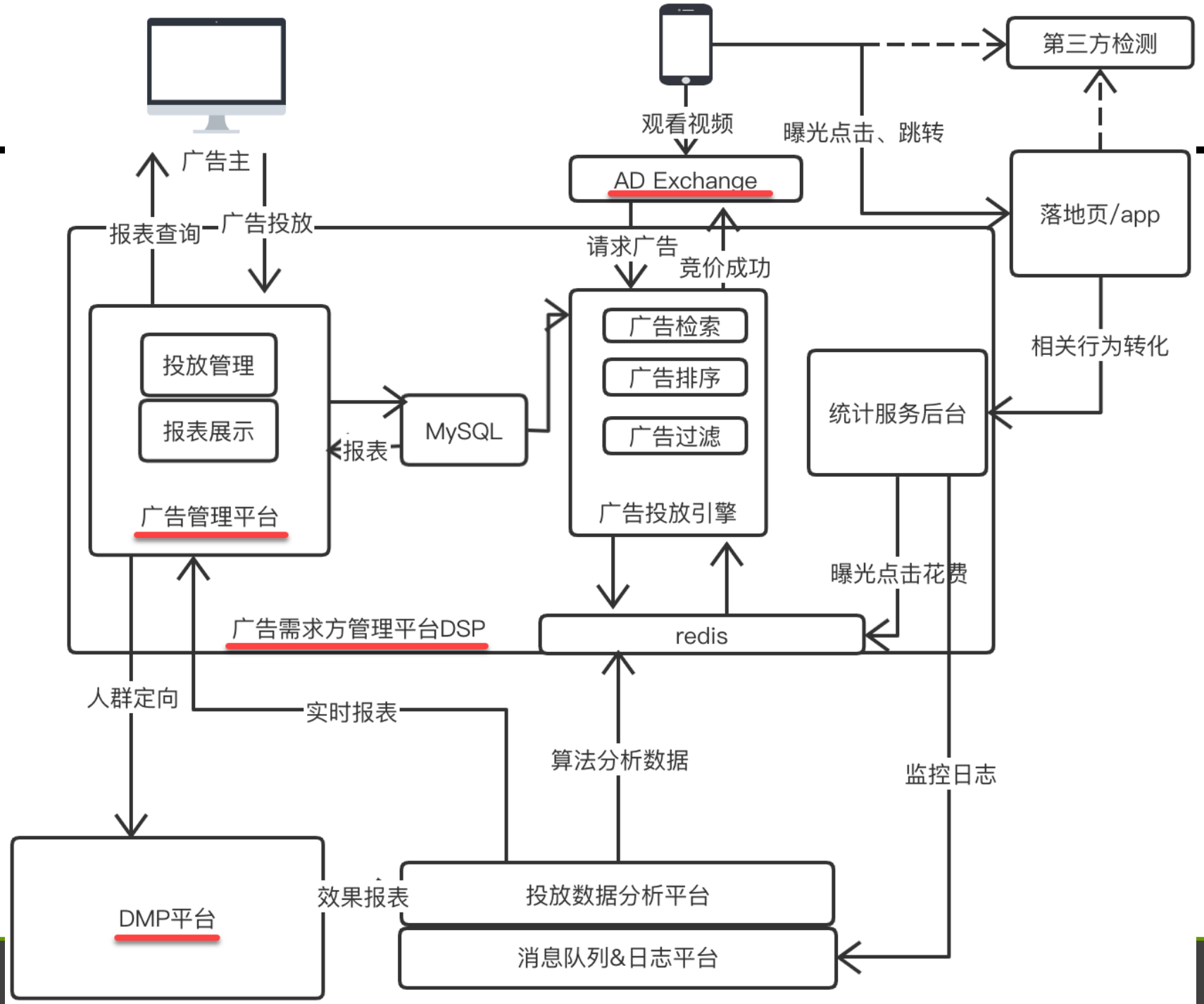
中国程序化广告技术生态图
China Programmatic Advertising Technology Landscape

RTBChina

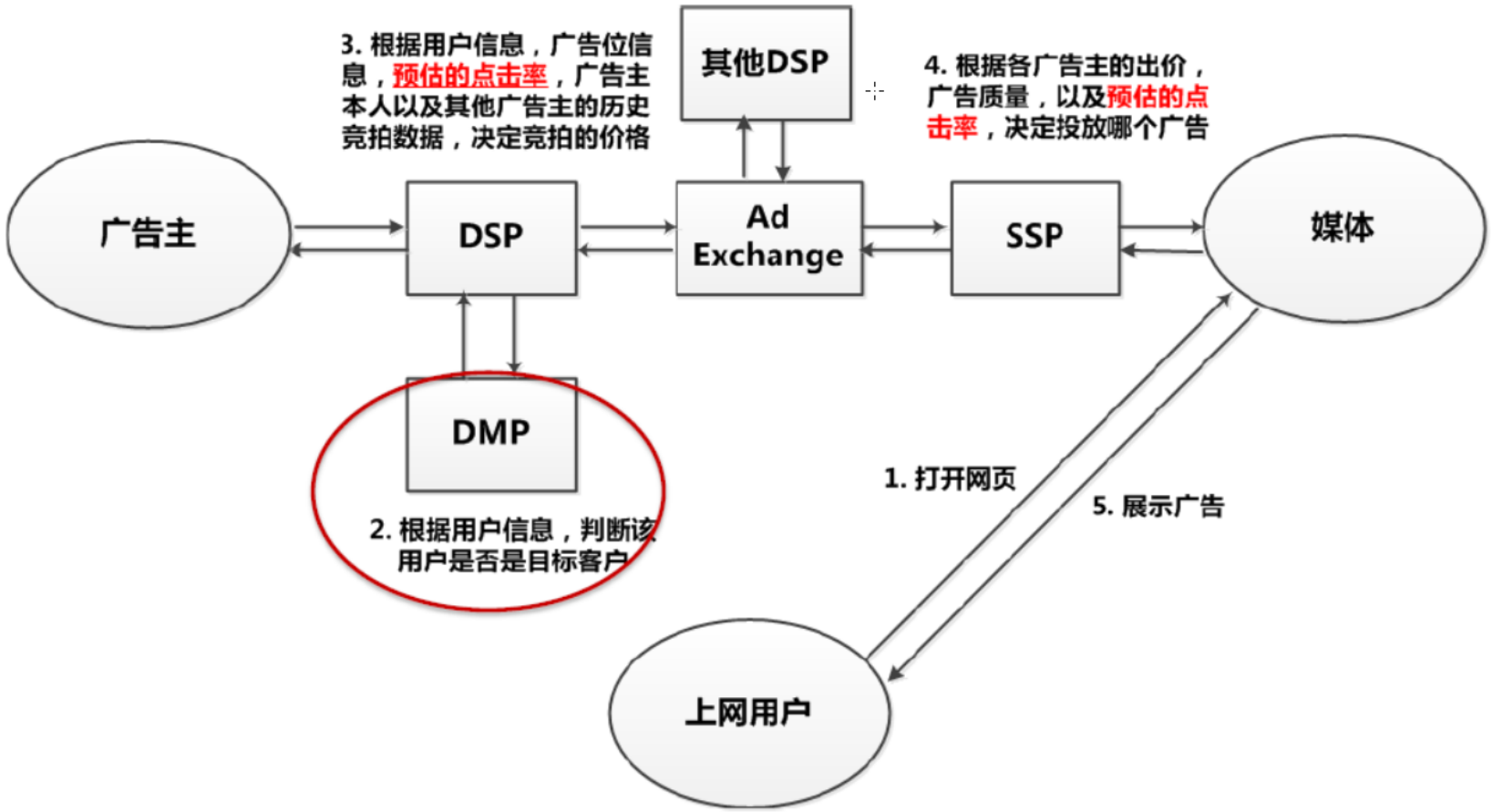
RTB is now the popular architecture/framework



- **DSP (Demand Side Platform) – for advertisers**
  - Describe your ads, buy keywords, select charging schemes …
- **SSP (Supply Side Platform) – for platform**
  - Manage the ad resources like positions, keywords …
- **DMP (Data Management Platform) – train and verify the effect of the ads**
- **Ad exchange – links DSP, SSP and DMP**
  - When a user input some keywords, this action will trigger ad exchange to
    - Trigger DMP and DSP to recall the candidate ads
    - Rank the candidates by models in DMP
    - Show the final ads to users based on SSP


第三方检测
观看视频
曝光点击、跳转
广告主
AD Exchange
落地页/app
报表查询
广告投放
请求广告
竞价成功
广告检索
广告排序
广告过滤
投放管理
报表展示
MySQL
报表
相关行为转化
统计服务后台
广告管理平台
广告投放引擎
曝光点击花费
广告需求方管理平台DSP
redis
人群定向
实时报表
算法分析数据
监控日志
DMP平台
效果报表
投放数据分析平台
消息队列&日志平台

一个互联网显示广告被拍卖的基本流程



- RTB (Real-time Bidding:实时竞价)
- DMP (Data Management Platform)
- DSP (Demand-Side Platform)
- Ad Exchange
- SSP (Supplier-Side Platform)

Bidding – ad exchange wants to balance …

[广告平台 – 如何赚钱是关键!]

- Google sells some keywords, positions (Header, Sidebar …)

- Advertisers propose their bids for some keywords, positions etc.

- When a user input the keywords, Google select candidate ads (Recall), and evaluate those candidates by using CPM (Cost per mille), CPC (Cost per click), eCPM (effective CPM), CPA (Cost per action) etc. to determine/rank the final ads which will be shown to the user

Platform balances its own Profit and Cost of the advisers

- **By recommend one user with one specific adv based on eCPM (effective CPM)**
  - CPM = Coverage*Depth*1000***CTR*BidPrice**
    - CTR – Click Through Rate: Statistical value – possibility of one user clicking one specific adv (一个具体的用户，点击某一个具体的广告的可能性)
    - BidPrice – also as CPC (Cost Per Click): 广告方的报价
  - When the platform recommends an adv to a specific user, eCPM is used

It seems Adidas is chosen for a user – $2 > $1

But, when considering “1000*”, we have

Nike : 1000*5%*1 = 50

Adidas: 1000*1%*2 = 20

Nike is chosen to show to the user

Bidding Price strategy

策略	原理	优点	缺点	采用者
GSP Generalized Second-Price	Ranking and pricing by: $\frac{ctr_{i+1} * bid_{i+1}}{ctr_i}$	市场稳定:locally envy-free equilibrium;	没有占优策略 Not truth telling 一户多开及串谋	Baidu, Google
VCG Vickrey-Clarke-Groves	if ranking by bid , pricing by $p_s = \sum_{t>s}(ctr_{t-1} - ctr_t)bid_t$	Truth telling	收入低于GSP	Facebook
GFP Generalized First Price	Ranking and pricing by: $ctr_i * bid_i$	容易理解	价格改动过于频繁导致收益不稳定	

GFP

广告位的平均点击数 (一般是 Per Million)

广义第一价格（GFP）

广告位置 Rank	广告主	出价(元)	点击数	广义第一价格（GFP）
1	A	10	200	10*200=2000
2	B	6	100	6*100=600
	C	3	~	

1.计费过程简单直接
2. 最高广告主出价为真实出价
3. 出价的稳定性很差，波动大

GSP

广义第二价格（GSP）

广告位置 Rank	广告主	出价(元)	点击数	广义第二价格（GSP）
1	A	10	200	6*200=1200
2	B	6	100	3*100=300
	C	3	~	

1.第二价格常为最高可承受价格，但也可被利用破坏竞争对手

2.广告主充分时，竞争格局稳定，在出价不变情况下，收费低于GFP.

3. 整个系统相对比较稳定，容易形成局部最优和整体稳定

Google 2002年开始采用该方式，之后的搜索引擎Baidu, Bing等都跟进采用。

VCG

VCG竞价机制（Vickrey-Clarke-Groves)

广告位置Rank	广告主	出价(元）	点击数	VCG竞价机制 (由于竞价成功者参与竞价，导致整体的收益减少量，即为竞价成功者需要的支付价格)
1	A	10	200	A不参加竞价的收益: B,C竞价成功，总收益为6*200+3*100=1500 A参加条件下，其他竞价收益：B获得Rank2机会，B带来收益为: 6*100=600 最后A需要支付： A不参加竞价整体收益—A参加条件下其他竞价者收益=1500-600=900
2	B	6	100	B不参加竞价的收益：C竞价Rank2成功，收益为3*100=300 B参加条件下，其他竞价者收益：0 B需要支付费用：300-0=300
	C	3	~	

1.VCG的计费思路：一个广告商获胜，在占有广告位的同时，导致某些广告主不能占有该广告位，那么获胜的广告主将支付该损失。

2.VCG价格并非来源于自己的出价，而是参考其它多个广告主的出价。（GSP只参考第二价格）

3. 在VCG下，广告主说真话是一种占优投放策略，整个系统容易达到稳定和纳什均衡。

VCG（Vickrey-Clarke-Groves）竞价机制

广告主为网民的一次点击支付他对其他广告主造成的效用损失。

广告位	平均每小时点击量
1	200
2	100

广告主	平均每次点击的收益
A	10
B	4
C	2

上面的竞价机制描述比较晦涩，举例以说明定价过程：

1）假设A不参加竞价，B和C的社会总效用为200*4 + 100*2 = 1000元

2）A如果参加竞价，B和C的社会总效用为100*4 + 0*2 = 400元

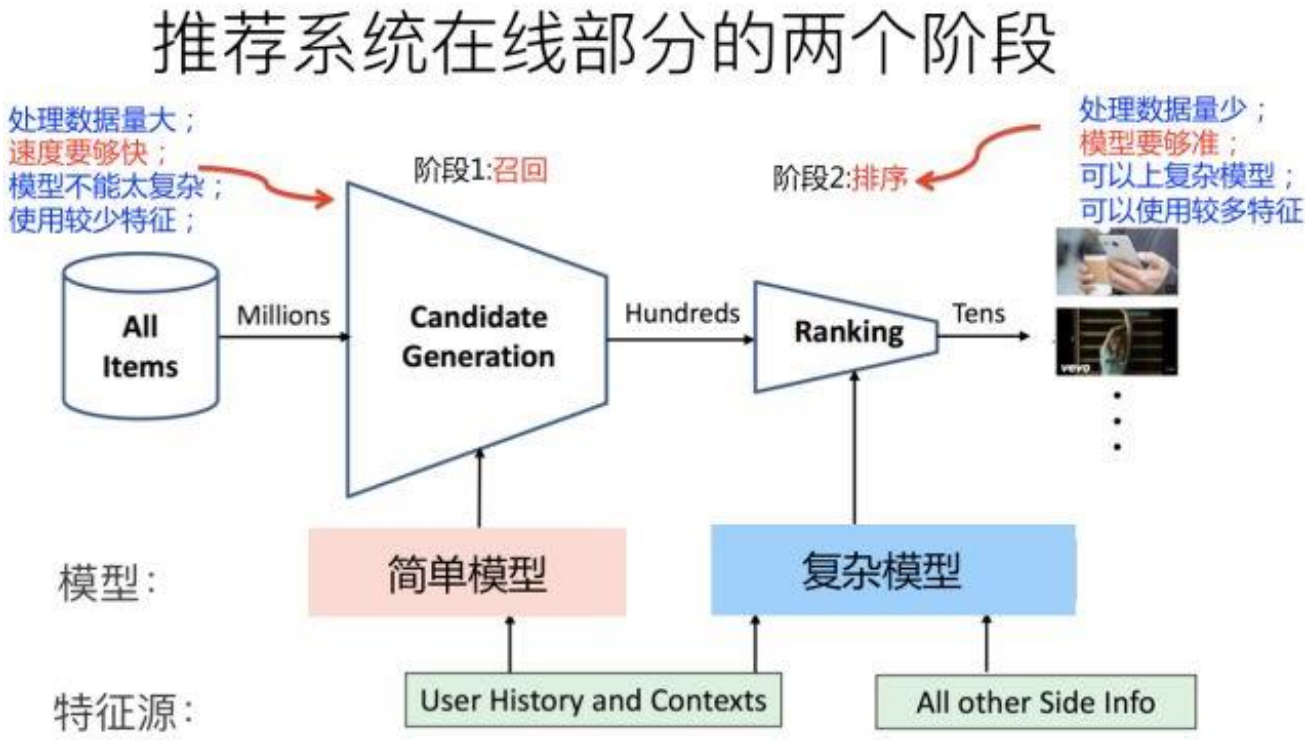
3）因为A参加了竞价，导致B和C的社会总效用损失了1000 – 400 = 600元

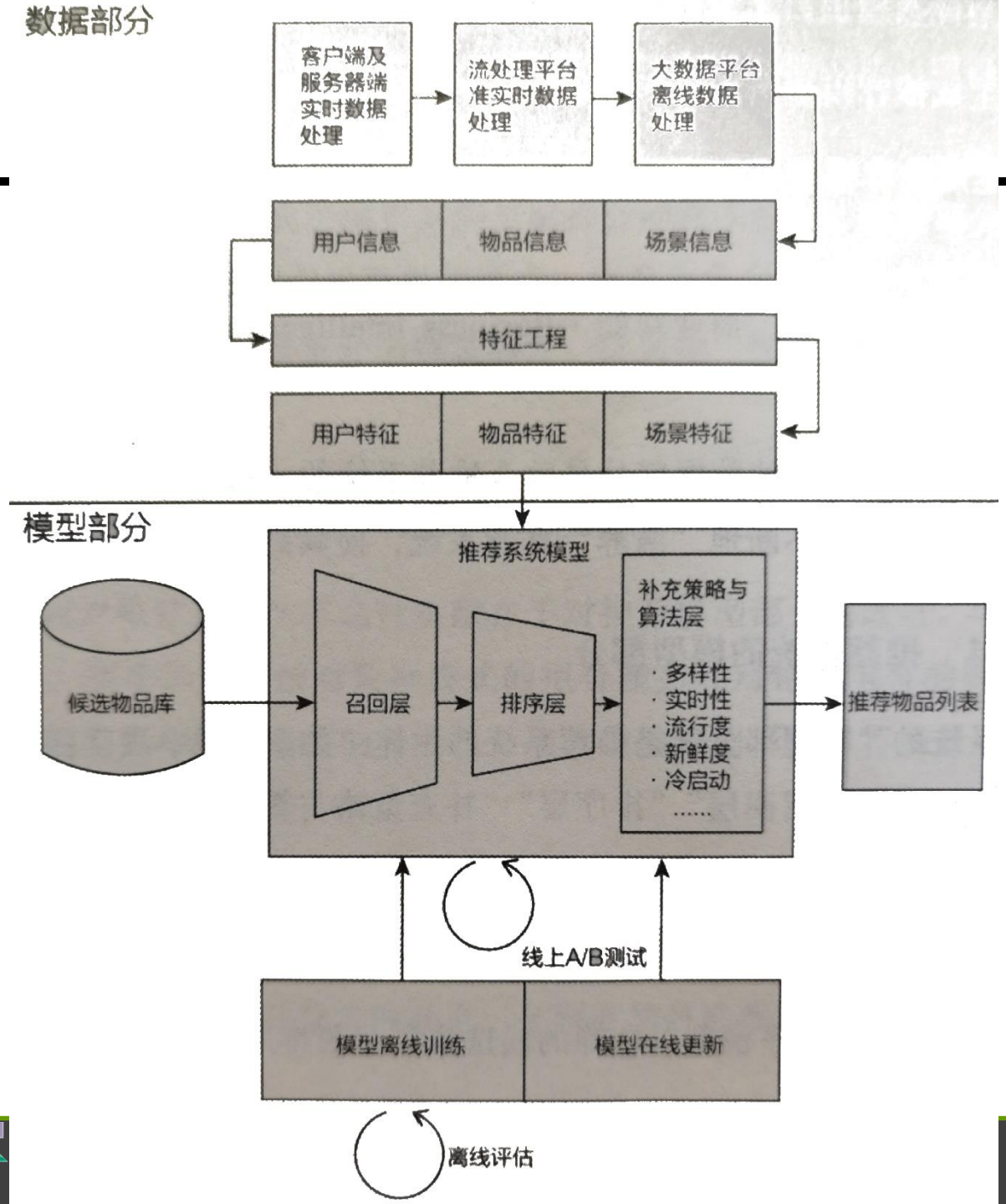
4）于是，A需要为每一次点击支付600 / 200 = 3元

但实际应用中，受损者的社会总效用损失难于计算，故VCG竞价机制几乎没有谁真的来使用。

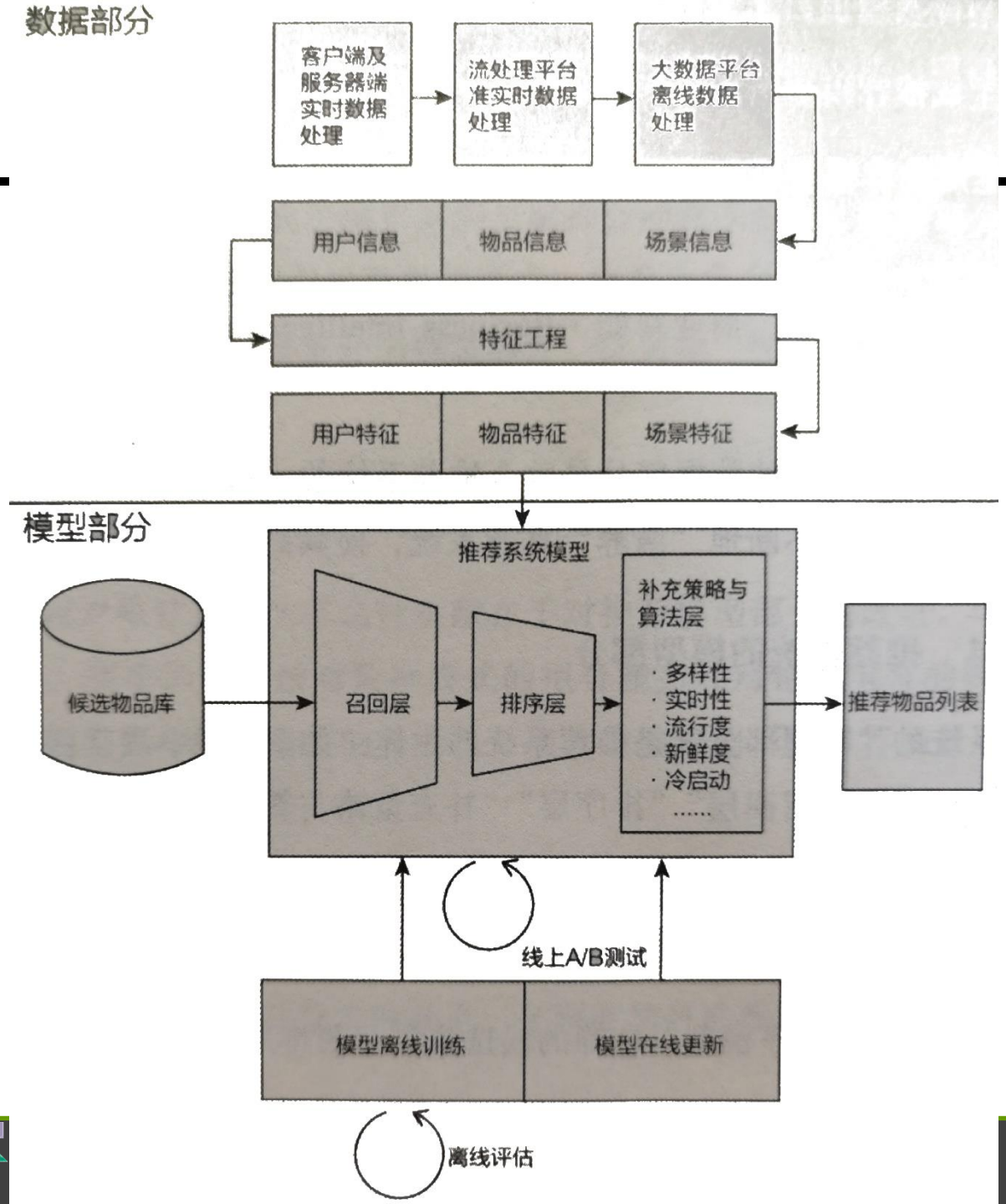
CTR? – Data flows

- **But with IT, especially with the Internet, Mobile, etc., the entities who could stick many users can also be Adv./Recommending ones to earn profit, such as Google – the greatest Search Engine company**
- **Triggered by banner advertising, CTR is the key now for Computing Adv.**
  - Click through rate – Click or not? Probability?





- **Recall [召回]**
  - Find the candidates
- **Ranking [排序]**
  - Merge and Top-K



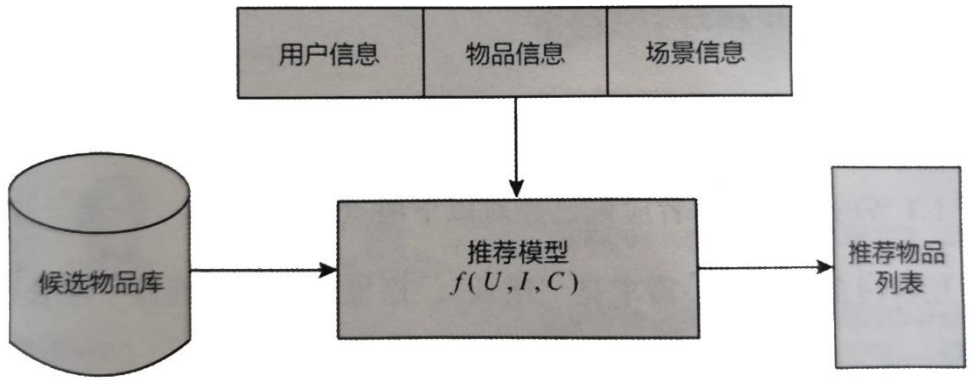


图 1-3　推荐系统逻辑框架

- **User information**
  - Personalities of the customers
  - History of the customers buying the items
- **Item Information**
  - Properties of the goods (Products and services)
- **Context Information**
  - Date, Location, Emergent event, etc.

一个预估 CTR 的最简单的模型

This is not the real way to use CTR

CTR is the key in Rank, because platform takes it as the value to filter the advs for one user

CTR could be “CORRECTLY” predicted

“精准广告”

- **About Recommendation and computing adv [关于计算广告]**
  - From Recommendation to Computing Advs
  - You should know how to earn money from ads in Google! – **RTB** (Real Time Bargain)
- **Algorithms**
  - CF, FM, LR, GBDT-LR (2014), Embedding, LDA (Latent Dirichlet Analysis) with EM, DL
    - Feature engineering is critical
- **Big Data way**

Google Rephil是Google AdSense背后广告相关性计算的头号秘密武器。但是这个系统没有发表过论文。只是其作者（博士Uri Lerner和工程师Mike Yar）在2002年在湾区举办的几次小规模交流中简要介绍过。

2016 - Youtube的深度推荐系统论文《Deep Neural Networks for YouTube Recommendations》

1992年 Xerox研究中心 协同过滤算法
2003年 亚马逊 ItemCF
2006年 Netflix Prize 矩阵分解模型
2010年 大阪大学 FM
2013年 谷歌 FTRL
2013年 谷歌 Word2vec
2014年 石溪大学 Graph Embedding、Deep Walk
2014年 Facebook GBDT+LR
2015年 澳大利亚国立大学 AutoRec
2016.1 伦敦大学学院 FNN
2016.3 微软 Item2vec
2016.6 谷歌 Wide&Deep
2016.7 斯坦福大学 Node2vec
2016.8 微软 Deep Crossing
2016.9 Criteo FFM
2016.11 上海交通大学 PNN
2017.3 华为、哈尔滨工业大学 DeepFM
2017.4 新加坡国立大学 NeuralCF
2017.4 阿里巴巴 MLR
2017.6 阿里巴巴 DIN
2017.8 斯坦福大学、谷歌 DCN
2017.8 新加坡国立大学 NeuralFM
2017.8 浙江大学 Attentional FM
2018.5 阿里巴巴 EGES
2018.8 Airbnb 实时个性化搜索中的Embedding
2019.5 Facebook DLRM
2019.7 阿里巴巴 DIEN

PCA

SVD - 奇异值分解的应用有很多，比如：用SVD解PCA、潜在语言索引也依赖于SVD算法。可以说，SVD是矩阵分解、降维、压缩、特征学习的一个基础的工具，所以SVD在机器学习领域相当的重要

LSA/LSI - SVD分解用在语义分析（称为LSA，latent semantic analysis）

pLSA/pLSI - 1999年Hofmann提出

NMF - Lee和Seung于1999年在自然杂志上提出的一种矩阵分解方法（Non-negative Matrix Factorization）

LDA - 2003年提出，NLP领域做Topic Model算法

- 从2007年至今，国内外很多团队都尝试过并行化pLSA和LDA - SparseLDA, AliasLDA, LightLDA, WarpLDA
- LDA一直没有大规模的火起来。这是因为不幸遇到了深度学习这股风：google在2013年提出了word2vec

[FM] Factorization Machines，由Steffen Rendle于2010提出

MF - Matrix Factorization

PMF（Probabilistic Matrix Factorization）

SVD++

Peacock - 2014, Yi WANG, ACM Trans on Intelligent Systems and Technology

Spotlight

Pytorch-based neural recommendation library

https://github.com/maciejkula/spotlight

```
conda install -c maciejkula -c pytorch spotlight=0.1.5
```

Also check out Microsoft RS repo

Collection of notebooks with many different techniques

https://github.com/Microsoft/Recommenders

So many algorithms
- The core is to match products for the given users

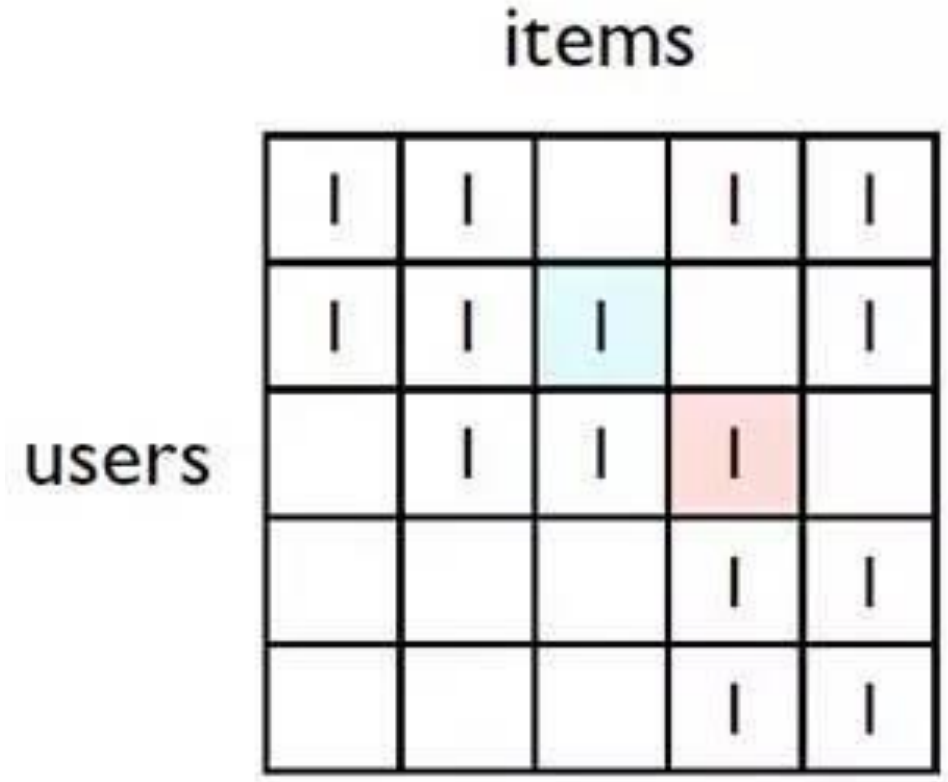
- It seems it's also OK to match users for the given products

Now these algorithms are usually used for Recall together with other methods. They are also good hints for feature engineering in later CTR prediction algorithms.

These algorithms are usually used for Rank, especially for CTR prediction. Feature engineering is critical – partly called embedding

Traditionally either directly by similar items (products) or by similar users [CF: Collaborative Filtering] (≤ 2003)

- The key is how to overcome the SPARSITY of the User-Item matrix



One-hot skill: binary way to represent data

- 3 items – A, B,C
- One-hot: A [1, 0, 0]; B [0, 1, 0], C [0, 0, 1]

- It's hard to compute the similarity of the users and items based on the Sparse U-I matrix
  - Similar users like similar items

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

$$Pearson(x, y) = \frac{\sum xy - \frac{\sum x \sum y}{N}}{\sqrt{(\sum x^2 - \frac{(\sum x)^2}{N})(\sum y^2 - \frac{(\sum y)^2}{N})}}$$

User-based nearest-neighbor collaborative filtering

- **The basic technique:**
  - Given an "active user" (Alice) and an item I not yet seen by Alice
  - The *goal is to estimate Alice's rating for this item*, e.g., by
    - find a set of users (peers) who liked the same items as Alice in the past **and** who have rated item I
    - use, e.g. the average of their ratings to predict, if Alice will like item I
    - do this for all items Alice has not seen and recommend the best-rated

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Measuring user similarity

- **A popular similarity measure in user-based CF: Pearson correlation**

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2}\ \sqrt{\sum_{p \in P}(r_{b,p} - \bar{r}_b)^2}}$$

a, b : users

$r_{a,p}$: rating of user a for item p

P : set of items, rated both by a and b

Possible similarity values between -1 and 1; $\overline{r_a}, \overline{r_b}$ = user's average ratings

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

sim = **0.85**
sim = 0.70
sim = -0.79

LSI/LSA – Latent Semantic Analysis(Index)

First for Topic modeling in NLP

TF*IDF + SVD

Index Words	Titles								
	T1	T2	T3	T4	T5	T6	T7	T8	T9
book			1	1					
dads						1			1
dummies		1						1	
estate							1		1
guide	1					1			
investing	1	1	1	1	1	1	1	1	1
market	1		1						
real							1		1
rich						2			1
stock	1		1					1	
value				1	1				

book	0.15	-0.27	0.04
dads	0.24	0.38	-0.09
dummies	0.13	-0.17	0.07
estate	0.18	0.19	0.45
guide	0.22	0.09	-0.46
investing	0.74	-0.21	0.21
market	0.18	-0.30	-0.28
real	0.18	0.19	0.45
rich	0.36	0.59	-0.34
stock	0.25	-0.42	-0.28
value	0.12	-0.14	0.23

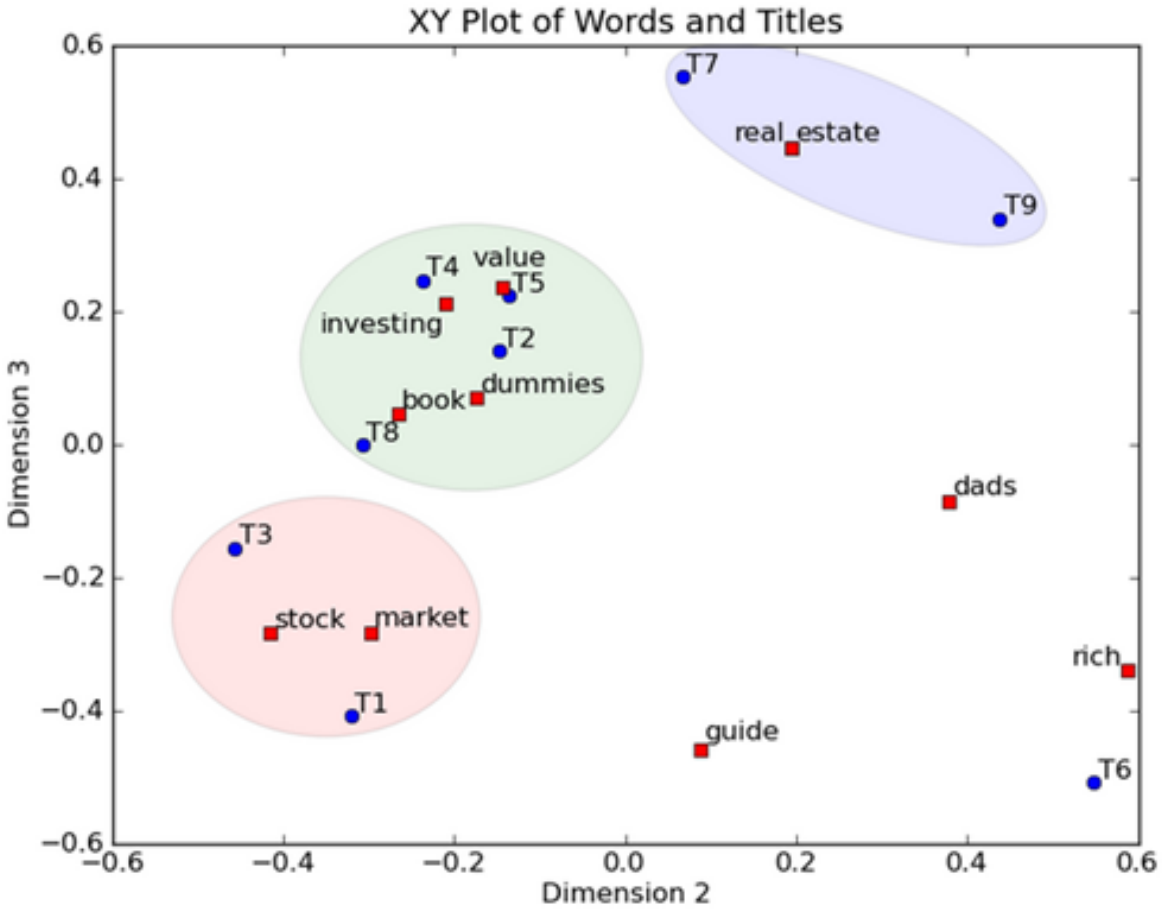
*

3.91	0	0
0	2.61	0
0	0	2.00

*

T1	T2	T3	T4	T5	T6	T7	T8	T9
0.35	0.22	0.34	0.26	0.22	0.49	0.28	0.29	0.44
-0.32	-0.15	-0.46	-0.24	-0.14	0.55	0.07	-0.31	0.44
-0.41	0.14	-0.16	0.25	0.22	-0.51	0.55	0.00	0.34

左奇异向量表示词的一些特性，右奇异向量表示文档的一些特性，中间的奇异值矩阵表示左奇异向量的一行与右奇异向量的一列的重要程序，数字越大越重要。
继续看这个矩阵还可以发现一些有意思的东西，首先，左奇异向量的第一列表示每一个词的出现频繁程度，虽然不是线性的，但是可以认为是一个大概的描述，比如book是0.15对应文档中出现的2次，investing是0.74对应了文档中出现了9次，rich是0.36对应文档中出现了3次；
其次，右奇异向量中一的第一行表示每一篇文档中的出现词的个数的近似，比如说，T6是0.49，出现了5个词，T2是0.22，出现了2个词。
然后我们反过头来看，我们可以将左奇异向量和右奇异向量都取后2维（之前是3维的矩阵），投影到一个平面上，可以得到：



- **在图上，每一个红色的点，都表示一个词，每一个蓝色的点，都表示一篇文档，这样我们可以对这些词和文档进行聚类**
  - 比如说stock 和 market可以放在一类，因为他们老是出现在一起
  - real和estate可以放在一类
  - dads，guide这种词就看起来有点孤立了，我们就不对他们进行合并了。
- **按这样聚类出现的效果，可以提取文档集合中的近义词，这样当用户检索文档的时候，是用语义级别（近义词集合）去检索了，而不是之前的词的级别**
- **这样一减少我们的检索、存储量，因为这样压缩的文档集合和PCA是异曲同工的，二可以提高我们的用户体验，用户输入一个词，我们可以在这个词的近义词的集合中去找，这是传统的索引无法做到的**

pLSA – probabilistic LSA (EM)

- Like GMM – Given data records X, $P(Z_k = 1|X)$ and $P(\mu_k, \sigma_k|Z_k = 1, X)$ are the parameters we want to compute
- pLSA – Given documents D, $P(Z_k = 1|D)$ and $P(w_j|Z_k = 1, D)$ are the parameters we want to compute

$L(P(z_k|d_i), P(w_j|z_k)) = P(D, W)$ 其中 D 是所有文档，W 表示单词，需要做的就是找一组 $P(z_k|d_i), P(w_j|z_k)$ 使得 $P(D, W)$ 的概率最大.....（4）

$$L(P(z_k|d_i), P(w_j|z_k)) = \prod_{i=1}^{N}\prod_{j=1}^{M} P(d_i, w_j) = \prod_{i}\prod_{j} P(d_i, w_j)^{n(d_i, w_j)}$$ 其中

$n(d_i, w_j)$ 表示的是 w_j 在 d_i 中出现的次数.....（5）

$$L(P(z_k|d_i), P(w_j|z_k)) = \sum_i \sum_j n(d_i, w_j) log P(d_i, w_j)$$ 取对数.....（6）

$$L(P(z_k|d_i), P(w_j|z_k)) = \sum_i \sum_j n(d_i, w_j) log \left[\sum_{k=1}^{K} P(w_j|z_k) P(z_k|d_i) P(d_i)\right]$$

由公式（3）代入可得.....（7）

通过上面的推导，我们得到了关于参数 $P(z_k|d_i), P(w_j|z_k)$ 的目标函数：

$$L(P(z_k|d_i), P(w_j|z_k)) = \sum_i \sum_j n(d_i, w_j) log \left[\sum_{k=1}^{K} P(w_j|z_k) P(z_k|d_i) P(d_i)\right]$$

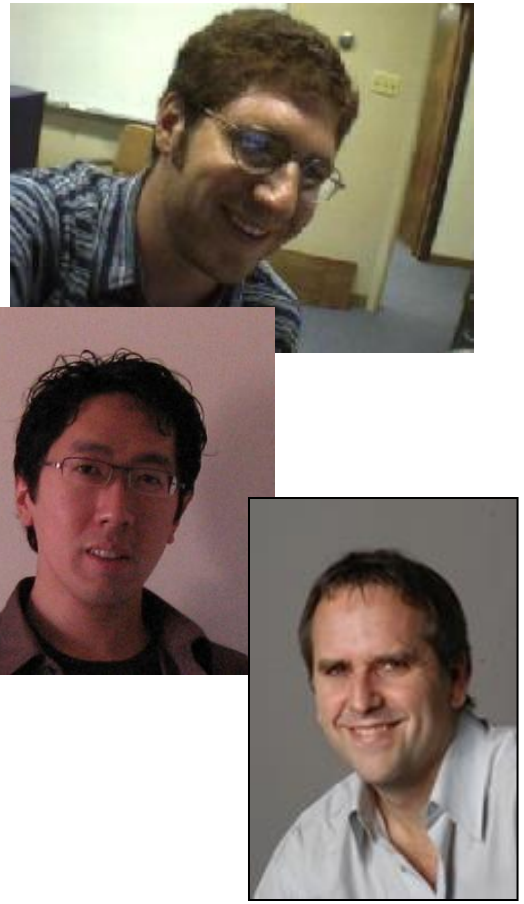
LDA - Latent Dirichlet Allocation

Latent Dirichlet Allocation

David M. Blei
Computer Science Division
University of California
Berkeley, CA 94720, USA

Andrew Y. Ng
Computer Science Department
Stanford University
Stanford, CA 94305, USA

Michael I. Jordan
Computer Science Division and Department of Statistics
University of California
Berkeley, CA 94720, USA



Peacock

Peacock: Learning Long-Tail Topic Features for Industrial Applications

YI WANG, Tencent
XUEMIN ZHAO, Tencent
ZHENLONG SUN, Tencent
HAO YAN, Tencent
LIFENG WANG, Tencent
ZHIHUI JIN, Tencent
LIUBIN WANG, Tencent
YANG GAO, School of Computer Science and Technology, Soochow University
CHING LAW, Tencent
JIA ZENG, School of Computer Science and Technology, Soochow University & Huawei Noah's Ark Lab

Latent Dirichlet allocation (LDA) is a popular topic modeling technique in academia but less so in industry, especially in large-scale applications involving search engine and online advertising systems. A main underlying reason is that the topic models used have been too small in scale to be useful; for example, some of the largest LDA models reported in literature have up to 10^3 topics, which cover difficultly the long-tail semantic word sets. In this paper, we show that the number of topics is a key factor that can significantly boost the utility of topic-modeling systems. In particular, we show that a "big" LDA model with at least 10^5 topics inferred from 10^9 search queries can achieve a significant improvement on industrial search engine and online advertising systems, both of which serving hundreds of millions of users. We develop a novel distributed system called Peacock to learn big LDA models from big data. The main features of Peacock include hierarchical distributed architecture, real-time prediction and topic de-duplication. We empirically demonstrate that the Peacock system is capable of providing significant benefits via highly scalable LDA topic models for several industrial applications.

CTR algorithms

LR 时期：

- Logistic Regression & SVM
- Degree-2 Polynomial Mapping（Poly2）2010 年[1]

FM 时期：

- FM （Factorization Machine）2010 年[2]
- FFM（Field-aware Factorization Machine）2016 年 recsys[3]

Deep 时期：

- FNN（Factorization machine supported Neural Network）2016 年 1 月[4]
- WDL（Wide&Deep）2016 年 6 月[5]
- PNN（Product-based Neural Network）2016 年 11 月[6]
- DeepFM（Factorization-Machine based Neural Network）2017 年[7]

FM (Factorization Machine) – 2010, Steffen Rendle

- **2010年，日本大阪大学(Osaka University)的 Steffen Rendle 在矩阵分解(MF)、SVD++[2]、PITF[3]、FPMC[4] 等基础之上，归纳出针对高维稀疏数据的因子机(Factorization Machine, FM)模型[11]。因子机模型可以将上述模型全部纳入一个统一的框架进行分析。并且，Steffen Rendle 实现了一个单机多线程版本的 libFM。**
  - 在随后的 KDD Cup 2012，track2 广告点击率预估(pCTR)中，国立台湾大学[4] 和 Opera Solutions[5] 两只队伍都采用了 FM，并获得大赛的冠亚军而使得 FM 名声大噪。
- **随后，台湾大学的 Yuchin Juan 等人在总结自己在两次比赛中的经验以及 Opera Solutions 队伍中使用的 FM 模型的总结，提出了一般化的 FFM 模型[6]，并实现了单机多线程版的 libFFM，并做了深入的试验研究。事实上，Opera Solutions 在比赛中用的 FM 就是FFM。**

https://www.csuldw.com/2019/02/08/2019-02-08-fm-algorithm-theory/

LR → SVM → FM

$$y = w_0 + \sum_{i=1}^{n} w_i x_i \tag{1}$$

其中n表示样本的特征数量，w_0为偏置，x_i表示第i个特征的值，后面一项为特征的Linear combination。FM增加了特征交叉组合部分，表达式如下：

$$y = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} x_i x_j \tag{2}$$

其中w_0、w_i、w_{ij}是模型参数。从公式来看，模型前半部分就是普通的LR线性组合[No Problem]，后半部分的交叉项：特征组合。首先，单从模型表达能力上来看，FM是要强于LR的，至少它不会比LR弱，当交叉项参数w_{ij}全为0的时候，整个模型就退化为普通的LR模型。对于有n个特征的模型，特征组合的参数数量共有$1+2+3+\cdots+n-1=\frac{n(n-1)}{2}$个，并且任意两个参数之间是独立的。所以说特征数量比较多的时候，特征组合之后，维度自然而然就高了。

代数知识：

> 任意一个实对称矩阵（正定矩阵）W都存在一个矩阵V，使得$W = V.V^T$成立。

类似地，所有二次项参数ω_{ij}可以组成一个对称阵W（为了方便说明FM的由来，对角元素可以设置为正实数），那么这个矩阵就可以分解为$W = V^T V$，V的第j列(v_j)便是第j维特征(x_j)的隐向量。

- Given labeled pairs (X_i, y_i) where X_i is a vector with M attributes, y_i is the class/label
- FM's hypothesis $h_w = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} x_i x_j$
- Trained by different errors like MSE, Logit loss etc.

$$\hat{y}(X) = \omega_0 + \sum_{i=1}^{n} \omega_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} < v_i, v_j > x_i x_j \tag{3}$$

需要估计的参数有$\omega_0 \in R$，$\omega_i \in R$，$V \in R$，$< \cdot, \cdot >$是长度为k的两个向量的点乘，公式如下：

$$< v_i, v_j >= \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f} \tag{4}$$

上面的公式(3)，(4)中，

- ω_0为全局偏置；
- ω_i是模型第i个变量的权重；
- $\omega_{ij} =< v_i, v_j >$特征$i$和$j$的交叉权重；
- v_i是第i维特征的隐向量；
- $< \cdot, \cdot >$代表向量点积；
- $k(k << n)$为隐向量的长度，包含 k 个描述特征的因子。

根据公式(2)，二次项的参数数量减少为 kn个，远少于多项式模型的参数数量。另外，参数因子化使得 $x_h x_i$ 的参数和 $x_i x_j$ 的参数不再是相互独立的，因此我们可以在样本稀疏的情况下相对合理地估计FM的二次项参数。具体来说，$x_h x_i$ 和 $x_i x_j$的系数分别为 $< v_h, v_i >$ 和 $< v_i, v_j >$，它们之间有共同项 v_i。也就是说，所有包含“ x_i 的非零组合特征”（存在某个 $j \neq i$，使得 $x_i x_j \neq 0$）的样本都可以用来学习隐向量v_i，这很大程度上避免了数据稀疏性造成的影响。而在多项式模型中,w_{hi} 和 w_{ij}是相互独立的。

公式(3)的推导

$$ab + ac + bc = \frac{1}{2}\left[(a+b+c)^2 - (a^2+b^2+c^2)\right] \tag{5}$$

$$\begin{aligned}
\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} < v_i, v_j > x_i x_j &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} < v_i, v_j > x_i x_j - \frac{1}{2}\sum_{i=1}^{n} < v_i, v_i > x_i x_i \\
&= \frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{f=1}^{k} v_{i,f} v_{j,f} x_i x_j - \sum_{i=1}^{n}\sum_{f=1}^{k} v_{i,f} v_{i,f} x_i x_i\right) \\
&= \frac{1}{2}\sum_{f=1}^{k}\left[\left(\sum_{i=1}^{n} v_{i,f} x_i\right)\cdot\left(\sum_{j=1}^{n} v_{j,f} x_j\right) - \sum_{i=1}^{n} v_{i,f}^2 x_i^2\right] \\
&= \frac{1}{2}\sum_{f=1}^{k}\left[\left(\sum_{i=1}^{n} v_{i,f} x_i\right)^2 - \sum_{i=1}^{n} v_{i,f}^2 x_i^2\right]
\end{aligned} \tag{6}$$

解释:

- $v_{i,f}$ 是一个具体的值;
- 第1个等号: 对称矩阵 W 对角线上半部分;
- 第2个等号: 把向量内积 v_i,v_j 展开成累加和的形式;
- 第3个等号: 提出公共部分;
- 第4个等号: i 和 j 相当于是一样的, 表示成平方过程。

ALGORITHM 1: Stochastic Gradient Descent (SGD)

Input: Training data S, regularization parameters λ, learning rate η, initialization σ
Output: Model parameters $\Theta = (w_0, \mathbf{w}, \mathbf{V})$

$w_0 \leftarrow 0$; $\mathbf{w} \leftarrow (0, \ldots, 0)$; $\mathbf{V} \sim \mathcal{N}(0, \sigma)$;
repeat
  for $(\mathbf{x}, y) \in S$ **do**
    $w_0 \leftarrow w_0 - \eta \left(\frac{\partial}{\partial w_0} l(\hat{y}(\mathbf{x}|\Theta), y) + 2\,\lambda^0\, w_0 \right)$;
    for $i \in \{1, \ldots, p\} \wedge x_i \neq 0$ **do**
      $w_i \leftarrow w_i - \eta \left(\frac{\partial}{\partial w_i} l(\hat{y}(\mathbf{x}|\Theta), y) + 2\,\lambda^w_{\pi(i)}\, w_i \right)$;
      for $f \in \{1, \ldots, k\}$ **do**
        $v_{i,f} \leftarrow v_{i,f} - \eta \left(\frac{\partial}{\partial v_{i,f}} l(\hat{y}(\mathbf{x}|\Theta), y) + 2\,\lambda^v_{f,\pi(i)}\, v_{i,f} \right)$;
      end
    end
  end
until *stopping criterion is met*;

综上，最终模型各参数的梯度表达式如下：

$$\frac{\partial y}{\partial \theta} = \begin{cases} 1, & \text{if } \theta \text{ is } w_0; \\ x_i, & \text{if } \theta \text{ is } w_i; \\ x_i \sum_{j=1}^{n} v_{jf} x_j - x_i^2 v_{if}, & \text{if } \theta \text{ is } v_{if}. \end{cases}$$

2.2 预测

FM算法可以应用在多种的预测任务中，包括：

- **Regression**： $\hat{y}(x)$ 可以直接用作预测，并且最小平方误差来优化。
- **Binary classification**： $\hat{y}(x)$ 作为目标函数并且使用hinge loss或者logit loss来优化。
- **Ranking**：向量 x 通过 $\hat{y}(x)$ 的分数排序，并且通过pairwise的分类损失来优化成对的样本 $(x^{(a)}, x^{(b)})$

对以上的任务中，正则化项参数一般加入目标函数中以防止过拟合。

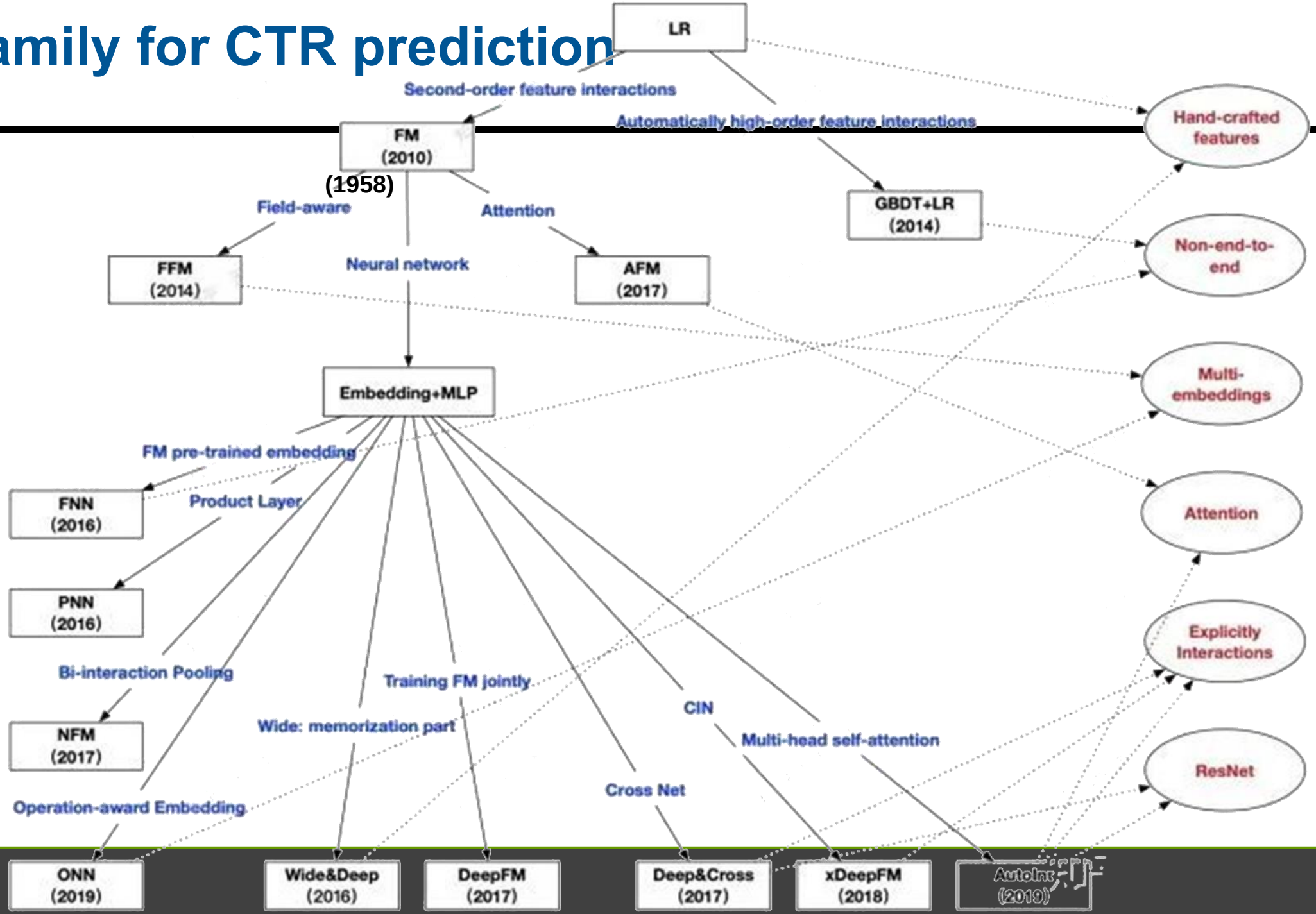
2.3 参数学习

从上面的描述可以知道FM可以在线性的时间内进行预测。因此模型的参数可以通过梯度下降的方法（例如随机梯度下降）来学习，对于各种的损失函数。FM模型的梯度是：

$$\frac{\partial}{\partial \theta}\hat{y}(x) = \begin{cases} 1, & if\ \theta\ is\ w_0 \\ x_i, & if\ \theta\ is\ w_i \\ x_i \sum_{j=1}^{n} v_{j,f} x_j - v_{i,f} x_i^2, & if\ \theta\ is\ v_{i,f} \end{cases} \tag{5}$$

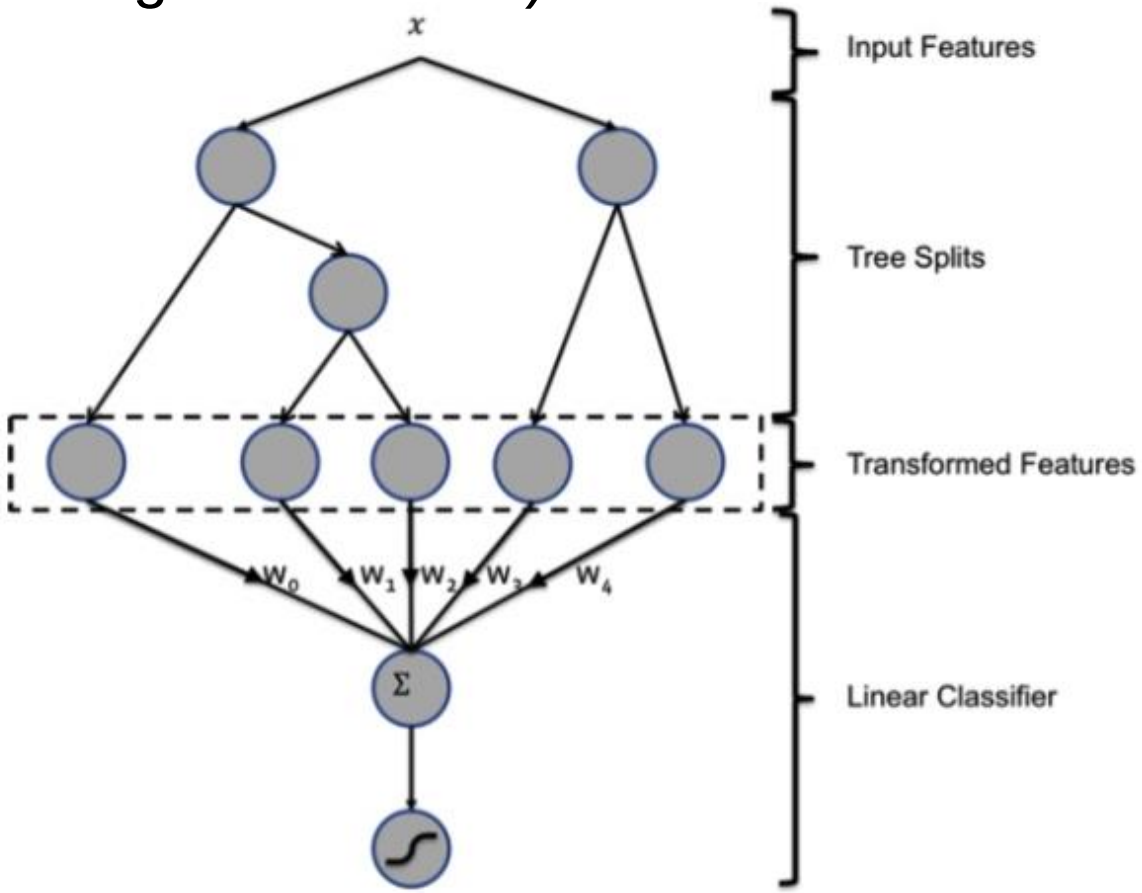
由于 $\sum_{j=1}^{n} v_{j,f} x_j$ 只与 f 有关，与 i 是独立的，可以提前计算出来，并且每次梯度更新可以在常数时间复杂度内完成，因此FM参数训练的复杂度也是 $O(kn)$ 。综上可知，FM可以在线性时间训练和预测，是一种非常高效的模型。

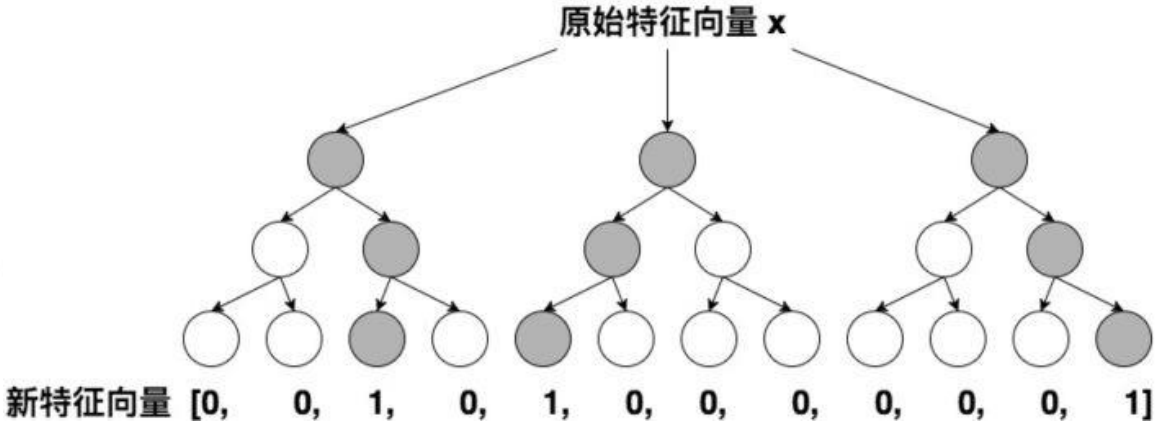
FM' s family for CTR prediction



NN till DL for CTR prediction [≥ 2013]

- **GBDT-LR (2014, Facebook)**
  - Gradient Boosting Decision Tree – many CARTs (Classification And Regression Tree)

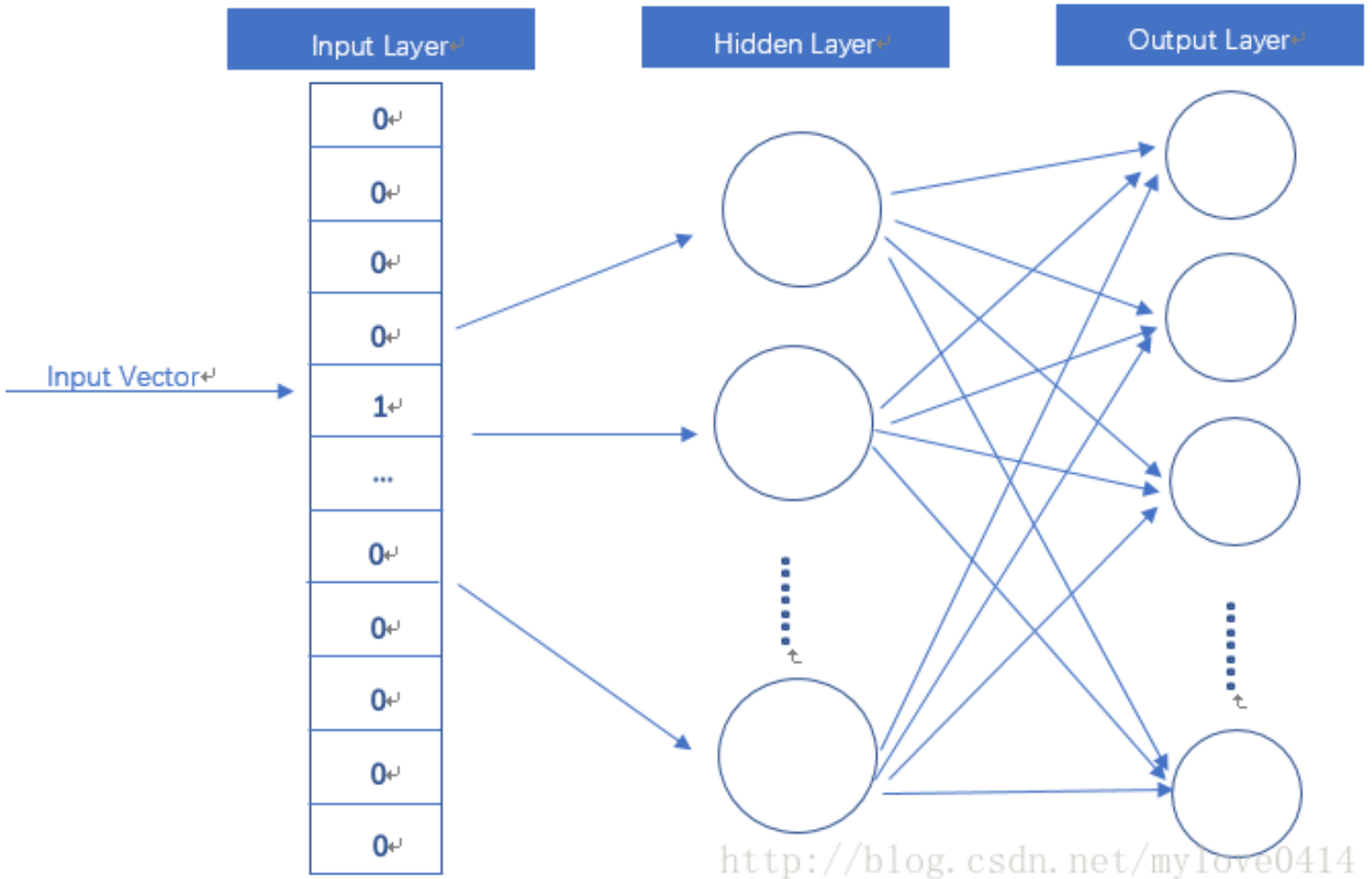




Feature Engineering

- **GBDT (2014) is used as feature engineering method in GBDT-LR**
  - Still designed manually
- **We've learned in NN part that NN has many interesting models to automatically learn features, called Embedding – word2vec,BERT, item2vec, Graph Embedding (DeepWalk)**
  - Embedding就是用一个低维的向量表示一个物体，可以是一个词，或是一个商品，或是一个电影等等。
    - 这个embedding向量的性质是能使距离相近的向量对应的物体有相近的含义，比如 Embedding(复仇者联盟)和Embedding(钢铁侠)之间的距离就会很接近，但Embedding(复仇者联盟)和Embedding(乱世佳人)的距离就会远一些。
  - 除此之外Embedding甚至还具有数学运算的关系，比如Embedding（马德里）-Embedding（西班牙）+Embedding(法国)≈Embedding(巴黎)

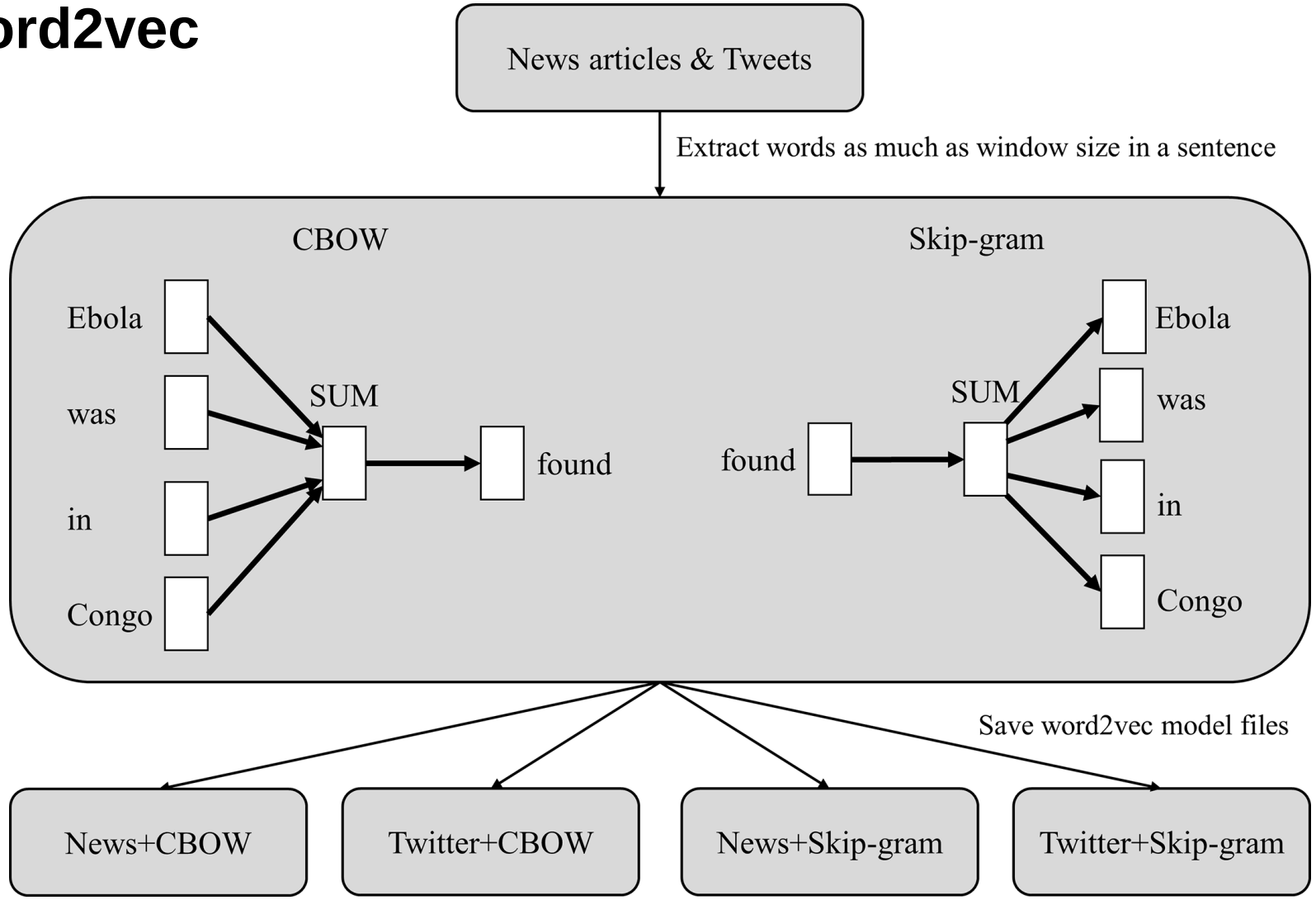
word2vec模型其实就是简单化的神经网络



输入是**One-Hot Vector**，Hidden Layer没有激活函数，也就是线性的单元。
Output Layer维度跟Input Layer的维度一样，用的是Softmax回归。
我们要获取的dense vector其实就是Hidden Layer的输出单元。**有的地方定为Input Layer和Hidden Layer之间的权重**，其实说的是一回事

$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

word2vec



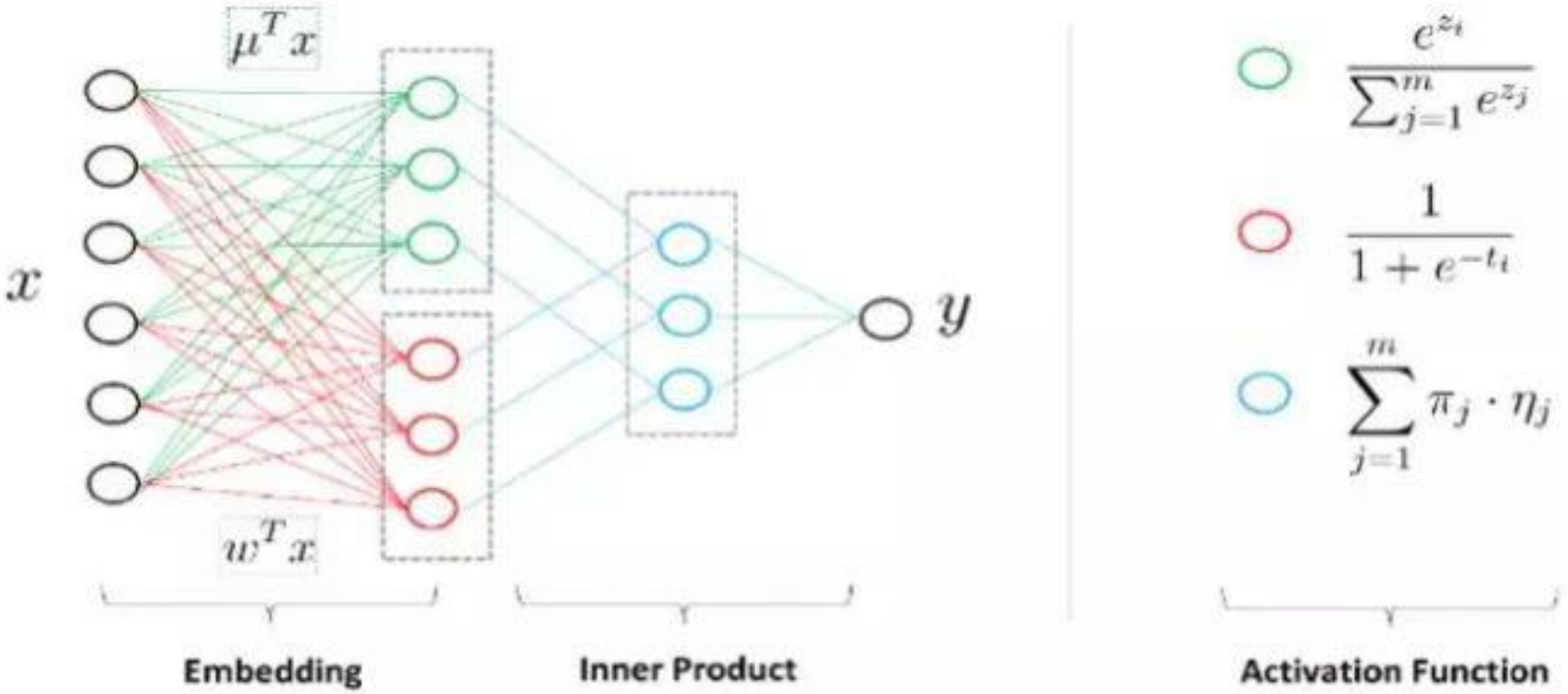
句子中的单词以one-hot的形式作为输入，然后乘以学好的word embedding矩阵Q，就直接取出单词对应的word embedding了

Dense embeddings you can download!

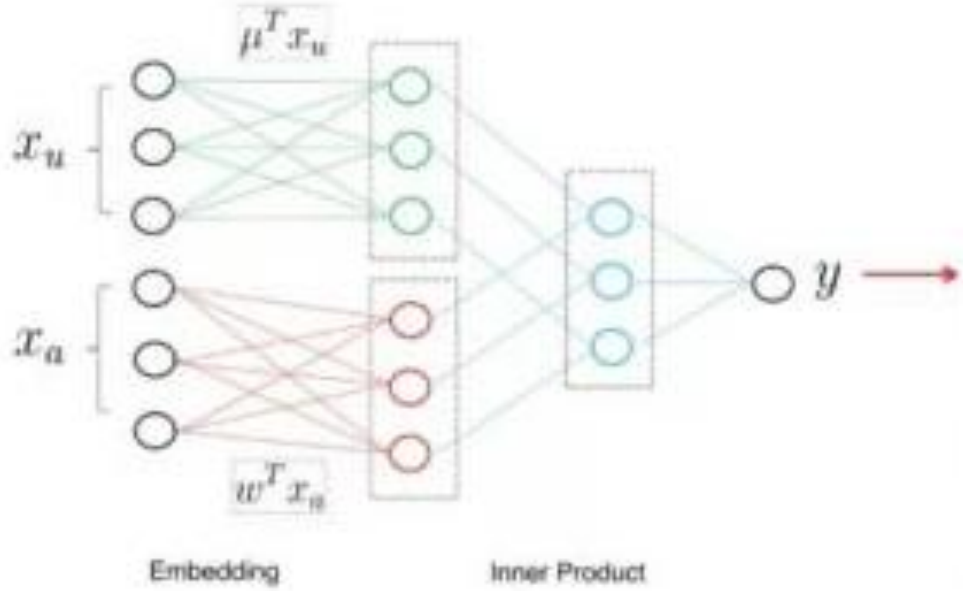
Word2vec (Mikolov et al.)
https://code.google.com/archive/p/word2vec/

Fasttext http://www.fasttext.cc/

Glove (Pennington, Socher, Manning)
http://nlp.stanford.edu/projects/glove/


$\mu^T x$
x
y
$w^T x$
$\frac{e^{z_i}}{\sum_{j=1}^{m} e^{z_j}}$
$\frac{1}{1+e^{-t_i}}$
$\sum_{j=1}^{m} \pi_j \cdot \eta_j$
Embedding
Inner Product
Activation Function

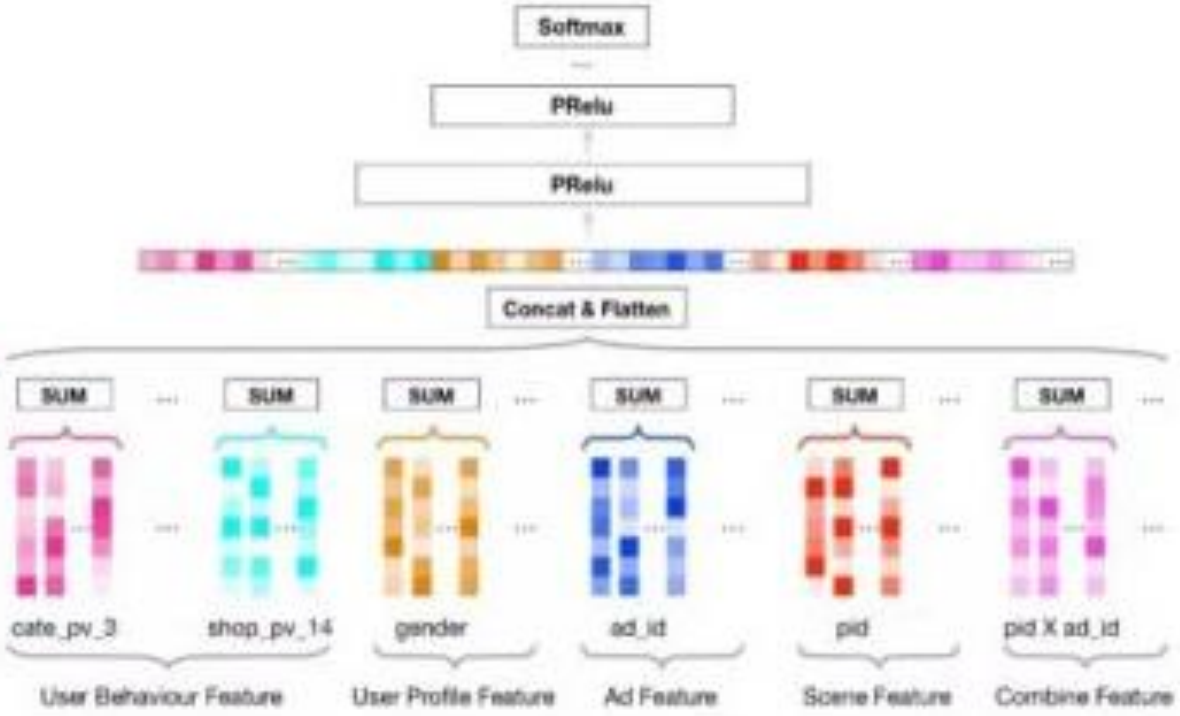
Dive into Deep too



MLR：三层网络

尝试：MLR中间层作为DNN输入
结果：没有提升效果！

端到端学习，有提升！--重要经验



大规模稀疏离散输入的分组embedding结构

- 基于XDL完成end-to-end训练，
- 数十亿样本，亿级参数，多GPU并行
- 16年流量实验确认提升效果

WDL – (2016)
Wide & Deep Learning for Recommender Systems

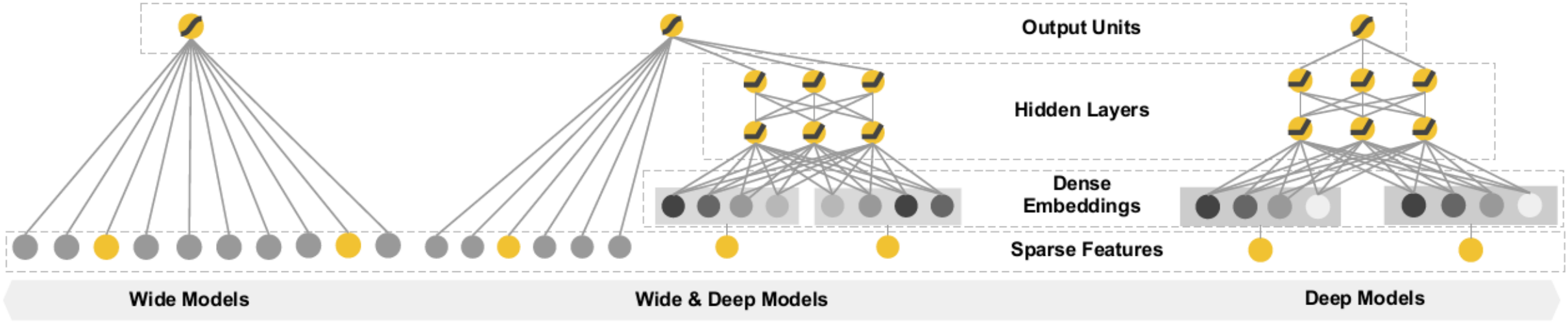
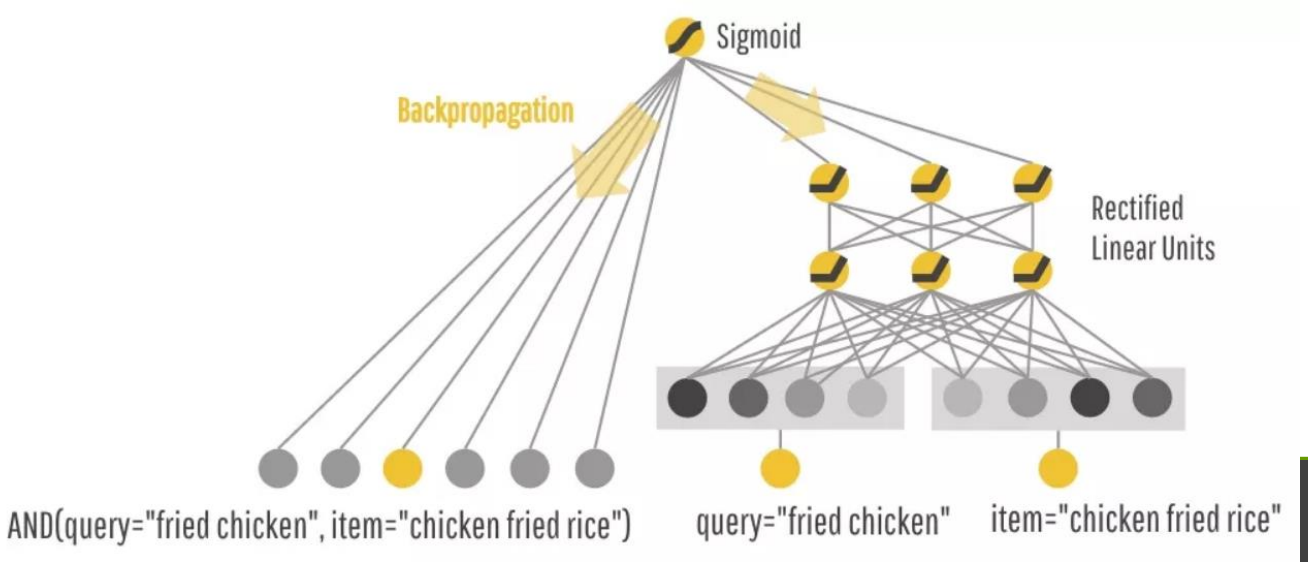
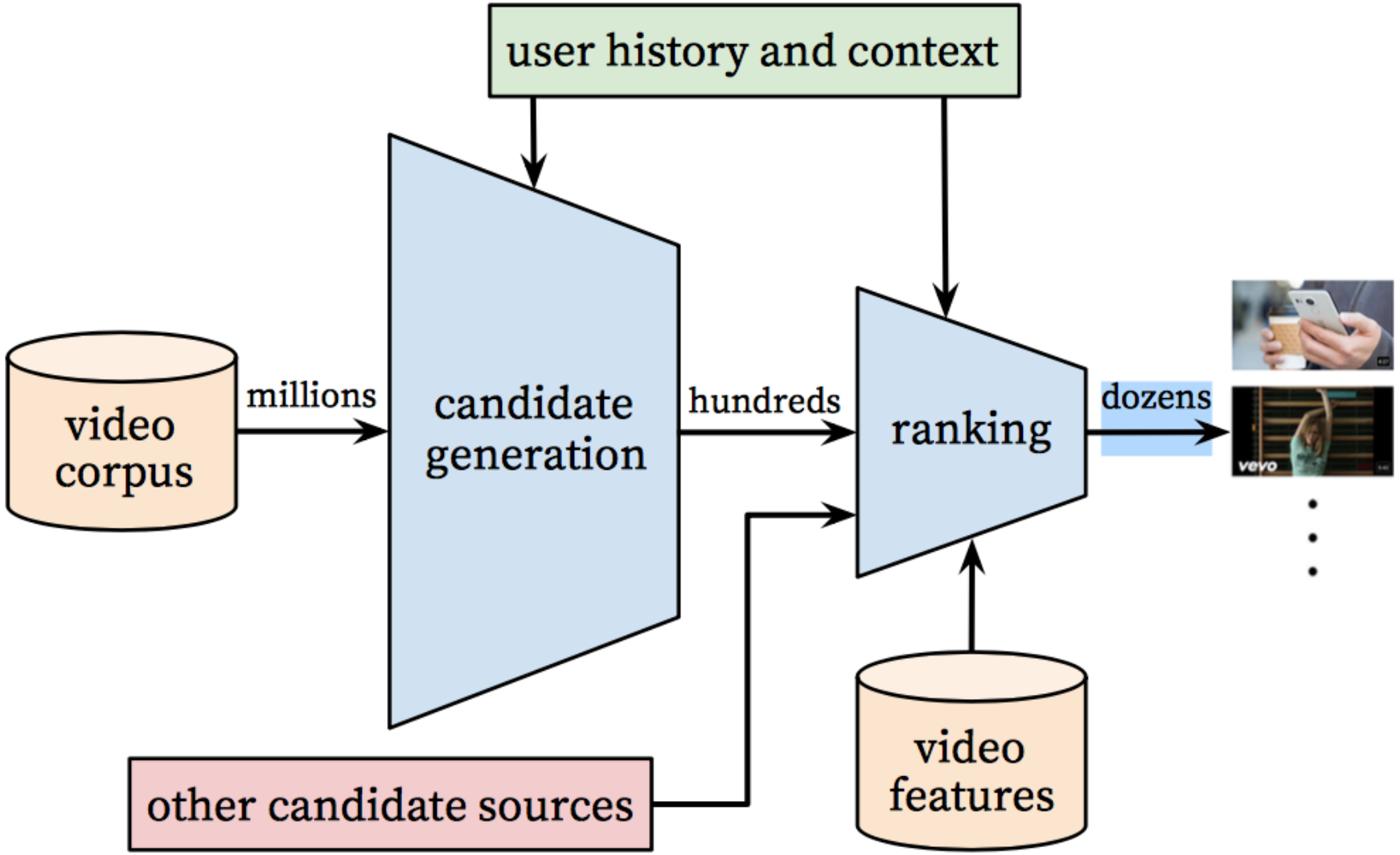
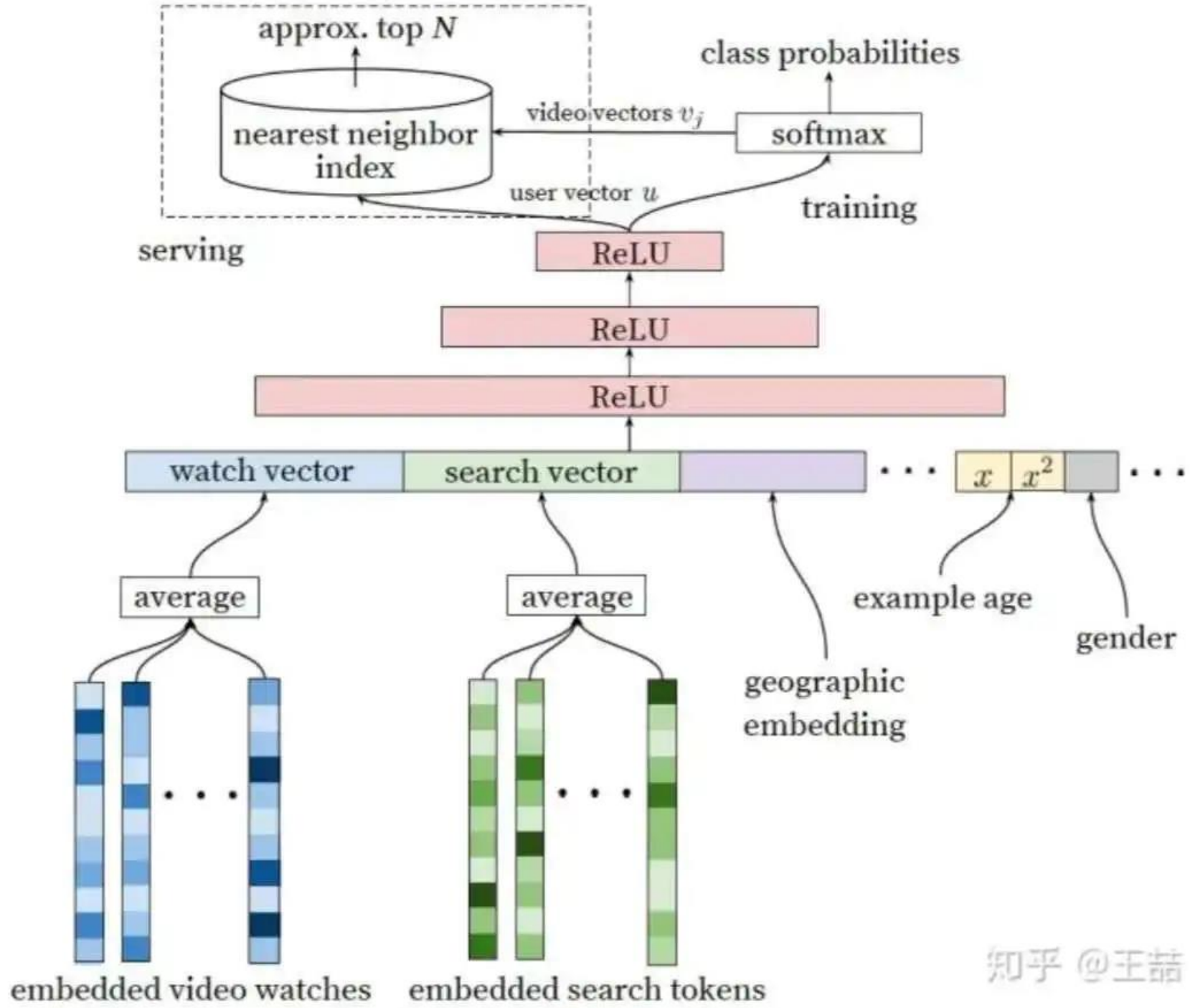


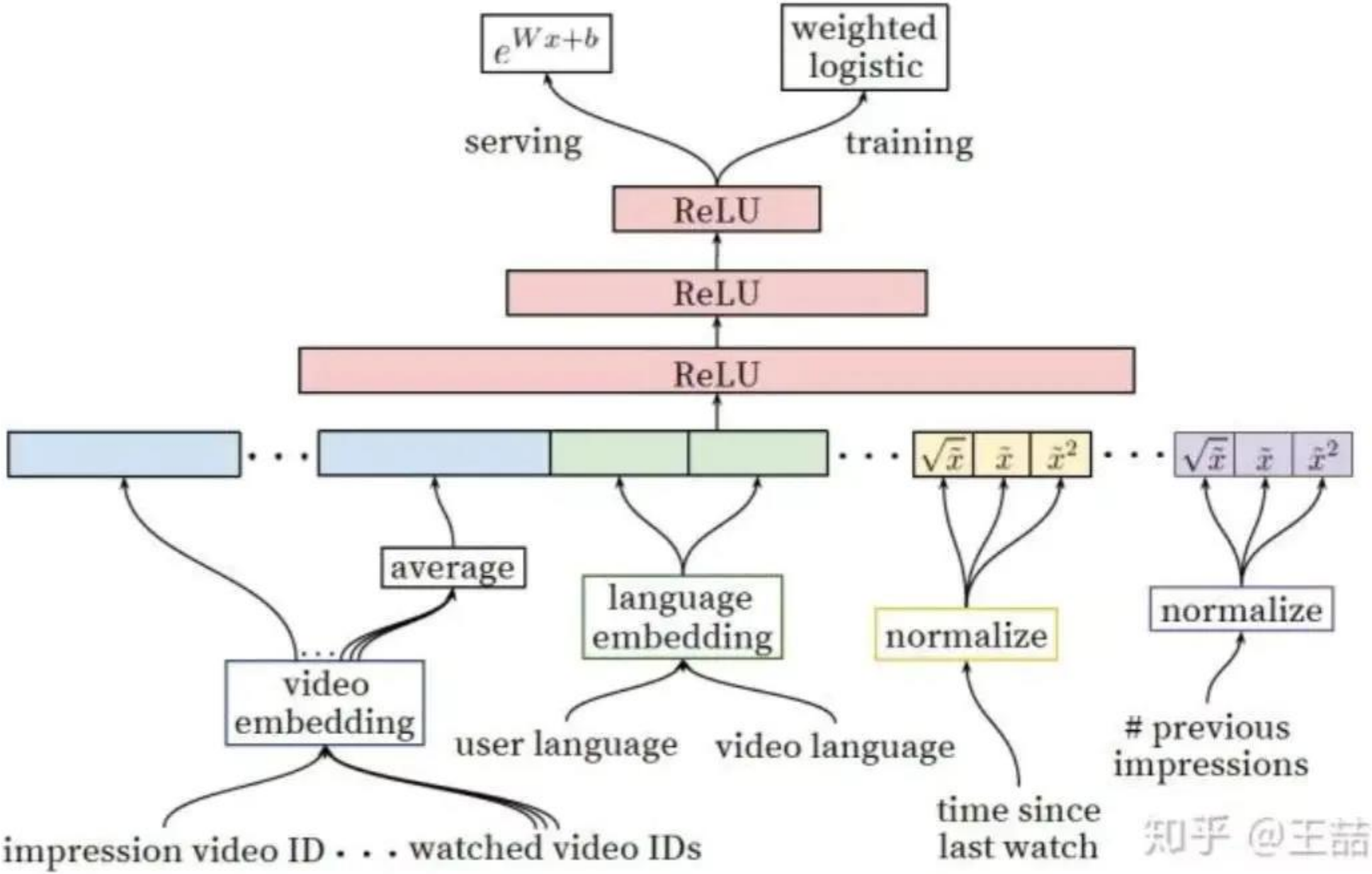
Figure 1: The spectrum of Wide & Deep models.



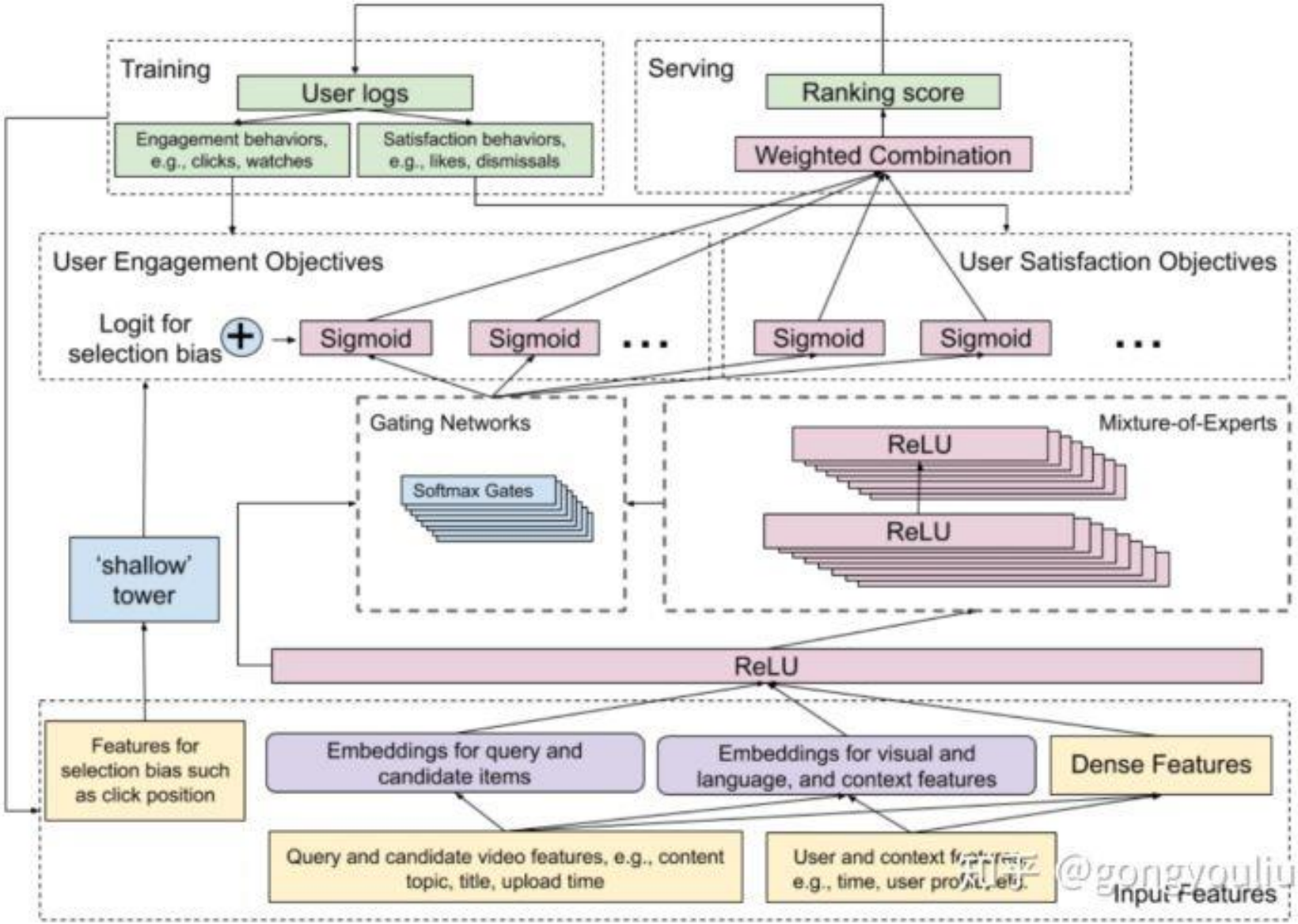
YouTube recommender system on DL (2016)

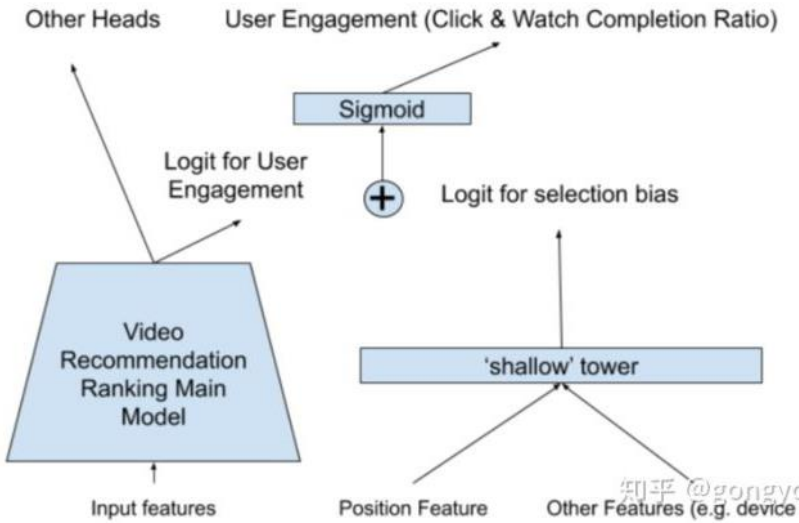



approx. top N
nearest neighbor index
video vectors v_j
softmax
class probabilities
user vector u
serving
training
ReLU
ReLU
ReLU
watch vector
search vector
x
x^2
average
average
example age
gender
geographic embedding
embedded video watches
embedded search tokens


e^{Wx+b}
weighted logistic
serving
training
ReLU
ReLU
ReLU
$\sqrt{\tilde{x}}$ $\tilde{x}$ $\tilde{x}^2$
$\sqrt{\tilde{x}}$ $\tilde{x}$ $\tilde{x}^2$
average
language embedding
normalize
normalize
video embedding
user language
video language
previous impressions
time since last watch
impression video ID
watched video IDs

Recommending What Video to Watch Next- A Multitask Ranking System [2019 Youtube]





DeepFM (2017)

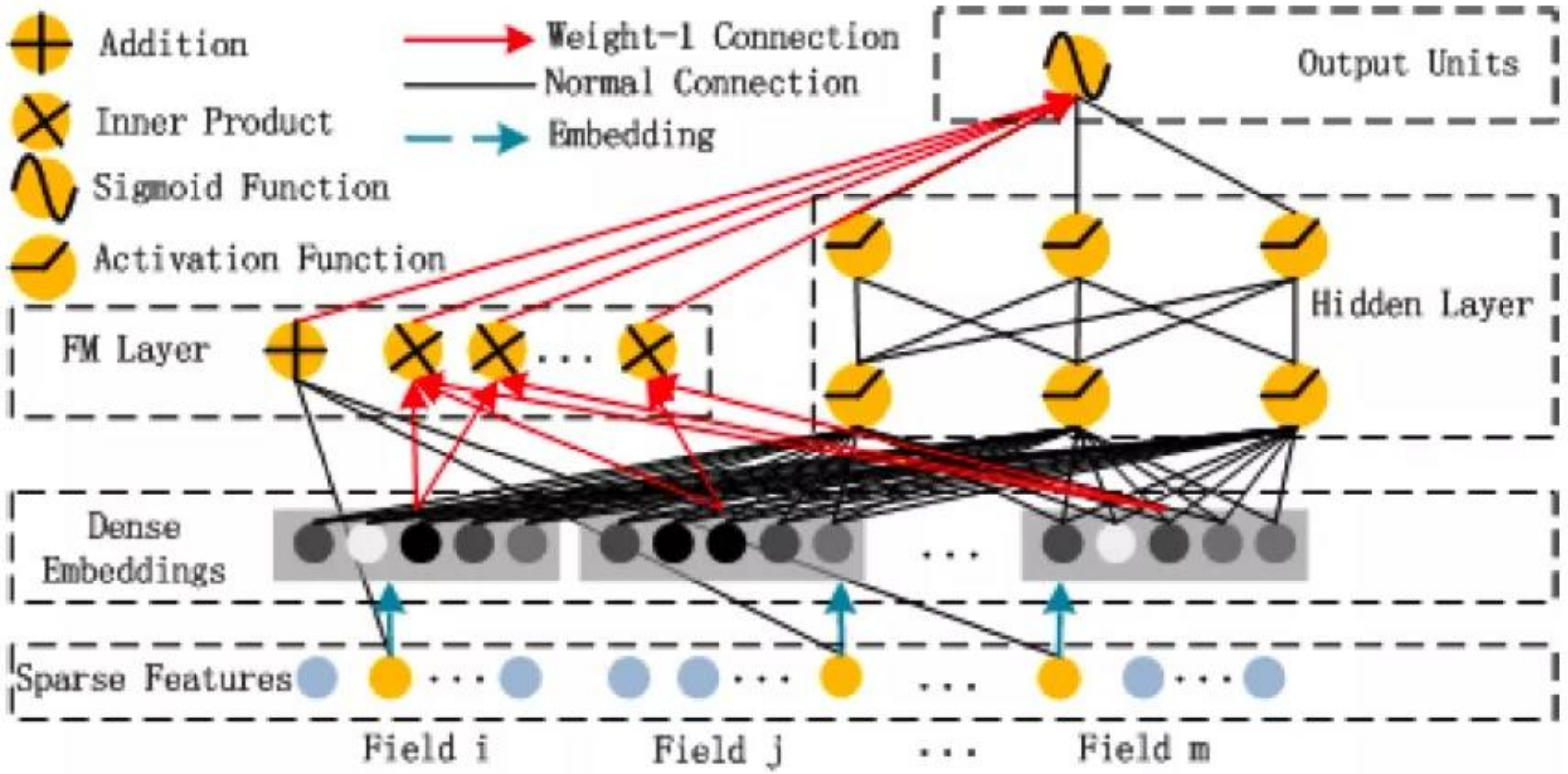
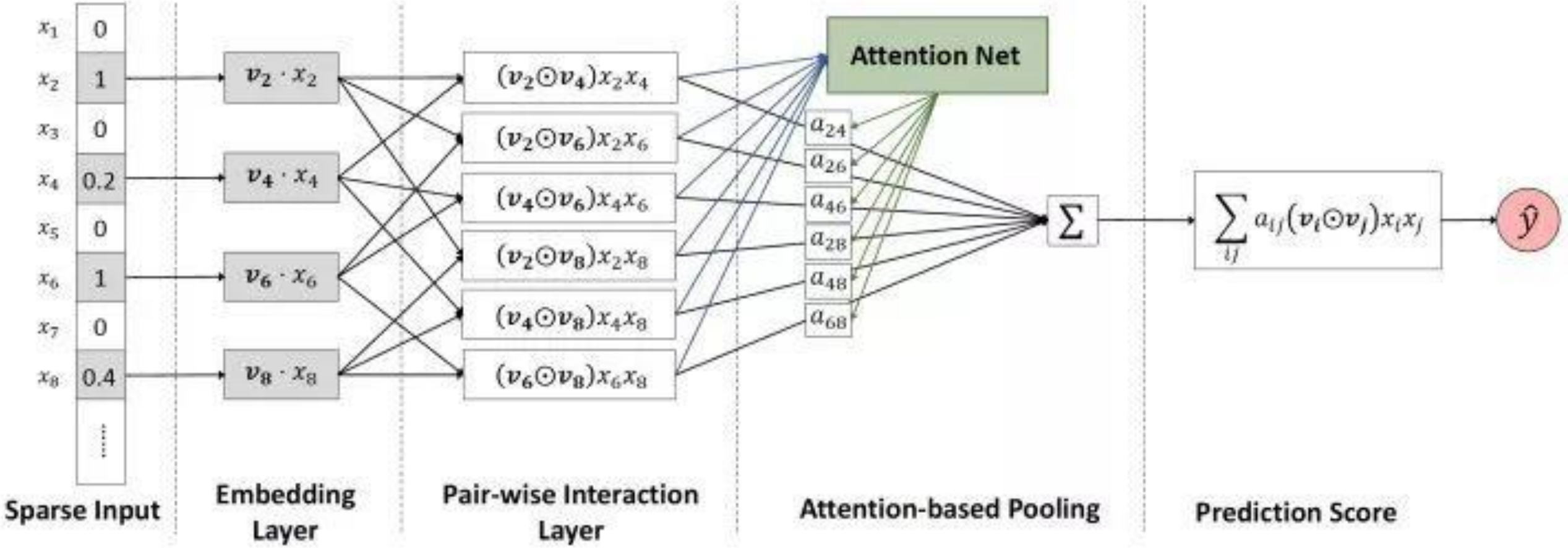


Figure 1: Wide & deep architecture of DeepFM. The wide and deep component share the same input raw feature vector, which enables DeepFM to learn low- and high-order feature interactions simultaneously from the input raw features.

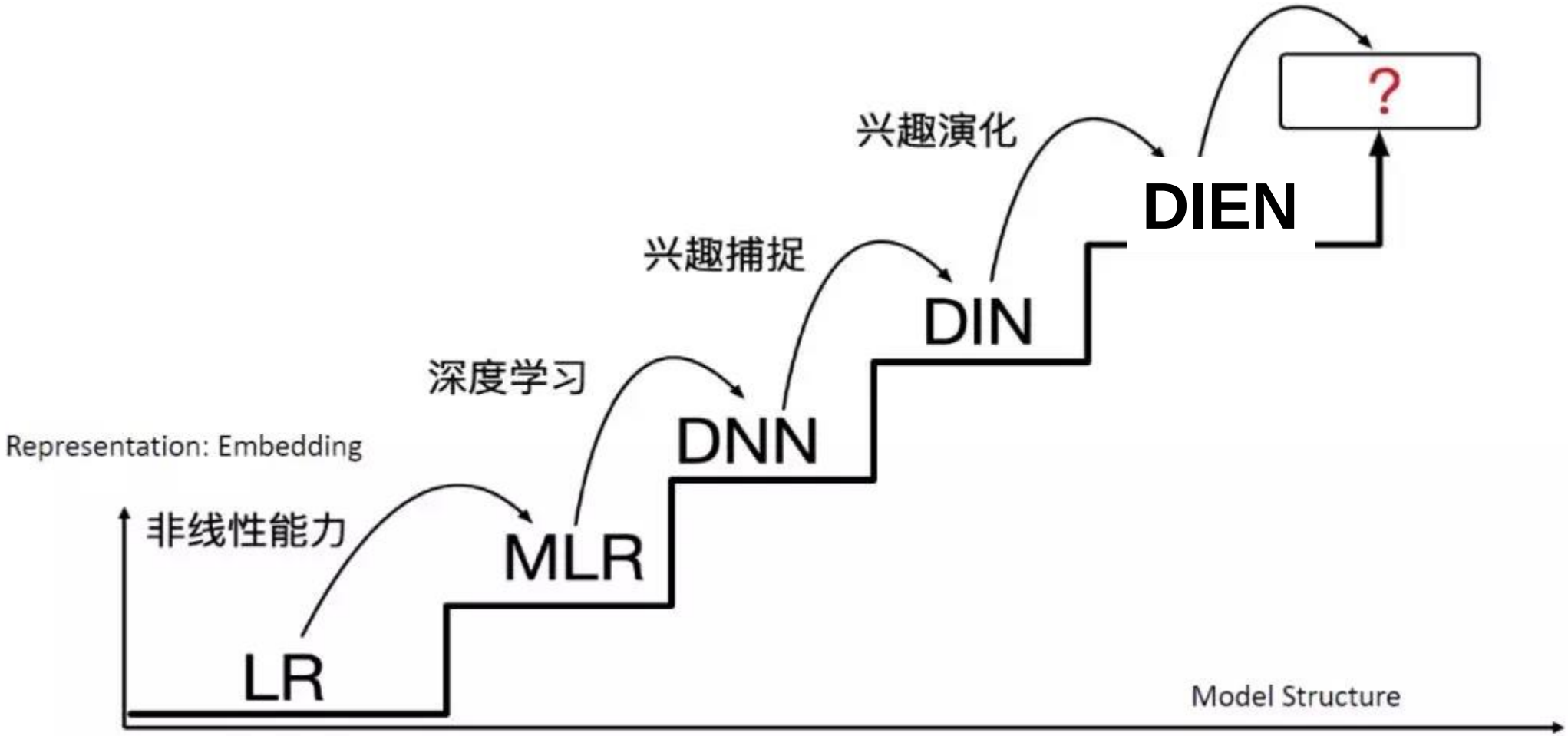
AFM（2017年）——引入Attention机制的FM



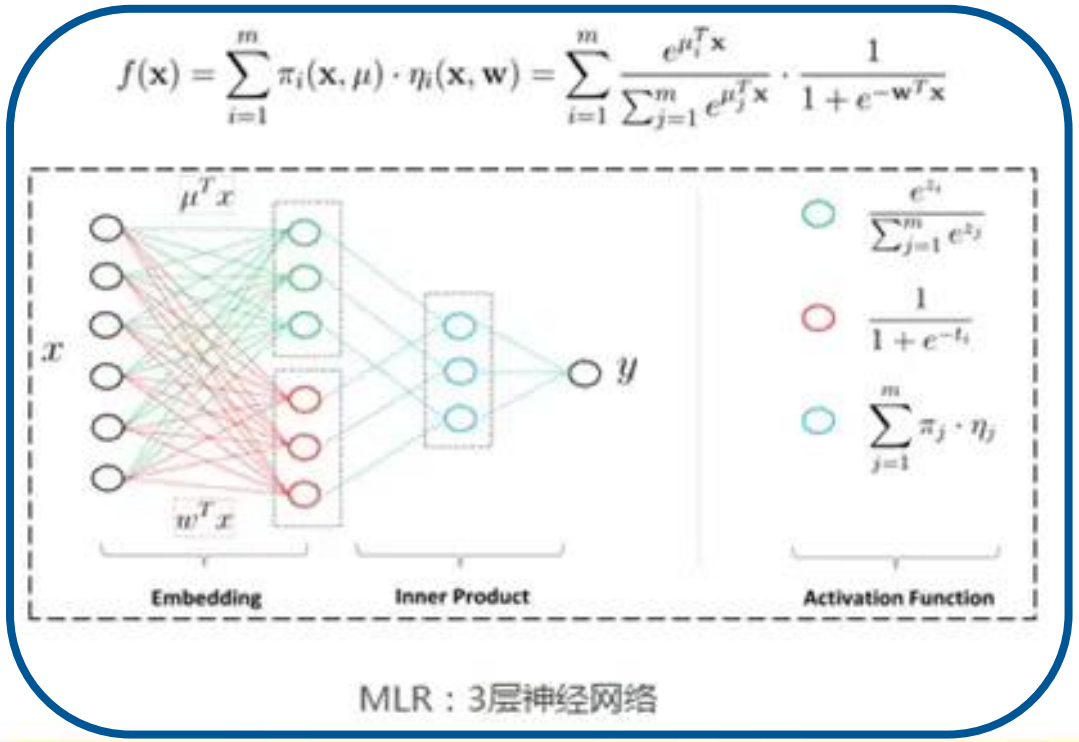
Alibaba - MLR (2017), DIN (2017), DIEN (2019)

- 阿里算法天才
- 盖坤

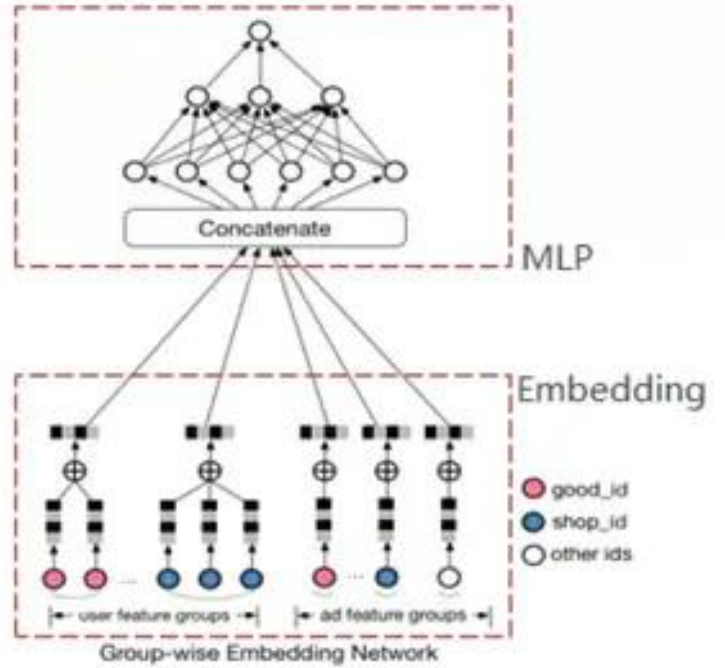

Logistic Regression
MLR
Embedding&MLP
2012
2013
2016.09
线性模型
非线性模型
深度模型
Deep Interest Network[1]
CrossMedia Network[2]
Rocket Launching[3]
Deep Interest Evolution[4]
2016.11
2017.08
2017.10
2018.06
电商OP
内容化特征
模型压缩
兴趣演化


?
兴趣演化
DIEN
兴趣捕捉
DIN
深度学习
DNN
Representation: Embedding
非线性能力
MLR
LR
Model Structure

- **MLR (Mixed Logistic Regression. Proposed and applied in 2011-2-12, Published in 2017)**



MLR：3层神经网络



GwEN：第一代deep CTR model

	特征分组	参数规模	表达能力	训练方法	样本量
MLR模型	2组	O(分片数m*输入D)	固定激活函数，内积函数拟合label	Batch	百亿
GwEN网络	20组	O(Embedding维度*输入D)	任意激活函数，MLP网络拟合label	Batch-based	百亿

推演结论：如果embedding维度跟MLR的分片数m相同，那么GwEN的参数规模跟MLR类似，但表达能力将大幅度提升，靠谱！

GMV（全称Gross Merchandise Volume），即商品交易总额[1]，是成交总额（一定时间段内）的意思。多用于电商行业，一般包含拍下未支付订单金额。[2]

https://baike.baidu.com/item/GMV/19473199

DIN (Deep Interest Network, 2017)

深度学习通用做法：

- Embedding嵌入技术
- 多个行为聚合成一个用户向量

问题： 用户兴趣多样，而一个向量表示能力受限

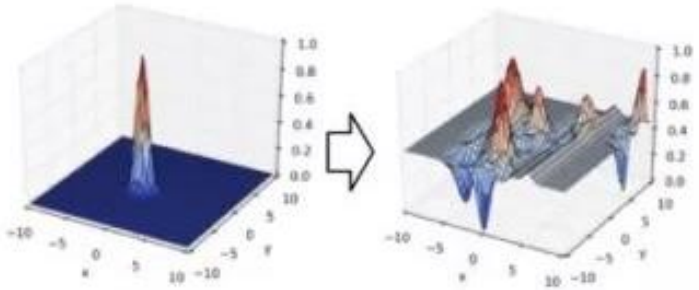
我们提出DIN（2016-2017）：

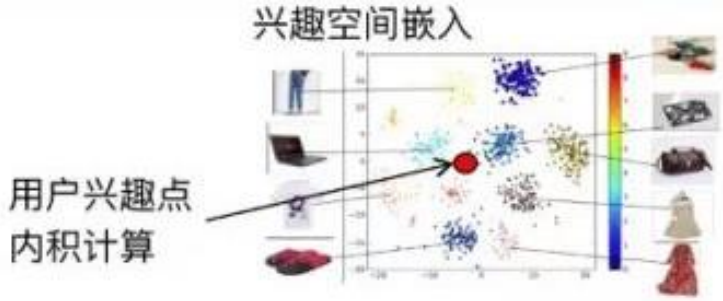
- 我们提出用户兴趣应该是多峰分布，而非一个向量点
- 当用户浏览某个商品时，只有部分（通常一个）兴趣被激发

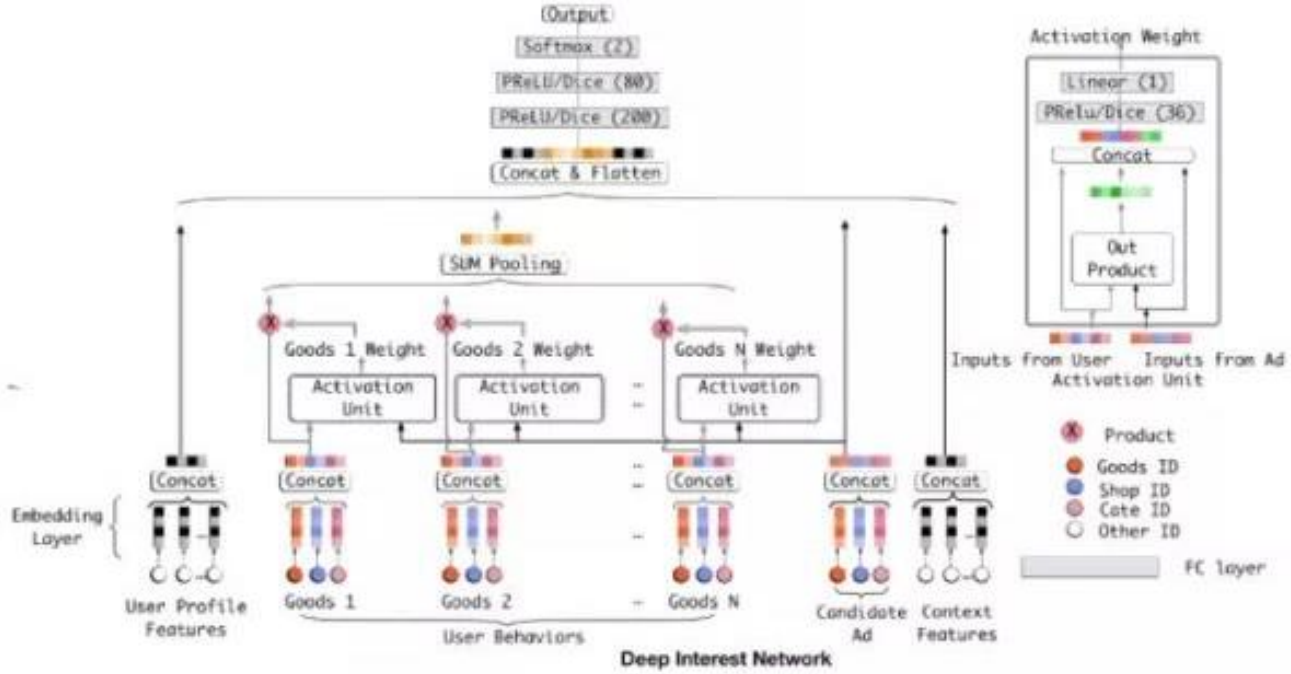
设计： 根据要预估的商品反向挑选行为子序列，去组成用户向量

非参数多峰兴趣，表示灵活性大大增加

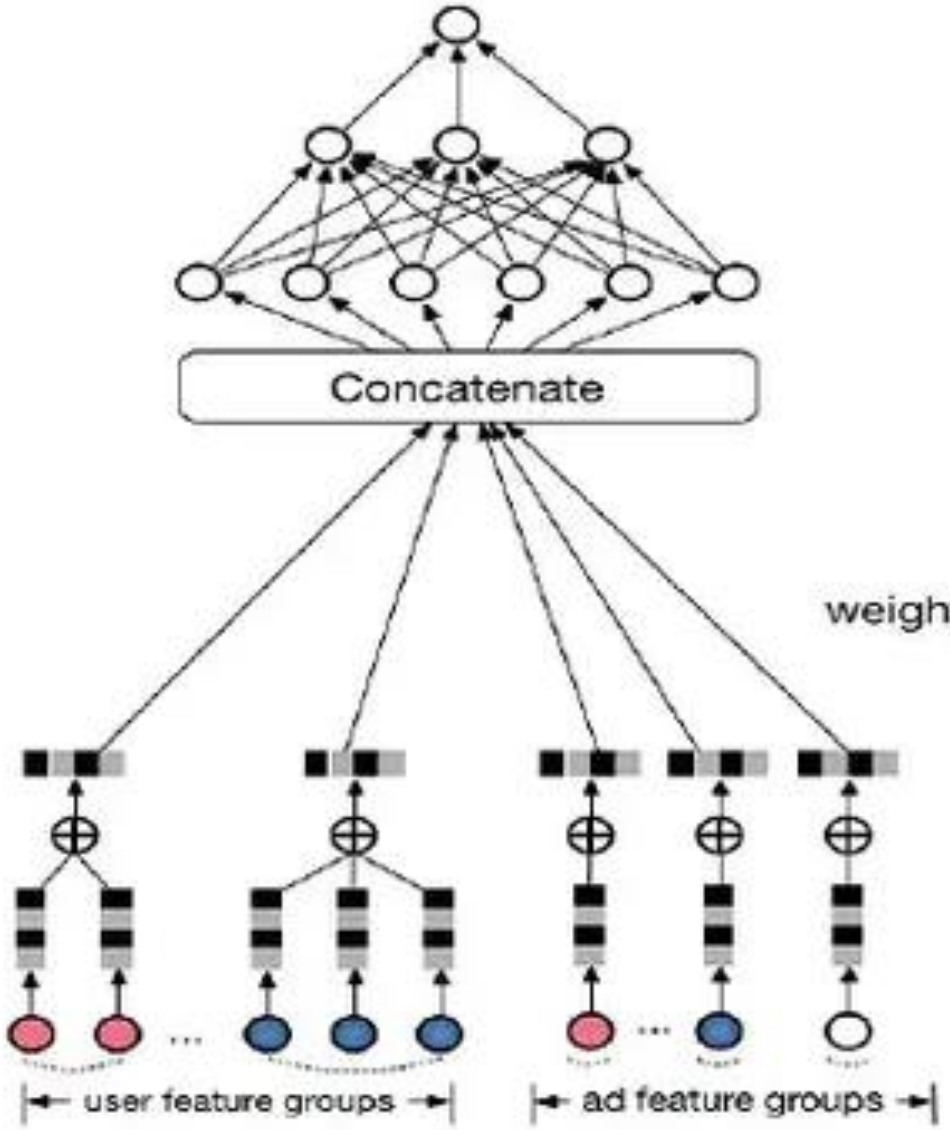
16年研制，17年开始上线，CTR/GMV显著提升



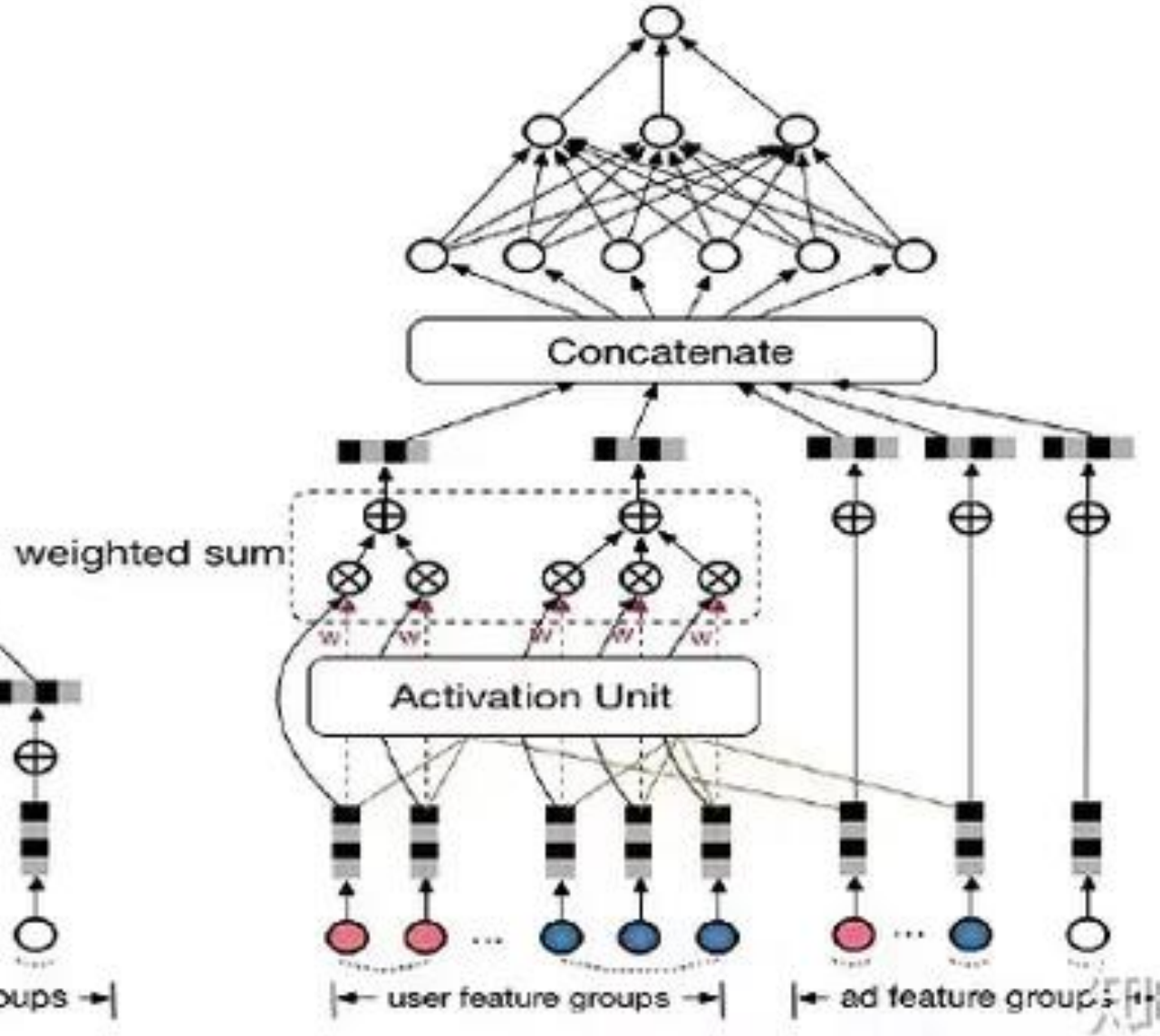




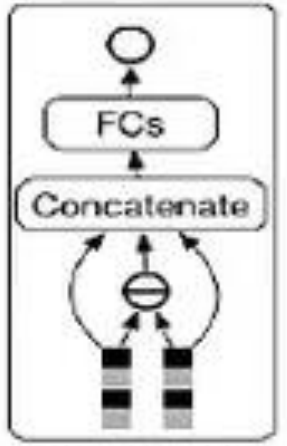
Guorui Zhou, Chengru Song, Xiaoqiang Zhu, Xiao Ma, Yanghui Yan, Xingya Dai, Han Zhu, Junqi Jin, Han Li, Kun Gai. Deep Interest Network for Click-Through Rate Prediction. https://arxiv.org/abs/1706.06978



Base Model

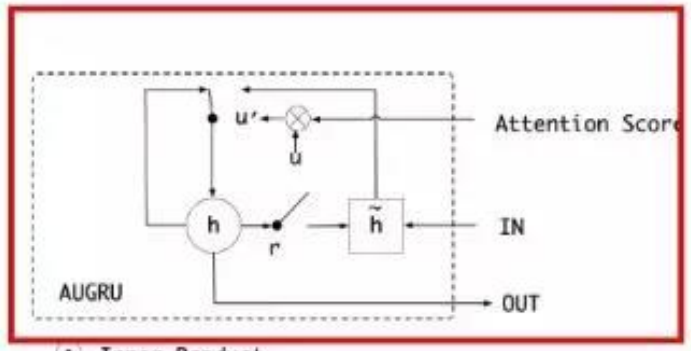


Deep Interest Network



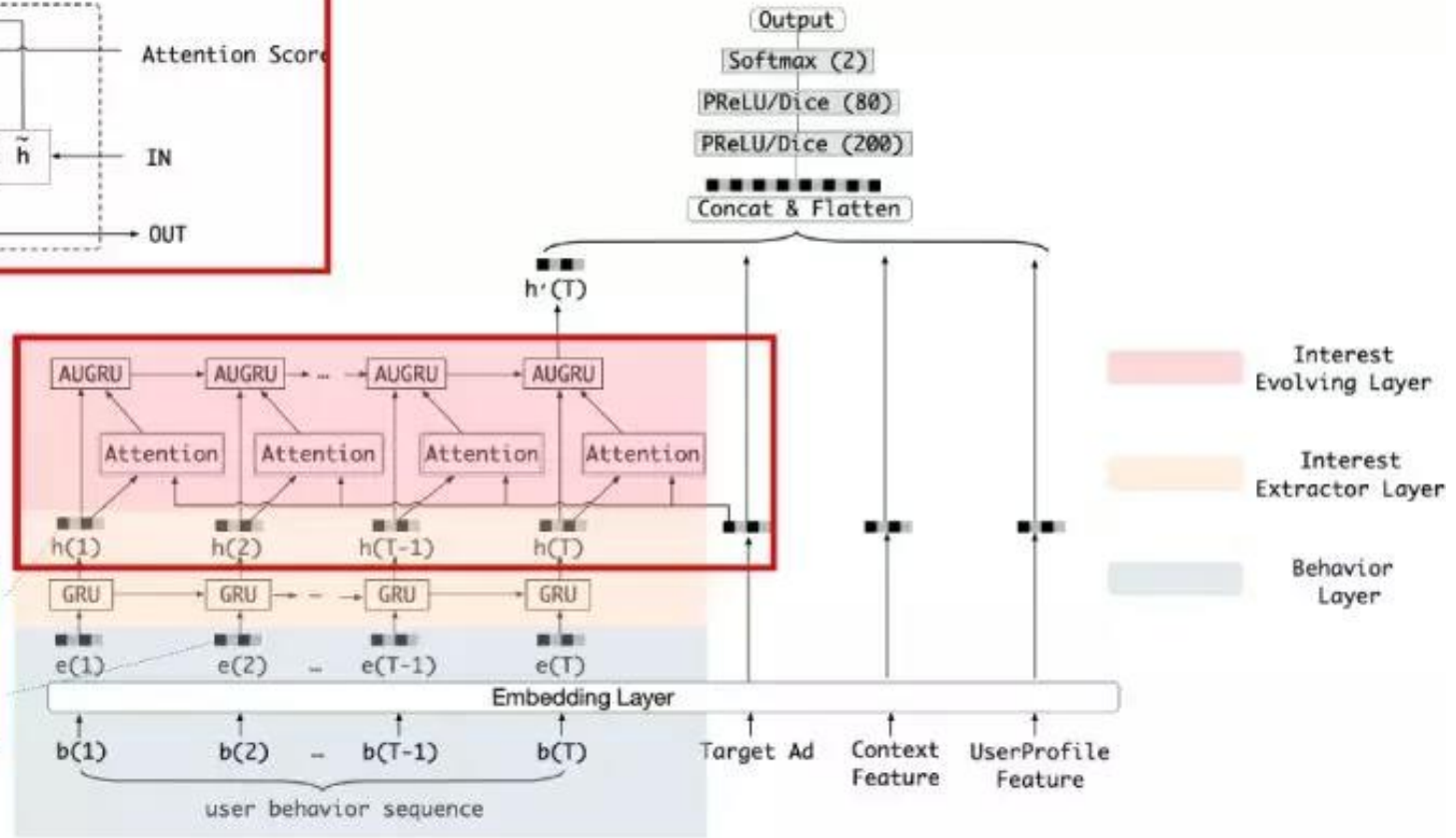
DIEN (Deep Interest Evolution Network, AAAI'19)

$$a_i = \frac{exp(h_i W e_{ad})}{\sum_{j=0}^{T} exp(h_j W e_{ad})}$$



$$u_t = \sigma(W^u i_t + U^u h_{t-1} + b^u)$$
$$r_t = \sigma(W^r i_t + U^r h_{t-1} + b^r)$$
$$\hat{h}_t = tanh(W^h i_t + r_t \circ U^h h_{t-1} + b^h)$$

$$u'_t = u_t * a_t$$
$$h_t = (1 - u'_t) \circ h_{t-1} + u'_t \circ \hat{h}_t$$



2020 CAN

https://arxiv.org/abs/2011.05625

- **CAN**: Revisiting Feature Co-Action for Click-Through Rate Prediction
- Guorui Zhou, Weijie Bian, Kailun Wu, Lejian Ren, Qi Pi, Yujing Zhang, Can Xiao, Xiang-Rong Sheng, Na Mou, Xinchen Luo, Chi Zhang, Xianjie Qiao, Shiming Xiang, Kun Gai, Xiaoqiang Zhu, Jian Xu

Computer Science > Information Retrieval

[Submitted on 11 Nov 2020]

CAN: Revisiting Feature Co-Action for Click-Through Rate Prediction

Guorui Zhou, Weijie Bian, Kailun Wu, Lejian Ren, Qi Pi, Yujing Zhang, Can Xiao, Xiang-Rong Sheng, Na Mou, Xinchen Luo, Chi Zhang, Xianjie Qiao, Shiming Xiang, Kun Gai, Xiaoqiang Zhu, Jian Xu

Inspired by the success of deep learning, recent industrial Click-Through Rate (CTR) prediction models have made the transition from traditional shallow approaches to deep approaches. Deep Neural Networks (DNNs) are known for its ability to learn non-linear interactions from raw feature automatically, however, the non-linear feature interaction is learned in an implicit manner. The non-linear interaction may be hard to capture and explicitly model the \textit{co-action} of raw feature is beneficial for CTR prediction. \textit{Co-action} refers to the collective effects of features toward final prediction.

In this paper, we argue that current CTR models do not fully explore the potential of feature co-action. We conduct experiments and show that the effect of feature co-action is underestimated seriously. Motivated by our observation, we propose feature Co-Action Network (CAN) to explore the potential of feature co-action. The proposed model can efficiently and effectively capture the feature co-action, which improves the model performance while reduce the storage and computation consumption. Experiment results on public and industrial datasets show that CAN outperforms state-of-the-art CTR models by a large margin. Up to now, CAN has been deployed in the Alibaba display advertisement system, obtaining averaging 12\% improvement on CTR and 8\% on RPM.

dblp.org/pid/59/2902.html

自媒体 翻译 有道词典 谷歌镜像 Bing 百度 StackOve 天眼查 西林街 盘天下 书格 少数派 V2ray Greenplum CCTV

Kun Gai

> Home > Persons

by year | Dagstuhl

https://dblp.org/pid/59/2902.html

[–] 2020 – today

GSP: Generalized Second Price

2021

[c34] Zhilin Zhang, Xiangyu Liu, Zhenzhe Zheng, Chenrui Zhang, Miao Xu, Junwei Pan, Chuan Yu, Fan Wu, Jian Xu, Kun Gai:
Optimizing Multiple Performance Metrics with Deep GSP Auctions for E-commerce Advertising. WSDM 2021: 993-1001

[i35] Jin Li, Jie Liu, Shangzhou Li, Yao Xu, Ran Cao, Qi Li, Biye Jiang, Guan Wang, Han Zhu, Kun Gai, Xiaoqiang Zhu:
Truncation-Free Matching System for Display Advertising at Alibaba. CoRR abs/2102.09283 (2021)

2020

[c33] Bo Wang, Quan Chen, Min Zhou, Zhiqiang Zhang, Xiaogang Jin, Kun Gai:
Progressive Feature Polishing Network for Salient Object Detection. AAAI 2020: 12128-12135

[c32] Zhaoqing Peng, Junqi Jin, Lan Luo, Yaodong Yang, Rui Luo, Jun Wang, Weinan Zhang, Miao Xu, Chuan Yu, Tiejian Luo, Han Li, Jian Xu, Kun Gai:
Sequential Advertising Agent with Interpretable User Hidden Intents. AAMAS 2020: 1966-1968

[c31] Liyi Guo, Rui Lu, Haoqi Zhang, Junqi Jin, Zhenzhe Zheng, Fan Wu, Jin Li, Haiyang Xu, Han Li, Wenkai Lu, Jian Xu, Kun Gai:
A Deep Prediction Network for Understanding Advertiser Intent and Satisfaction. CIKM 2020: 2501-2508

[c30] Zhaoqing Peng, Junqi Jin, Lan Luo, Yaodong Yang, Rui Luo, Jun Wang, Weinan Zhang, Haiyang Xu, Miao Xu, Chuan Yu, Tiejian Luo, Han Li, Jian Xu, Kun Gai:
Learning to Infer User Hidden States for Online Sequential Advertising. CIKM 2020: 2677-2684

[–] Refine list

showing all 71 records

refine by search term

refine by type
- Journal Articles (only)
- Conference and Workshop Papers (only)
- Informal Publications (only)

select all | deselect all

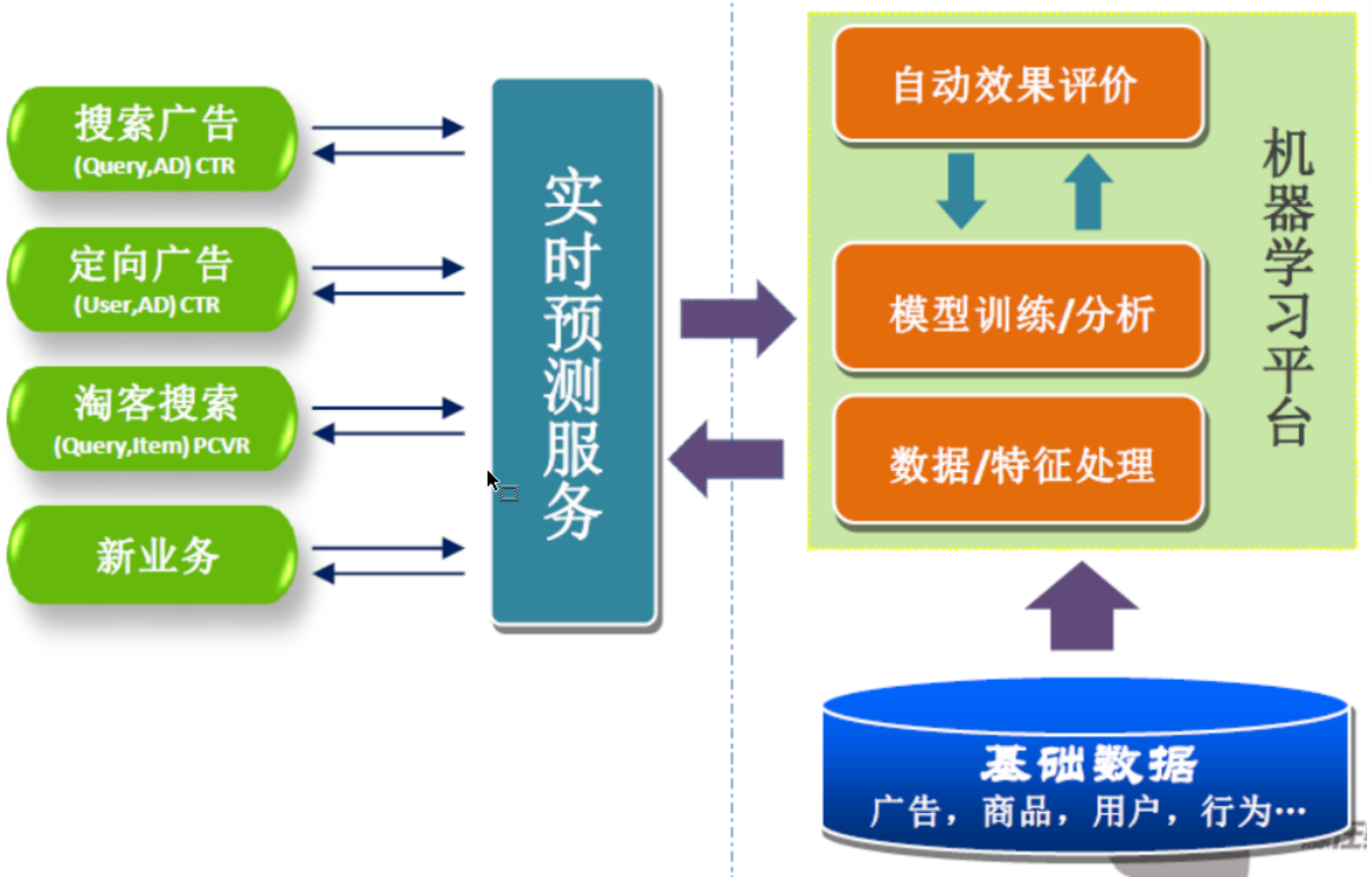
refine by coauthor
Jian Xu (30)
Han Li (28)
Xiaoqiang Zhu (25)
Guorui Zhou (20)
Junqi Jin (19)
Weinan Zhang 0001 (13)
Jun Wang 0012 (11)
Han Zhu (11)
Chuan Yu (11)
Weijie Bian (10)
127 more options

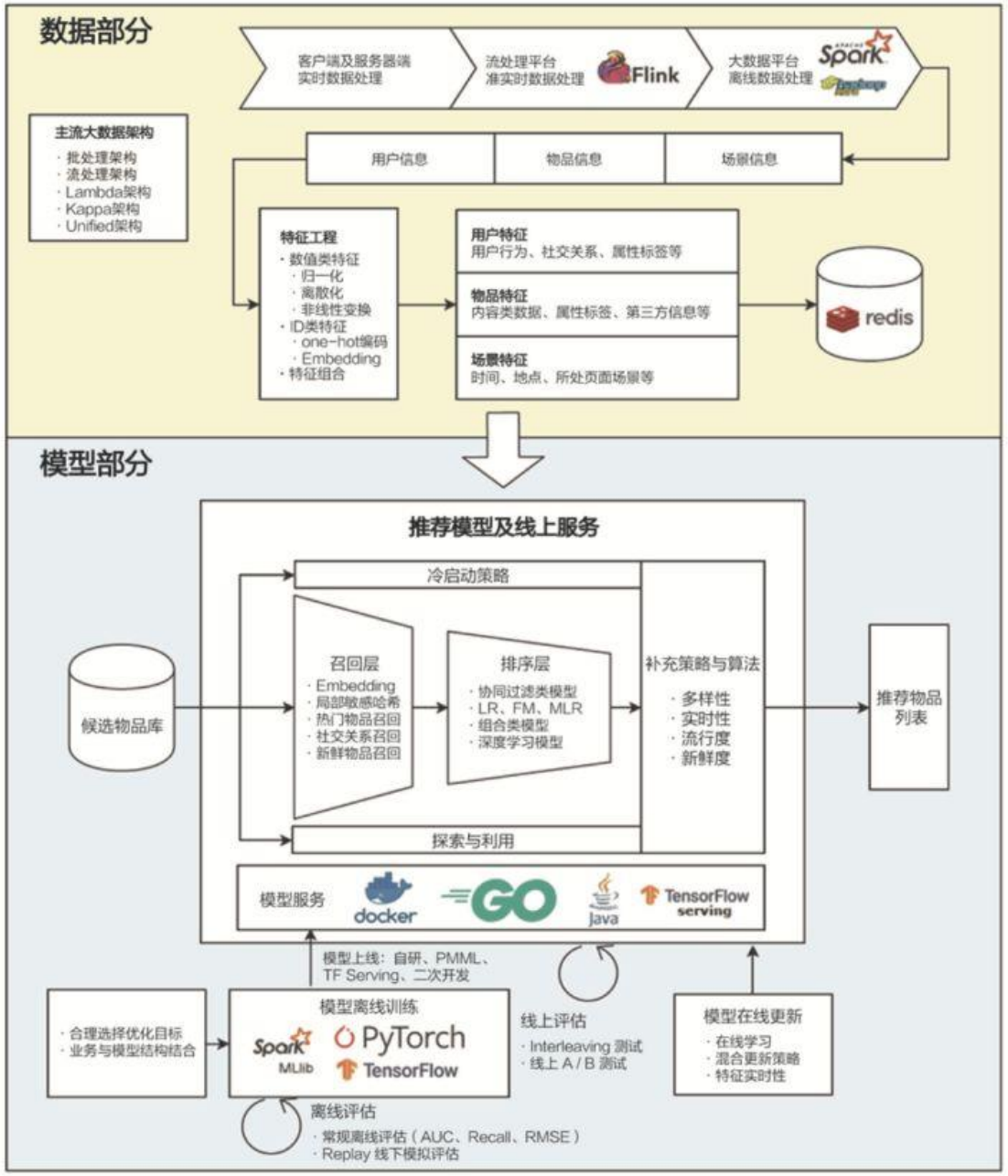
refine by venue
CoRR (35)
CIKM (8)
KDD (5)
AAAI (4)
SIGIR (3)
ICASSP (2)
ICML (2)
CVPR (2)
Neurocomputing (1)

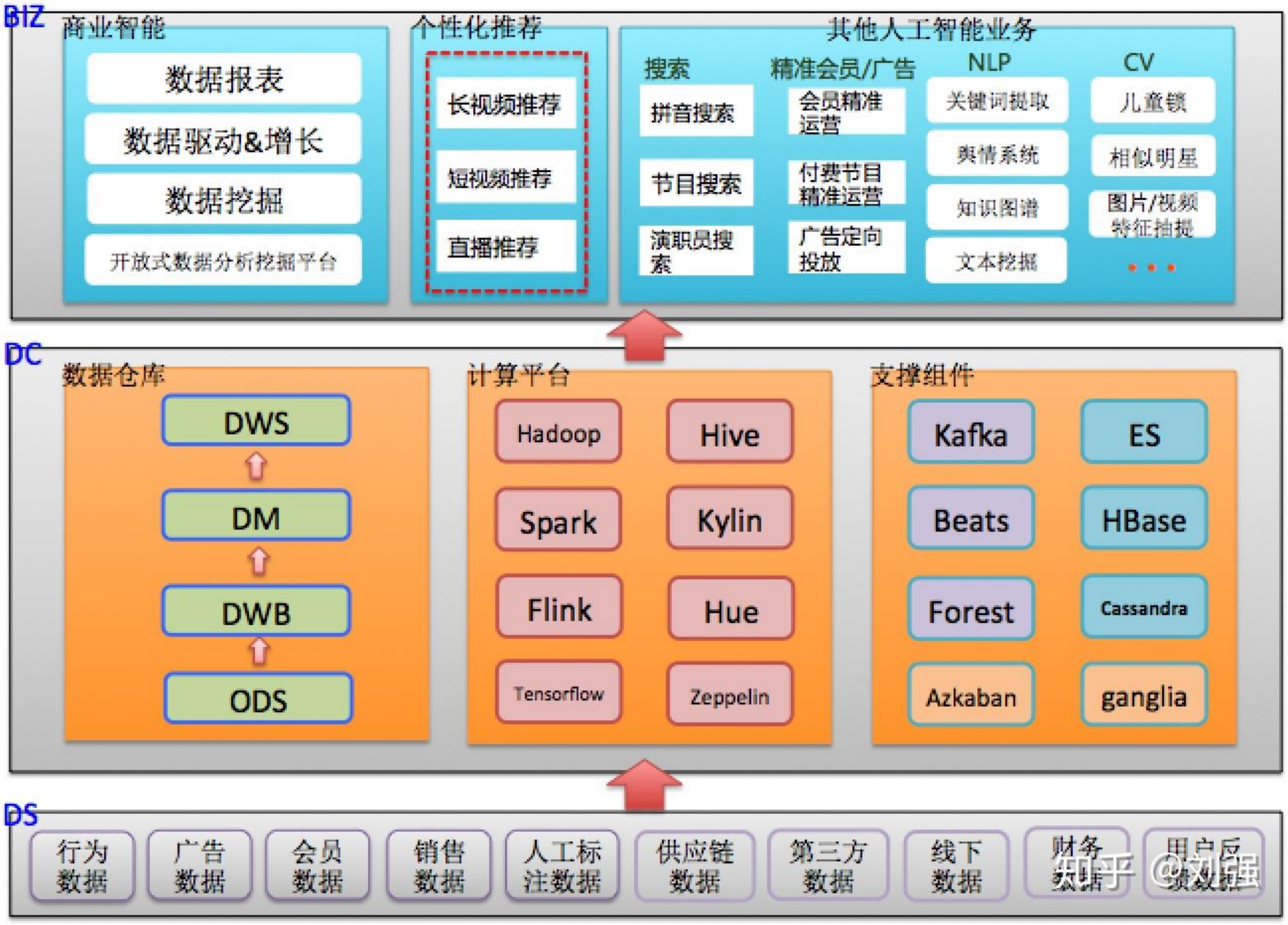
“精准广告”

- **About Recommendation and computing adv [关于计算广告]**
  - From Recommendation to Computing Advs
  - You should know how to earn money from ads in Google! – **RTB** (Real Time Bargain)
- **Algorithms**
  - CF, FM, LR, GBDT-LR (2014), Embedding, LDA (Latent Dirichlet Analysis) with EM, DL
    - Feature engineering is critical
- **Big Data way**

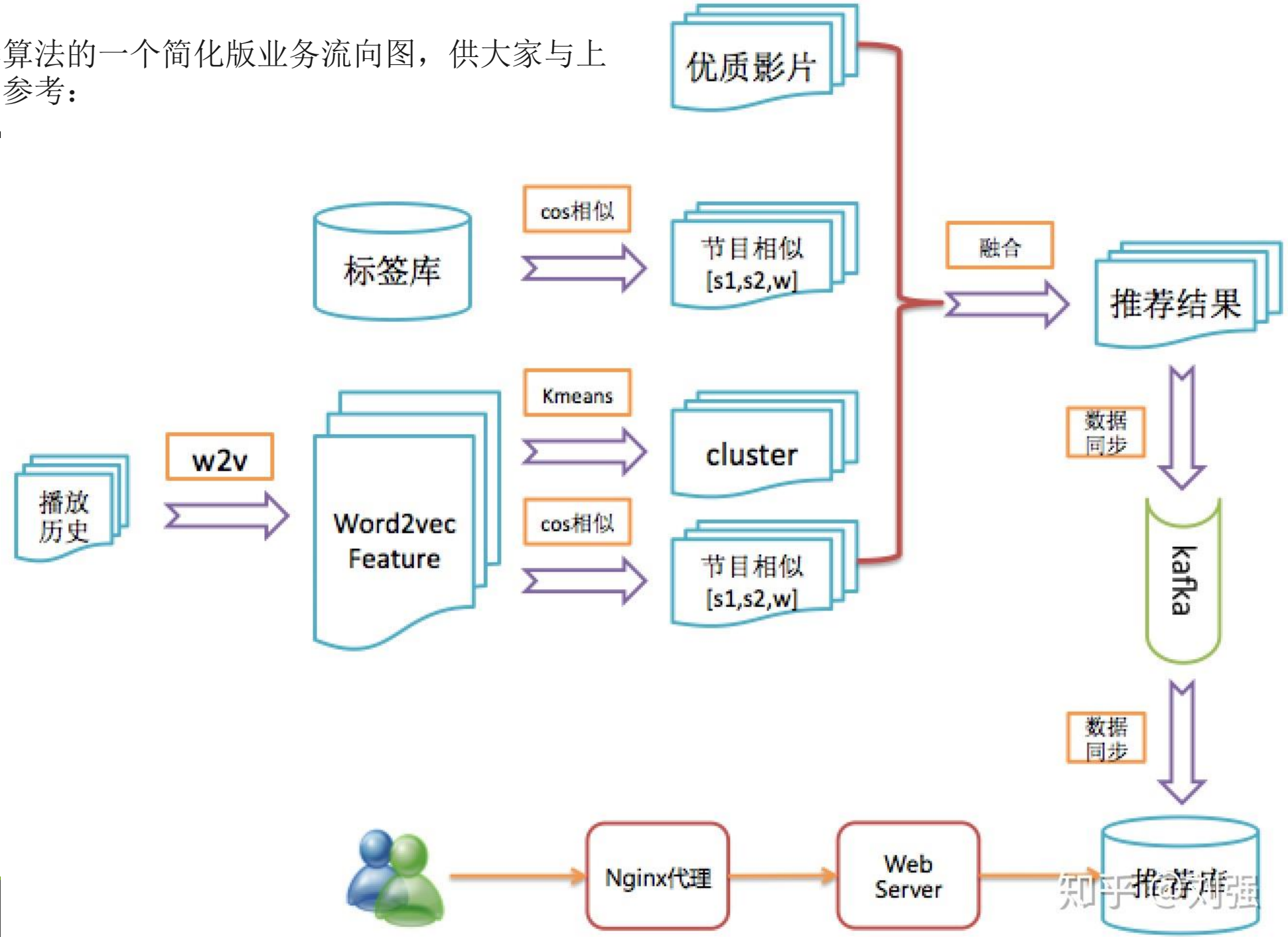
大规模机器学习平台

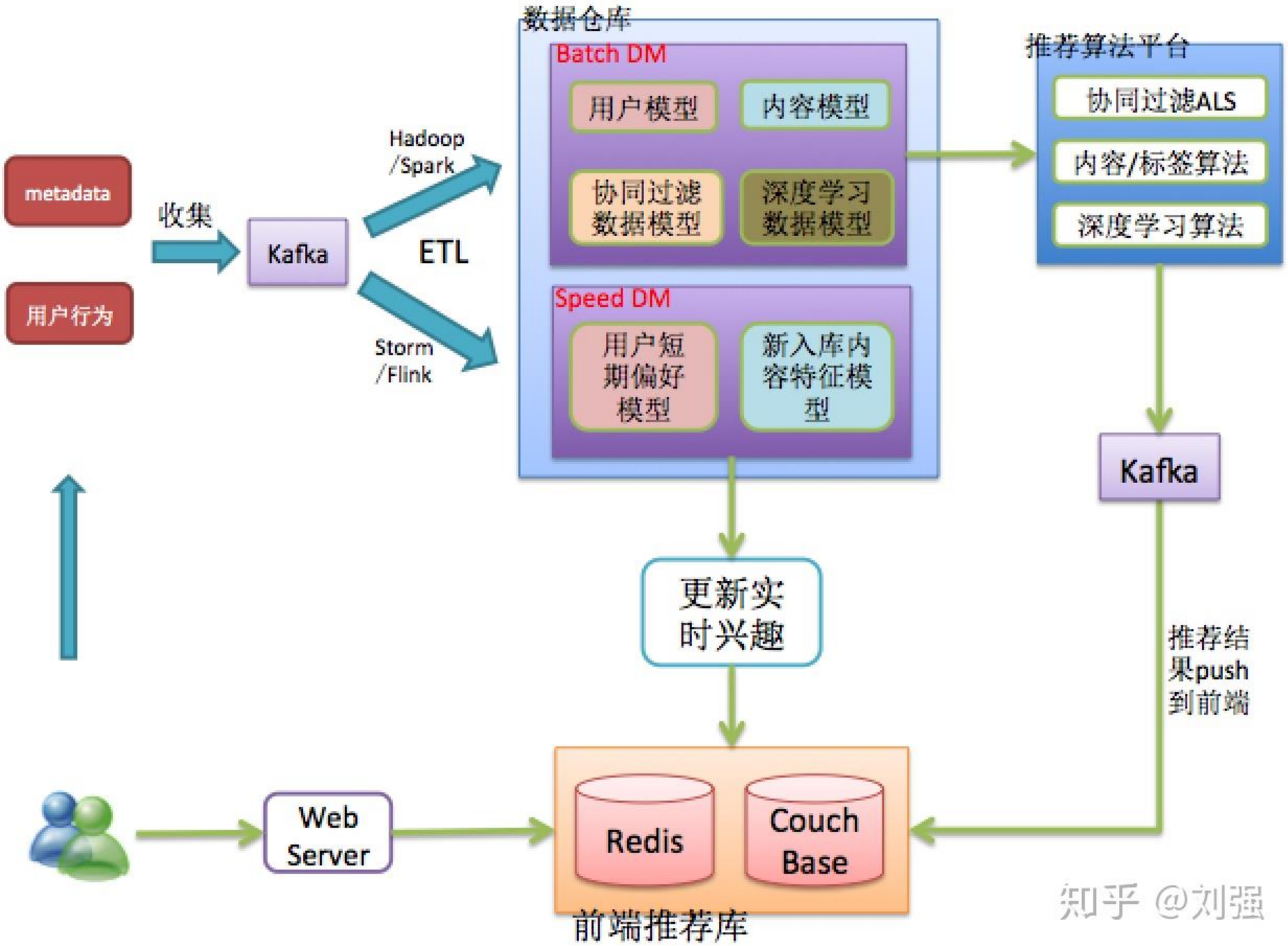

搜索广告
(Query,AD) CTR
定向广告
(User,AD) CTR
淘客搜索
(Query,Item) PCVR
新业务
实时预测服务
自动效果评价
模型训练/分析
数据/特征处理
机器学习平台
基础数据
广告，商品，用户，行为…


数据部分
客户端及服务器端
实时数据处理
流处理平台
准实时数据处理
Flink
大数据平台
离线数据处理
Spark
主流大数据架构
· 批处理架构
· 流处理架构
· Lambda架构
· Kappa架构
· Unified架构
用户信息
物品信息
场景信息
特征工程
· 数值类特征
· 归一化
· 离散化
· 非线性变换
· ID类特征
· one-hot编码
· Embedding
· 特征组合
用户特征
用户行为、社交关系、属性标签等
物品特征
内容类数据、属性标签、第三方信息等
场景特征
时间、地点、所处页面场景等
redis
模型部分
推荐模型及线上服务
冷启动策略
候选物品库
召回层
· Embedding
· 局部敏感哈希
· 热门物品召回
· 社交关系召回
· 新鲜物品召回
排序层
· 协同过滤类模型
· LR、FM、MLR
· 组合类模型
· 深度学习模型
补充策略与算法
· 多样性
· 实时性
· 流行度
· 新鲜度
推荐物品列表
探索与利用
模型服务
docker
GO
Java
TensorFlow serving
模型上线：自研、PMML、TF Serving、二次开发
· 合理选择优化目标
· 业务与模型结构结合
模型离线训练
Spark MLlib
PyTorch
TensorFlow
线上评估
· Interleaving 测试
· 线上 A / B 测试
模型在线更新
· 在线学习
· 混合更新策略
· 特征实时性
离线评估
· 常规离线评估（AUC、Recall、RMSE）
· Replay 线下模拟评估


BIZ
商业智能
数据报表
数据驱动&增长
数据挖掘
开放式数据分析挖掘平台
个性化推荐
长视频推荐
短视频推荐
直播推荐
其他人工智能业务
搜索
拼音搜索
节目搜索
演职员搜索
精准会员/广告
会员精准运营
付费节目精准运营
广告定向投放
NLP
关键词提取
舆情系统
知识图谱
文本挖掘
CV
儿童锁
相似明星
图片/视频特征抽提
DC
数据仓库
DWS
DM
DWB
ODS
计算平台
Hadoop
Hive
Spark
Kylin
Flink
Hue
Tensorflow
Zeppelin
支撑组件
Kafka
ES
Beats
HBase
Forest
Cassandra
Azkaban
ganglia
DS
行为数据
广告数据
会员数据
销售数据
人工标注数据
供应链数据
第三方数据
线下数据
财务数据
用户反馈数据

相似影片推荐算法的一个简化版业务流向图，供大家与上面的模块对照参考：




数据仓库
Batch DM
用户模型
内容模型
协同过滤数据模型
深度学习数据模型
Speed DM
用户短期偏好模型
新入库内容特征模型
推荐算法平台
协同过滤ALS
内容/标签算法
深度学习算法
metadata
用户行为
收集
Kafka
Hadoop /Spark
ETL
Storm /Flink
Kafka
更新实时兴趣
推荐结果push到前端
Web Server
Redis
Couch Base
前端推荐库

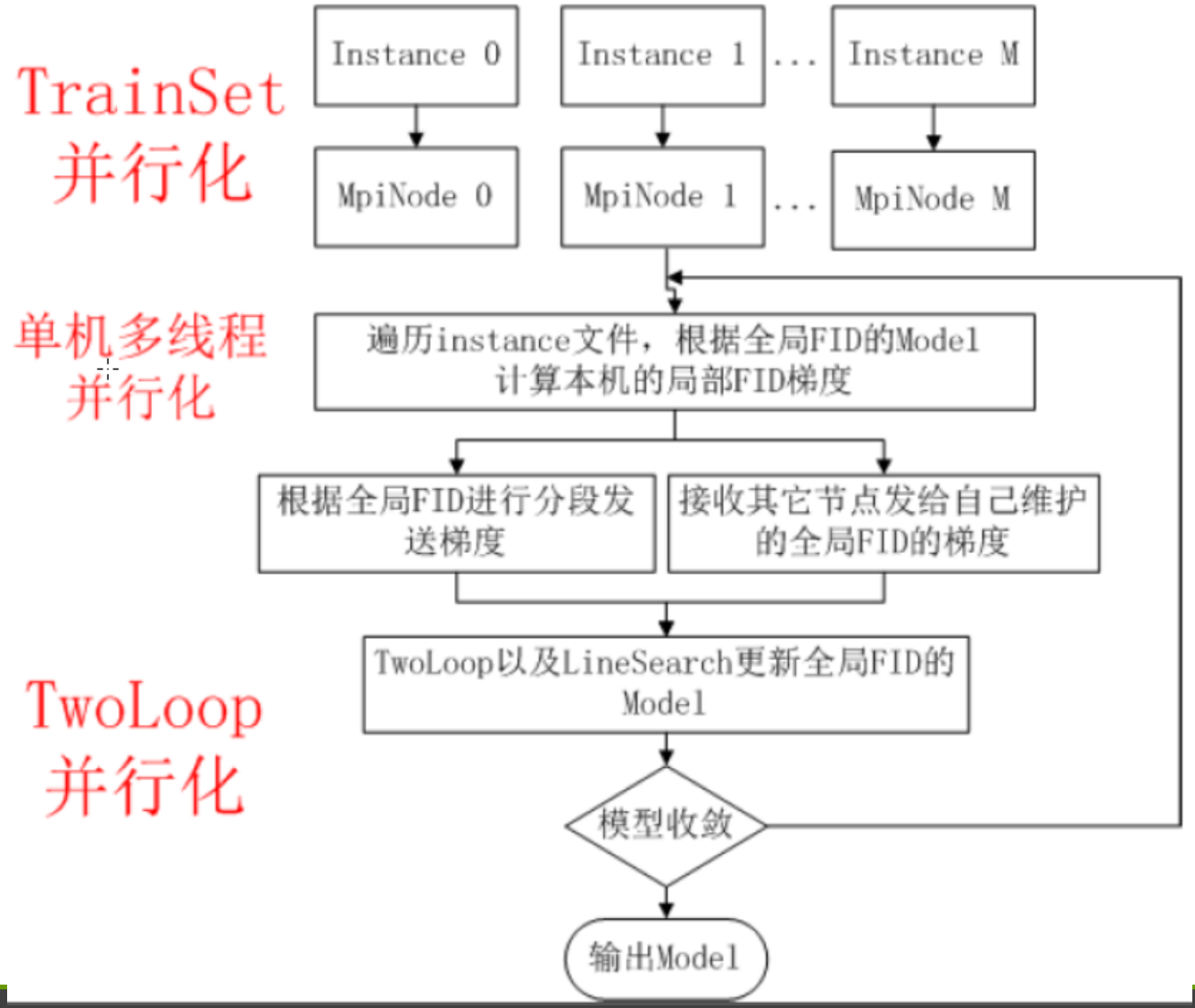
大数据处理平台

- 大规模MapReduce+HDFS集群
  - 4000+节点，80PB+存储
  - 搜索广告点击率预估
    - 原始日志分析：10T+
    - 特征提取：吞吐50T
    - 训练数据：20T
- 基于MPI的机器学习算法平台
  - 500+台机器（单机24 CPUs, 96G内存）
  - 自动任务调度和监控系统，部署多种算法，如LR，PLSA，LDA，SVM，GBDT等
  - 搜索广告点击率预估
    - 十亿级特征，百亿级训练样例
    - 运算时间：~2小时

MPI与Map-Reduce

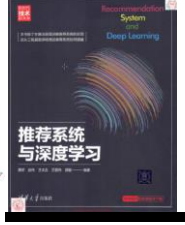
- MPI(Message Passing Interface)：消息传递接口
  - 消息传递函数库的标准规范
  - 通过在进程间传递消息完成数据交换，如Send, Recv
  - 程序员可以深入控制数据交互
- MPI VS. Map-Reduce模型
  - MPI：适合逻辑复杂的迭代运算，如机器学习算法
  - MR：适合计算独立，迭代少的任务

基于MPI的并行LR训练

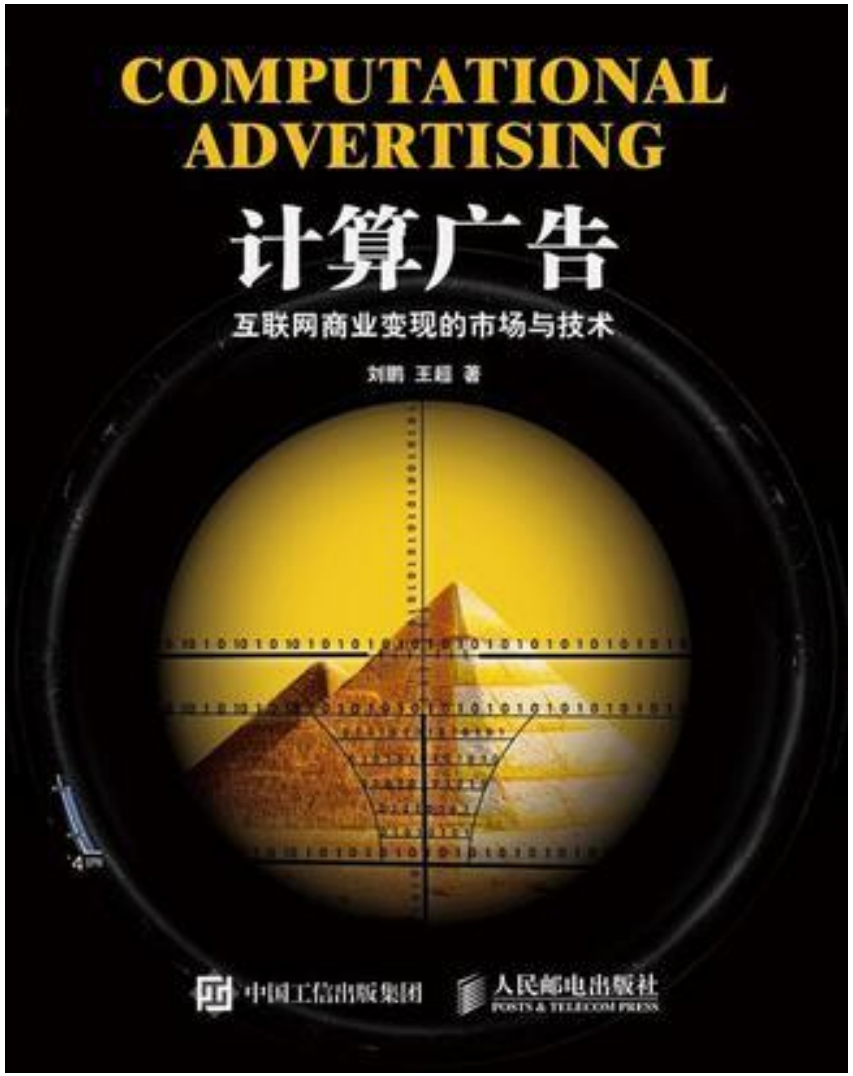



推荐系统实践

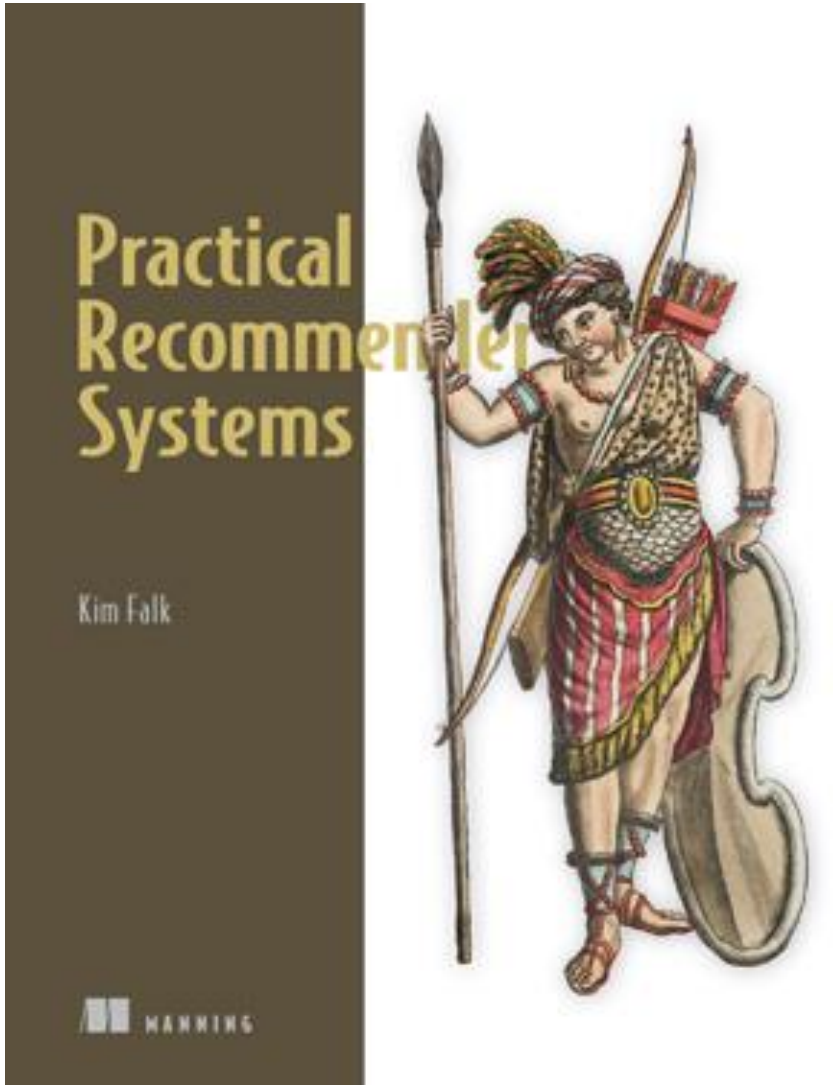

推荐系统
算法实践


推荐系统
与深度学习


Broadview
深度学习推荐系统
王喆 / 编著
中国工信出版集团


COMPUTATIONAL
ADVERTISING
计算广告
互联网商业变现的市场与技术
刘鹏 王超 著
中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS


异步图书
计算广告
互联网商业变现的市场与技术
（第2版）
刘鹏 王超 著
COMPUTATIONAL ADVERTISING
SECOND EDITION
中国工信出版集团
人民邮电出版社



- Practical Recommender Systems
- Kim Falk
- January 2019