

Recommender System for Real Mobile Applications: Two Case Studies

Big data vs. small data & Cloud vs. terminal

Zhenhua Dong, Huawei Noah's Ark Lab.

Content

- **Overview of recommender system**
- Case study 1: App recommender system in Android market
- Case study 2: Next App suggestion in mobile phone

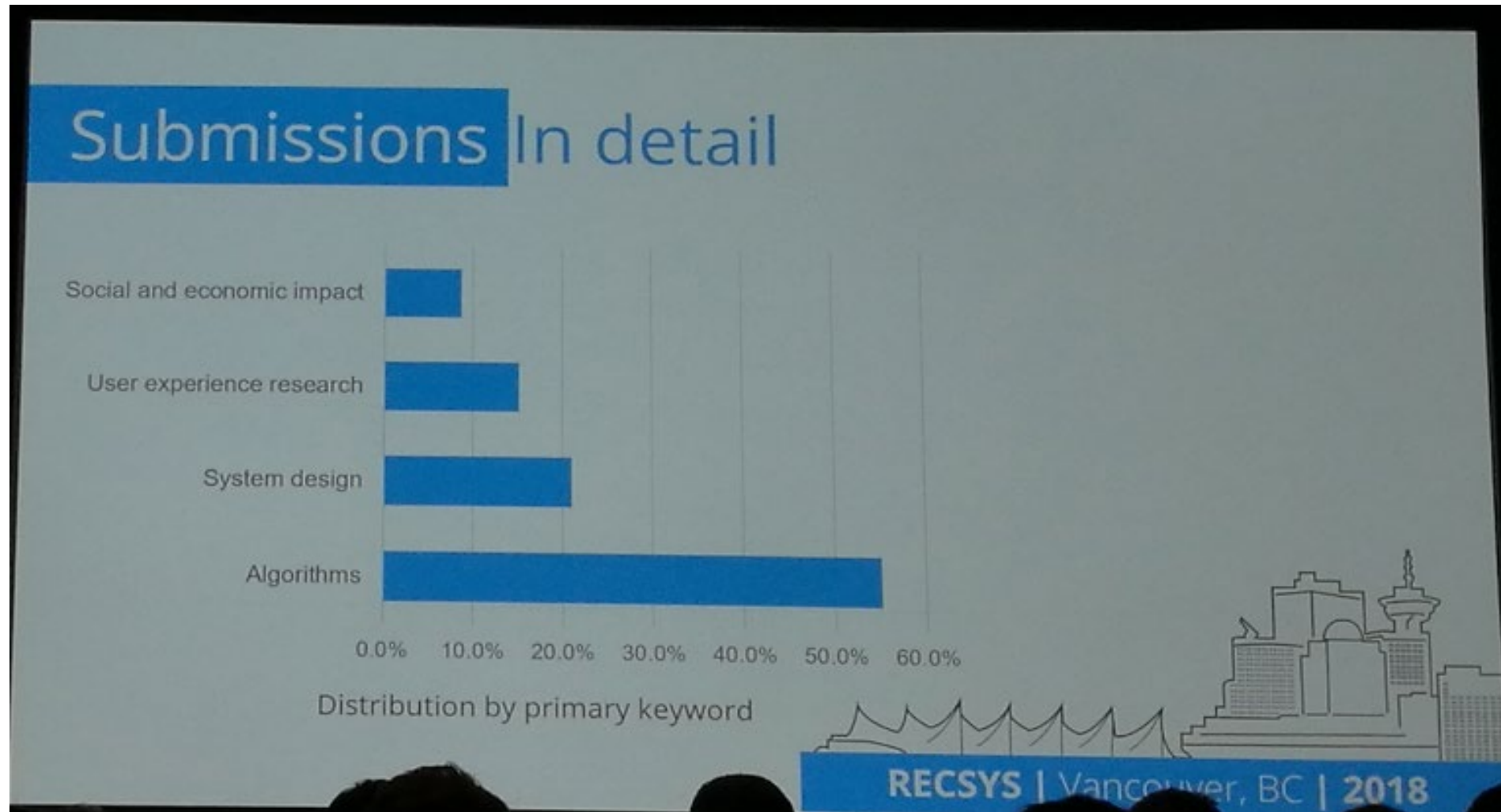
Brief history of recommender system research

- 1992, Information filtering and information retrieval: two sides of the same coin, CACM 1992.
- 1994, GroupLens: news recommendation system based on collaborative filtering technologies. “GroupLens: An Open Architecture for Collaborative Filtering of Netnews”, CSCW 1994.
- 1996, Net perceptions, Inc. was founded, which may be the first company focus on recommender system, Amazon was their customers.
- 1997, MovieLens: non-commercial and personalized movie recommendations for academic research. The MovieLens data set is the most popular data set for recommender system research.

- 2000, SVD model was proposed to reduce the dimensionality of user-item-rating matrix data set, “Application of Dimensionality Reduction in Recommender System -- A Case Study”, KDD 2000.
- Before 2001, the collaborative filtering is the dominated recommendation technology: user based or item based collaborative filtering. “Item-based collaborative filtering recommendation algorithms”, WWW 2001.
- 2006-2009, Netflix Prize, the low rank model has been well studied, such as matrix factorization.
- 2007, the first ACM RecSys was held in UMN.

- 2010, Rendle proposed factorization machines (FM) model for CTR prediction.
- 2011, user centric recommender systems: more comprehensive metrics have been studied, such as diversity, serendipity, novelty, trust, transparency.
 - A user-centric evaluation framework for recommender systems, RecSys 2011
 - Recommender systems: from algorithms to user experience, UMUI 2012.
- Since 2015, Deep learning was applied in recommender system
 - Collaborative deep learning for recommender systems, KDD 2015
 - DeepFM: A Factorization-Machine based Neural Network for CTR Prediction, IJCAI2017.
- 2017, more than 40% paper about DL in RecSys2017
- 2018, reinforcement learning are used in recommender system

Research topics



Recommender system: the most successful and widely used technology



Music



Video



E-Commerce



News Feed



Social network



LBS



Advertising

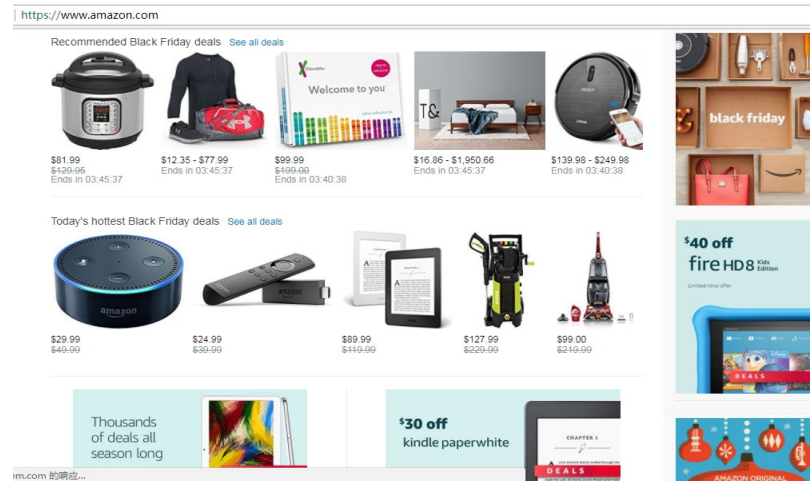


App distribution

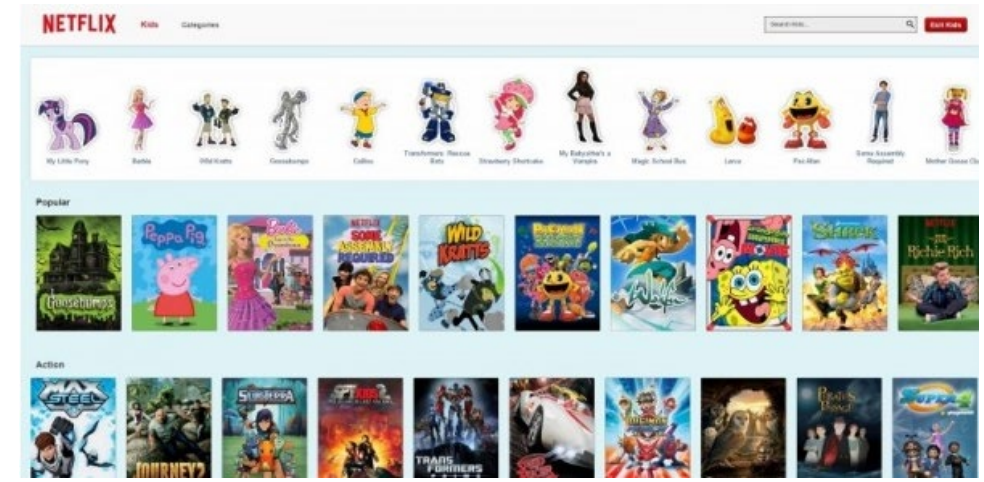


Short Video

Transfer the big data into the big value



"35% of Amazon.com's revenue is generated by its recommendation engine"



"80% of watched content is based on algorithmic recommendations"

“Personalized News recommender system helps ByteDance become decacorn company”



"In 2018, Google's ad revenue amounted to almost 116.3 billion US dollars"

Content

- Overview of recommender system
- **Case study 1: App recommender system in Android market**
- Case study 2: Next App suggestion in mobile phone

Overview of one Android App market

- One of the most popular Chinese Android application markets
- Preloaded on all one brand's mobile phones
- 300 million registered users, 2 million applications
- In each day:

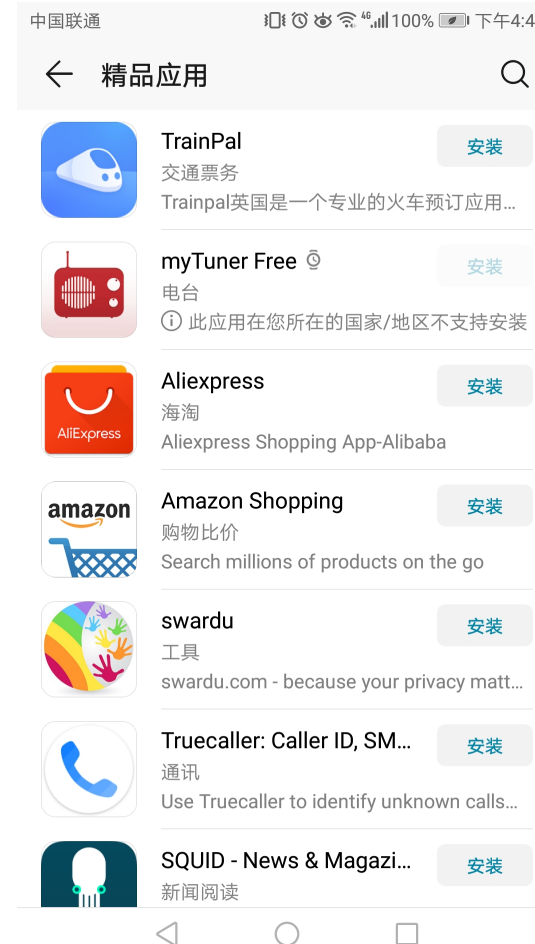
Description	Number
Visitors	XX million
Downloads (include updating)	XXX million
Search queries	XX million



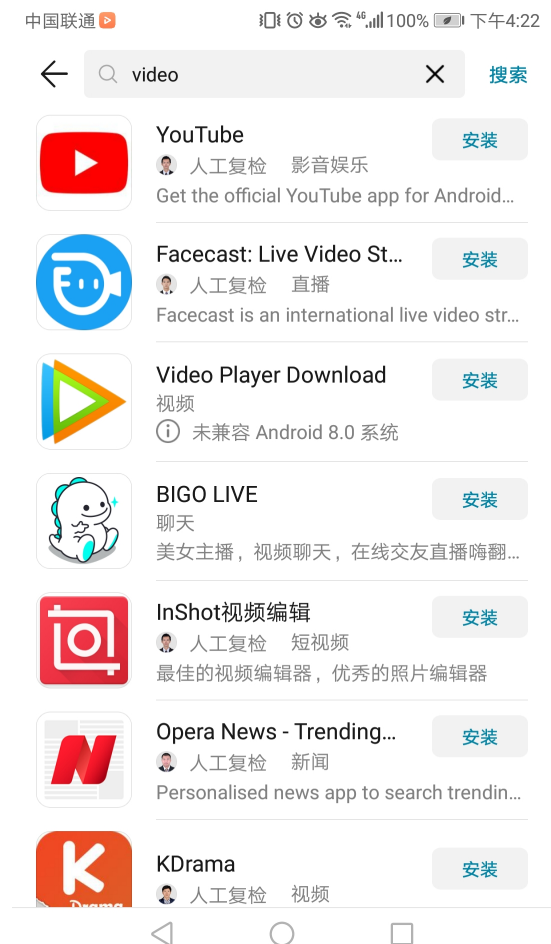
Sponsored App Ads recommendation

- Most important revenue source
- Ranked by: $CTR * CPD$ (cost per download)
- eCPM is the online metric
- Recommendation technologies:
 - ✓ Models: state-of-the-art ML models
 - ✓ Recall: ensemble methods, RT-update
 - ✓ Data: sampling, accurate exposure

App Ads in list

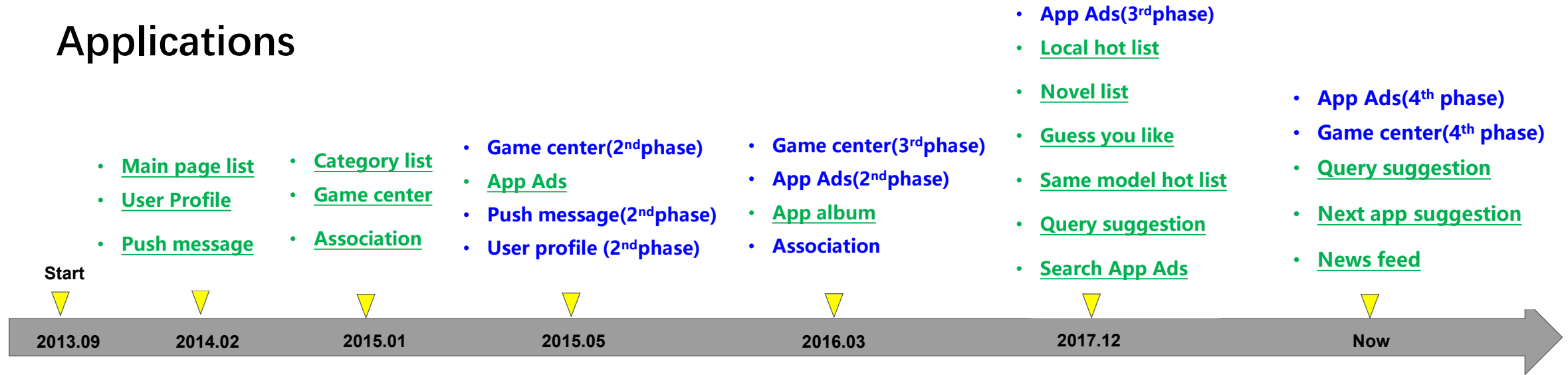


App Ads in search results



The technology evolution of App recommender system

Applications



Models




Architectures:



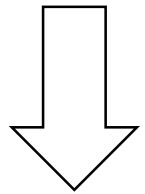
RecSys 1.0: High dimensional sparse linear model

- Model: logistic regression

$$p(\text{click}|\text{show}) = \text{Logistic}(w \cdot x)$$



$$P(y | x) = \frac{1}{1 + \exp(-yw^T x)}$$



Maximum
Likelihood

$$\min \lambda \|w\|^2 + \sum_{i=1}^n \log(1 + \exp(-y_i w^T x_i))$$

- Feature engineering

- ✓ Application

- ID: App ID, developer ID
- Attributes: category, tag , size , rate
- Semantic: name, description

- ✓ User

- ID: user ID
- Phone: screen size, phone type
- User behaviors

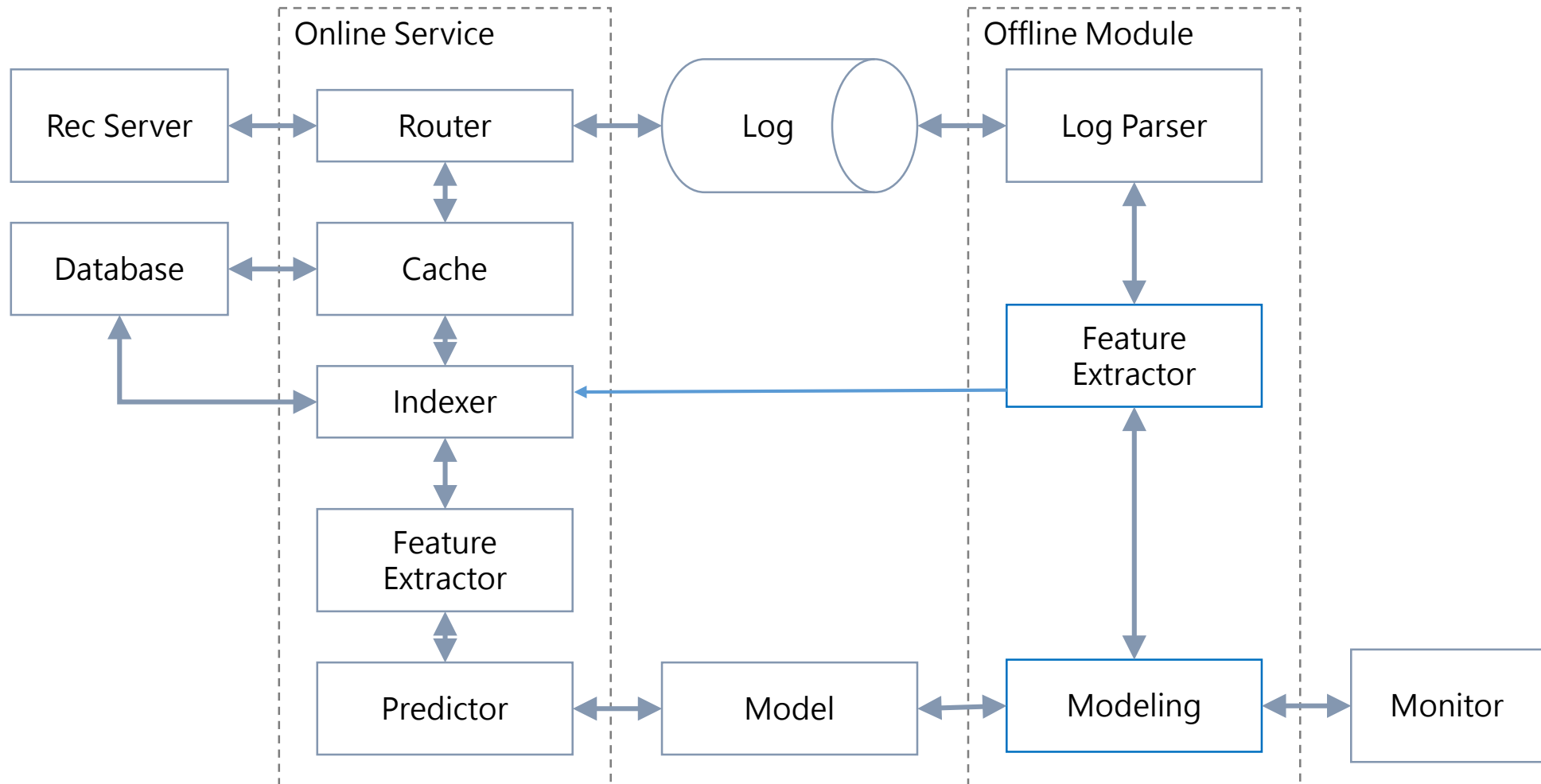
- ✓ Bias

- Position, source, list ID

- ✓ Combined features

- (history download App, current App)

2 layers-Architecture of RecSys 1.0



Performance: LR vs. user-based collaborative filtering

- #Download / #impression  70%+
- #Download / #user  70%+


RecSys 2.0: Real time technology

- Update model in real time
 - ✓ Logistic regression based on FTRL(follow-the-regularized-leader) optimization
 - ✓ Advantages: simple, theory, one pass update, online learning

Stochastic gradient

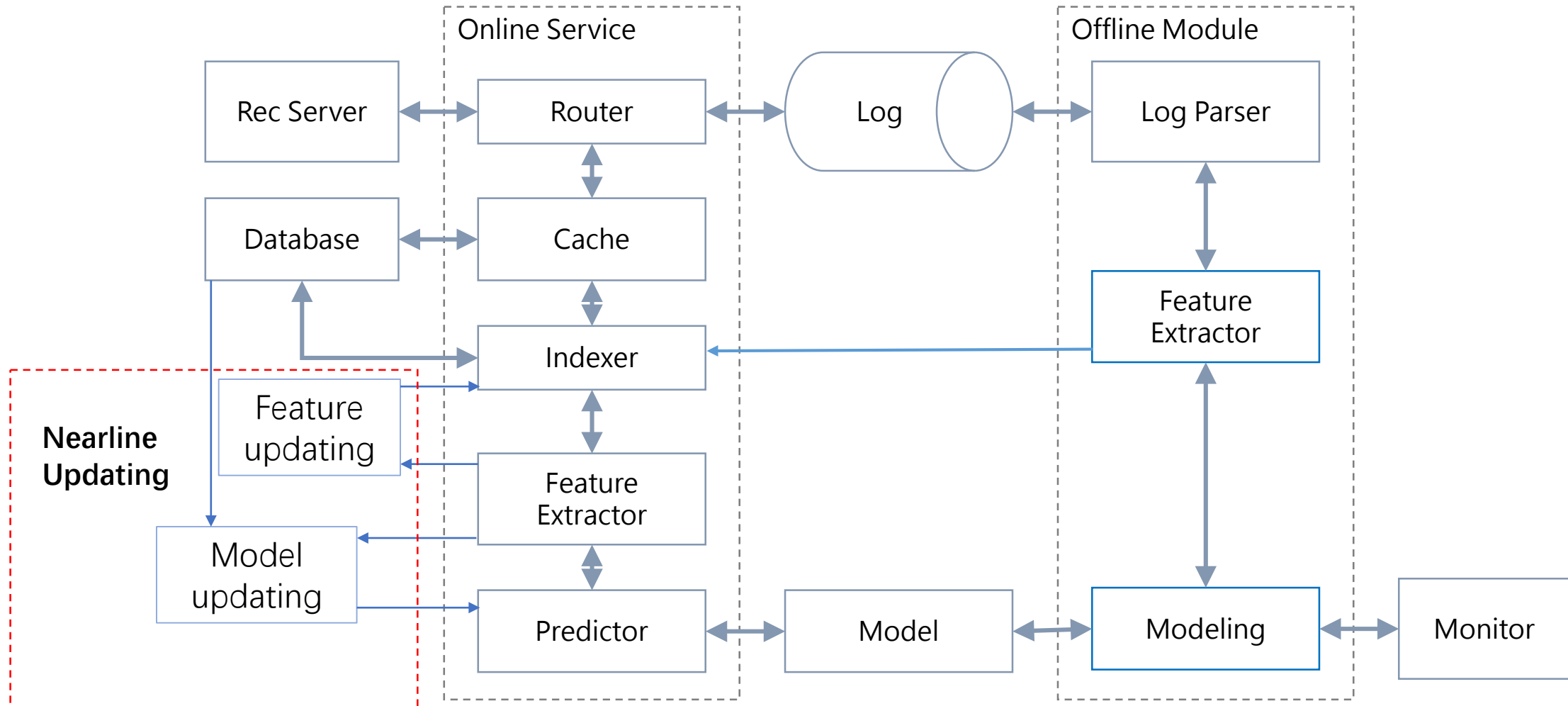
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{g}_t \quad \text{VS.} \quad \mathbf{w}_{t+1} = \arg \min_{\mathbf{w}} \left(\mathbf{g}_{1:t} \cdot \mathbf{w} + \frac{1}{2} \sum_{s=1}^t \sigma_s \|\mathbf{w} - \mathbf{w}_s\|_2^2 + \lambda_1 \|\mathbf{w}\|_1 \right)$$

Follow-the-regularized-leader

- Update feature in real time (more important)
 - ✓ Update user's instant behavior
 - ✓ Advantages: catch each user's interests immediately
- Real example:  *Shenzhen, Mate 20, download apps such as fitness, car price, VOA, Honor reading*

Round 1: results based user's initialized state	Round 2: Results after download Travel App2	Model weight of Travel App2 * current App	Round 3: Results after download Shopping App1	Model weight of Shopping App1 *current app
Housing App1	Travel App1	1.06	Express App	0.90
Joke App	Housing App1	0.50	Joke App	0.41
Shopping App1	Joke App	0.18	Housing App2	0.42
Travel App1	Shopping App1	0.19	Travel App1	-0.09
Car App	Shopping App2	0.35	Car App	0.54
Shopping App2	Housing App2	0.44	Car price App	0.31
Housing App2	Car App	0.40	Rent car App	0.48
Travel App2	Express App	0.37	Shopping App2	0.64
Express App	Car price App	0.36	Shopping App3	0.64
News App	Travel App3	0.72	Shopping App4	0.75

3 layers-Architecture of RecSys 2.0



Performance: Real time vs. Daily update

eCPM  **22%**

CTR  **27%**

CVR  **28%**

Income  **19%**

RecSys 3.0: automatic feature conjunction

Human feature engineering



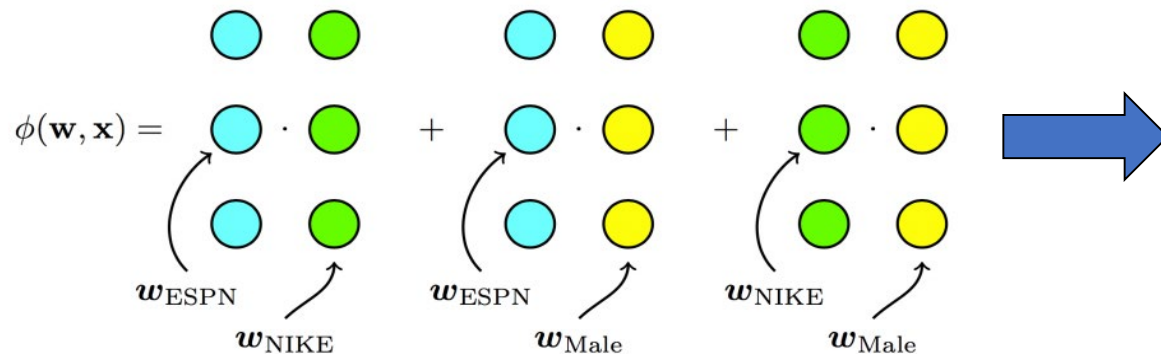
Automatic feature conjunction

- Field-aware Factorization Machine:
- Advantages:

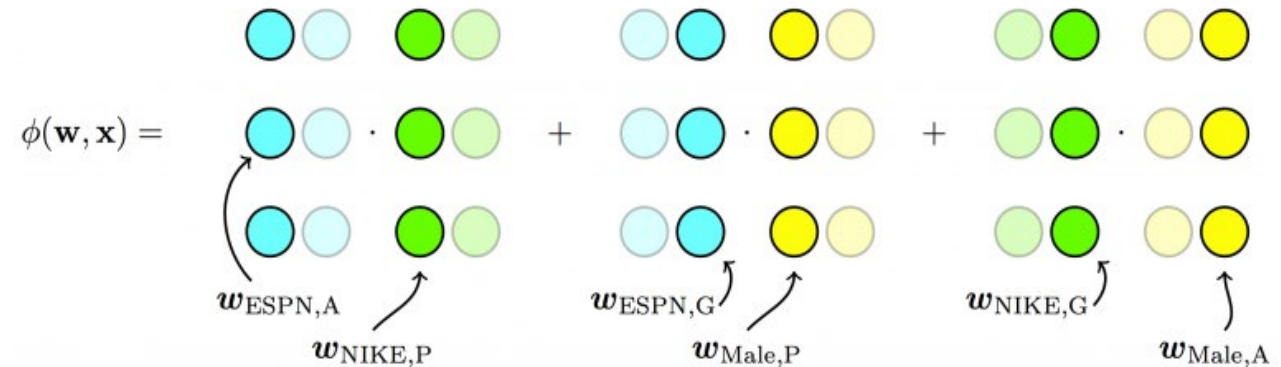
- ✓ Good at sparse and categorical data
- ✓ Automatic feature conjunction methods
- ✓ Feature space is much less than degree 2 polynomial
- ✓ Champion model of several CTR prediction contest

$$y_{\text{FFM}}(\mathbf{x}) = \text{sigmoid} \left(\underbrace{w_0 + \sum_{i=1}^N w_i x_i}_{\text{Logistic Regression}} + \underbrace{\sum_{i=1}^N \sum_{j=i+1}^N \langle \mathbf{v}_{i, \text{field}(j)}, \mathbf{v}_{j, \text{field}(i)} \rangle x_i x_j}_{\text{Field-aware field embedding}} \right)$$

Factorization Machine



Field-aware Factorization Machine

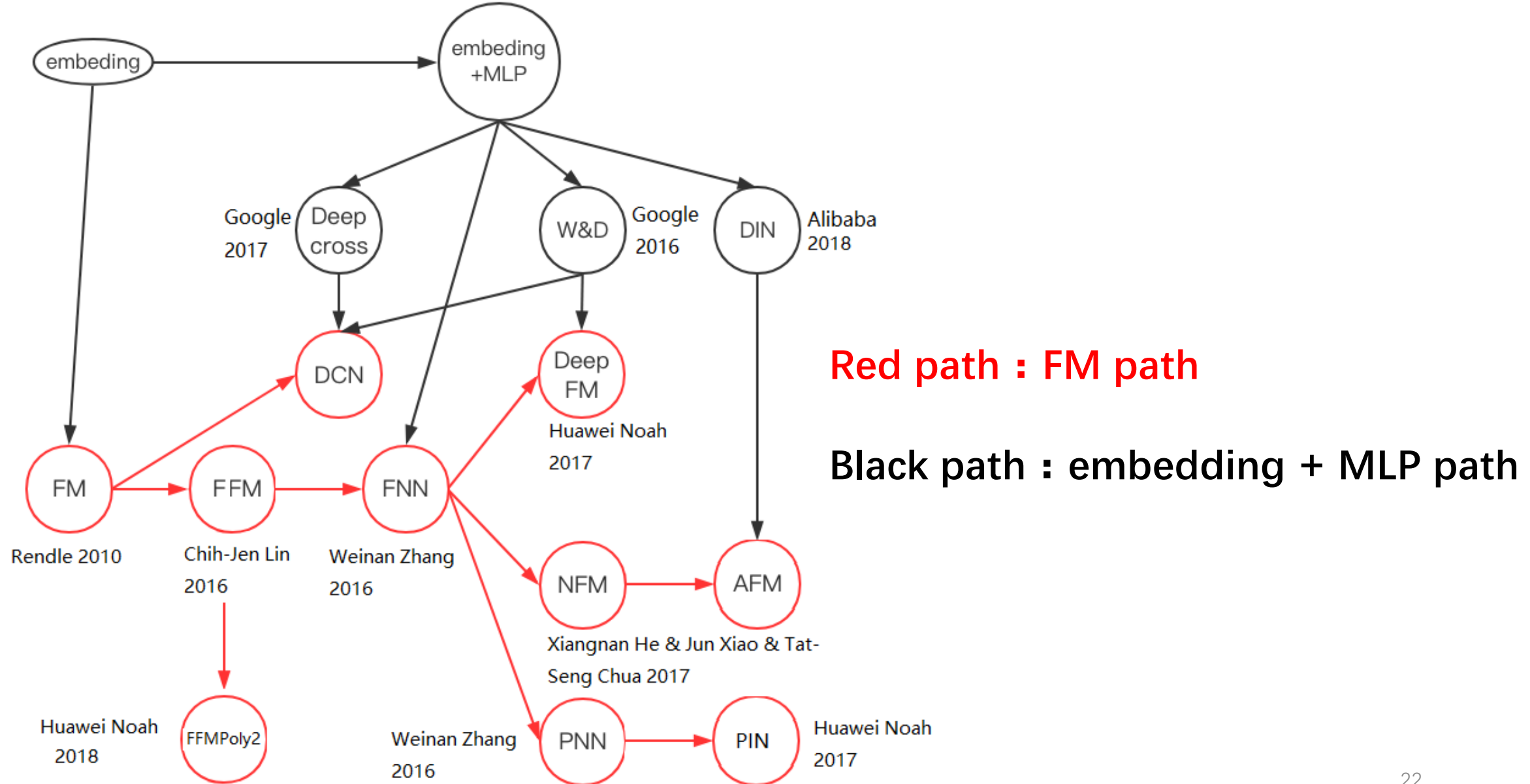


Performance: FFM vs. LR

eCPM  **6%**

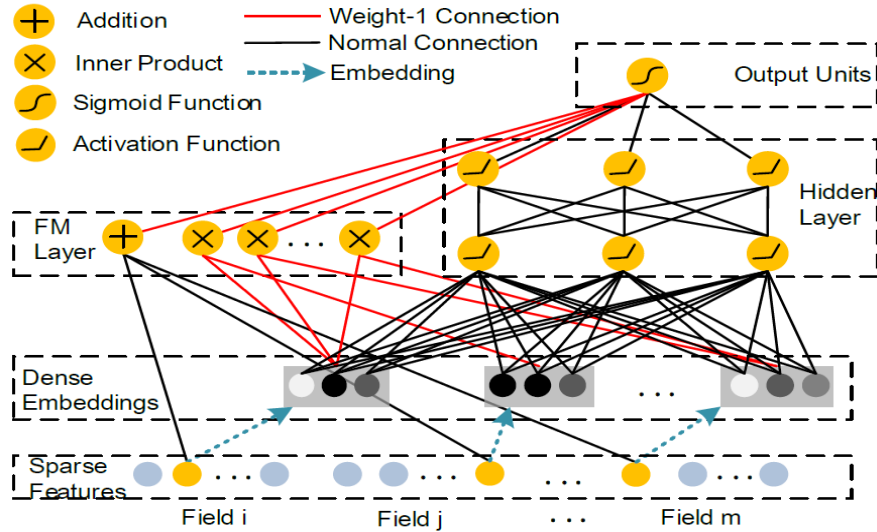
CTR  **12%**

Evolution of deep learning for recommender system

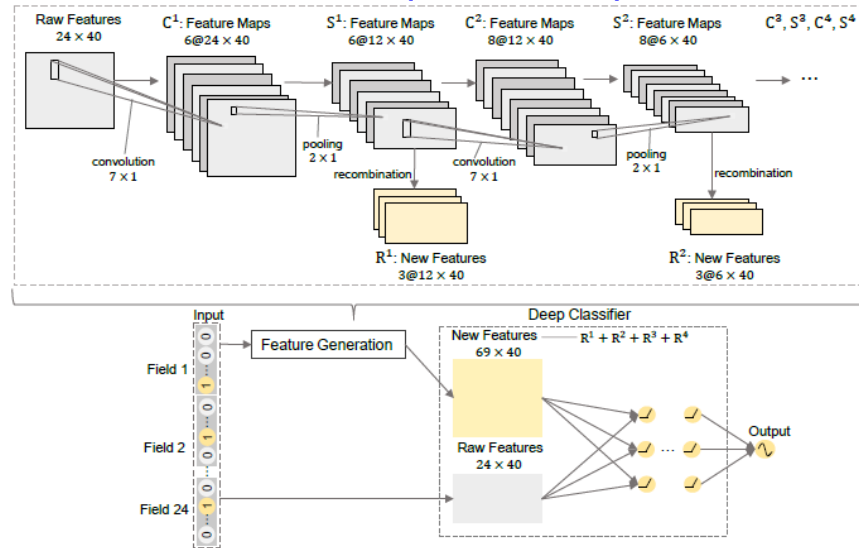


Deep learning for recommender system

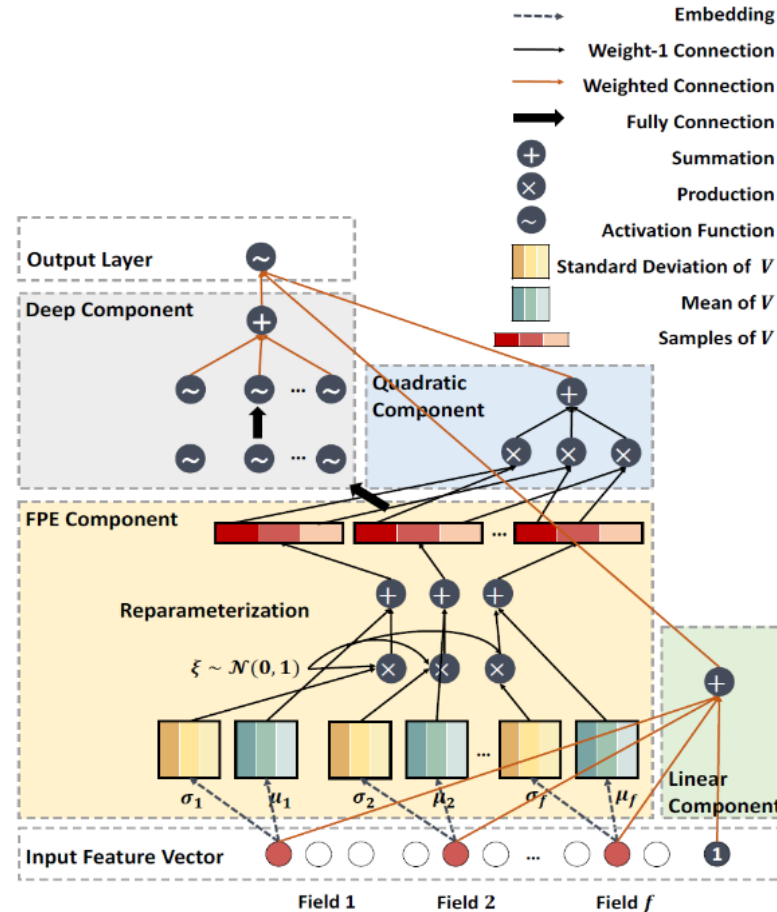
DeepFM (IJCAI2017)



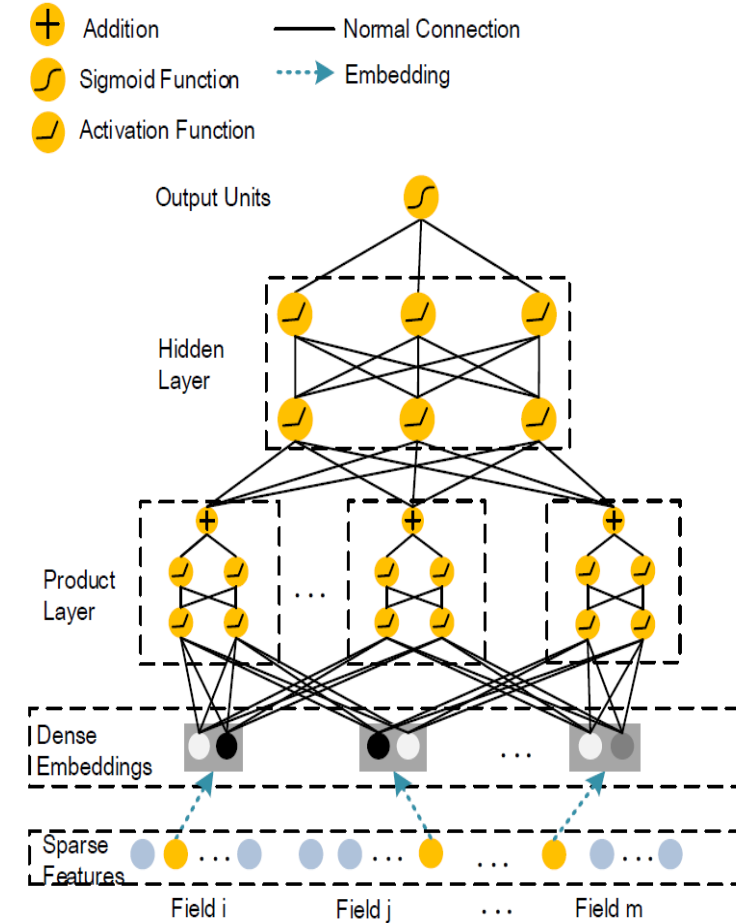
FGCNN (WWW 2019)



FPENN (RecSys 2018)



PIN (TOIS 2018)



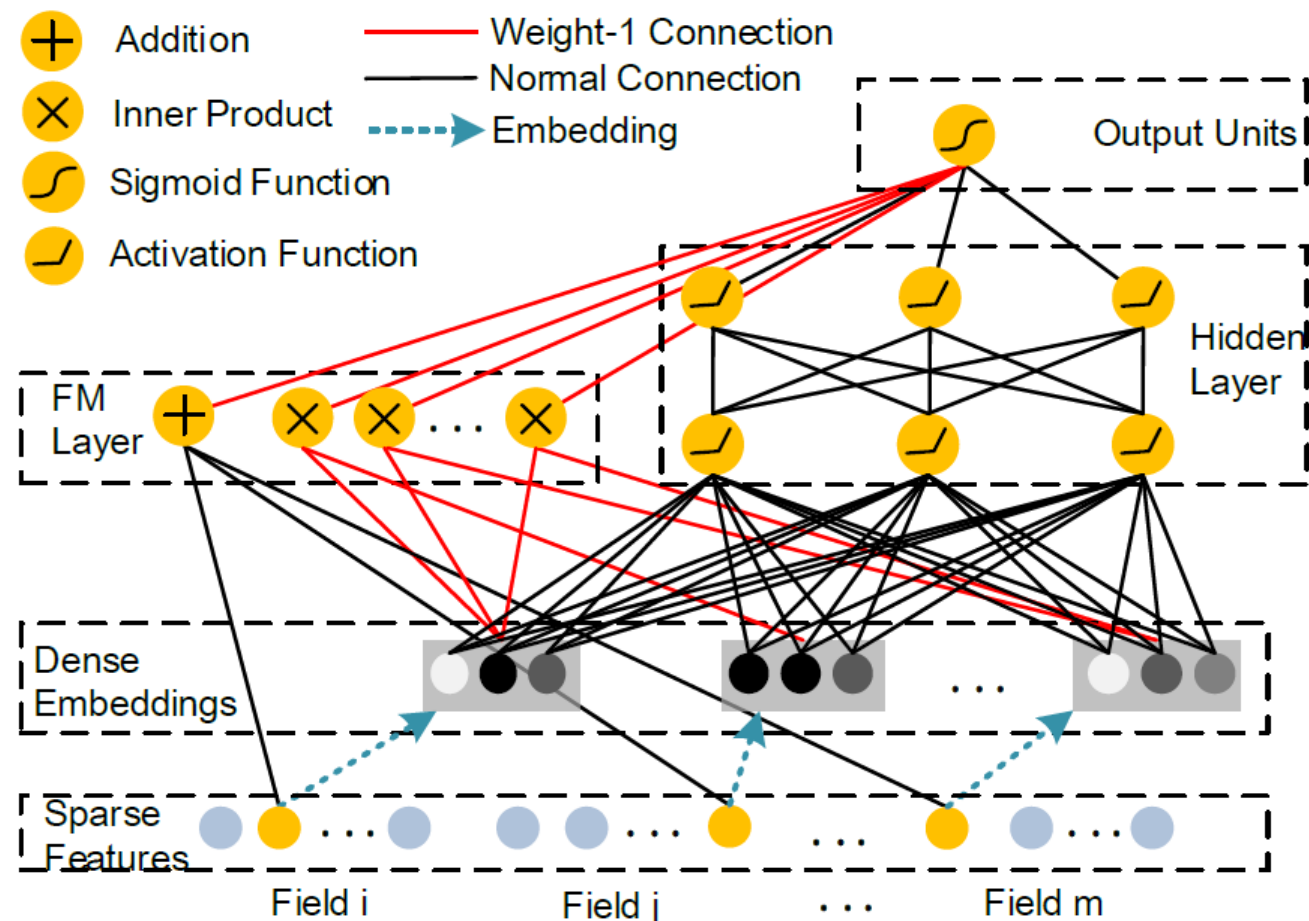
DeepFM

- Wide: FM automatically learns degree 2 feature combination
- Deep: DNN learns high dimension feature combination
- Sharing embedding: learn the embedding by both FM and DNN through back-propagation
- Advantages:

Table 1: comparison of deep models for CTR prediction

	Pre-training	High-order Feature	Low-order Feature	Feature Engineering
FNN	Yes	Yes	No	No
PNN{1,2,3}	No	Yes	No	No
Wide & Deep	No	Yes	Yes	Yes
DeepFM	No	Yes	Yes	No

Model architecture



PIN: product-network in network

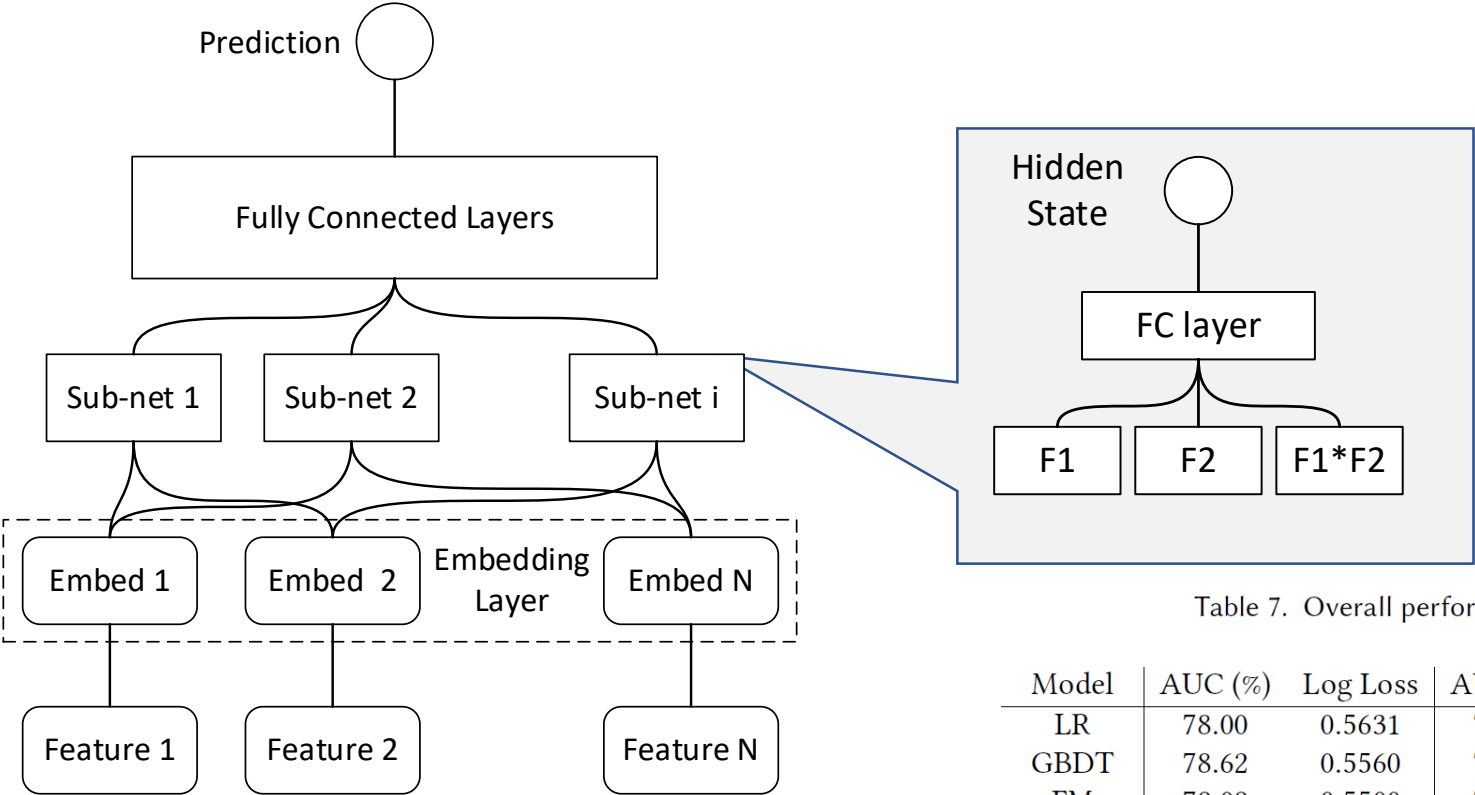


Table 7. Overall performance. (Left-Right: Criteo, Avazu, iPinYou, Huawei)

Model	AUC (%)	Log Loss	AUC (%)	Log Loss	AUC (%)	Log Loss	AUC (%)	Log Loss
LR	78.00	0.5631	76.76	0.3868	76.38	0.005691	86.40	0.02648
GBDT	78.62	0.5560	77.53	0.3824	76.90	0.005578	86.45	0.02656
FM	79.09	0.5500	77.93	0.3805	77.17	0.005595	86.78	0.02633
FFM	79.80	0.5438	78.31	0.3781	76.18	0.005695	87.04	0.02626
CCPM	79.55	0.5469	78.12	0.3800	77.53	0.005640	86.92	0.02633
FNN	79.87	0.5428	78.30	0.3778	77.82	0.005573	86.83	0.02629
AFM	79.13	0.5517	78.06	0.3794	77.71	<u>0.005562</u>	86.89	0.02649
DeepFM	<u>79.91</u>	<u>0.5423</u>	<u>78.36</u>	<u>0.3777</u>	<u>77.92</u>	0.005588	<u>87.15</u>	<u>0.02618</u>
KFM	79.85	0.5427	78.40	0.3775	76.90	0.005630	87.00	0.02624
NIFM	79.80	0.5437	78.13	0.3788	77.07	0.005607	87.16	0.02620
IPNN	80.13	0.5399	78.68	0.3757	78.17	0.005549	87.27	0.02617
KPNN	80.17	0.5394	78.71	0.3756	78.21	0.005563	87.28	0.02617
PIN	80.21	0.5390	78.72	0.3755	78.22	0.005547	87.30	0.02614

Content

- Overview of recommender system
- Case study 1: App recommender system in Android App market
- **Case study 2: Next App suggestion in mobile phone**

Overview of next App suggestion

- Objective: predict which services a user will use, and preload them on the top of leftmost screen
- Challenges:
 - ✓ Local RecSys: privacy issues, works even without network
 - ✓ Small data in term of sample # and feature dimensions
 - ✓ Need efficient methods for training and prediction
 - ✓ Cold start problem

Leftmost screen



Service candidates

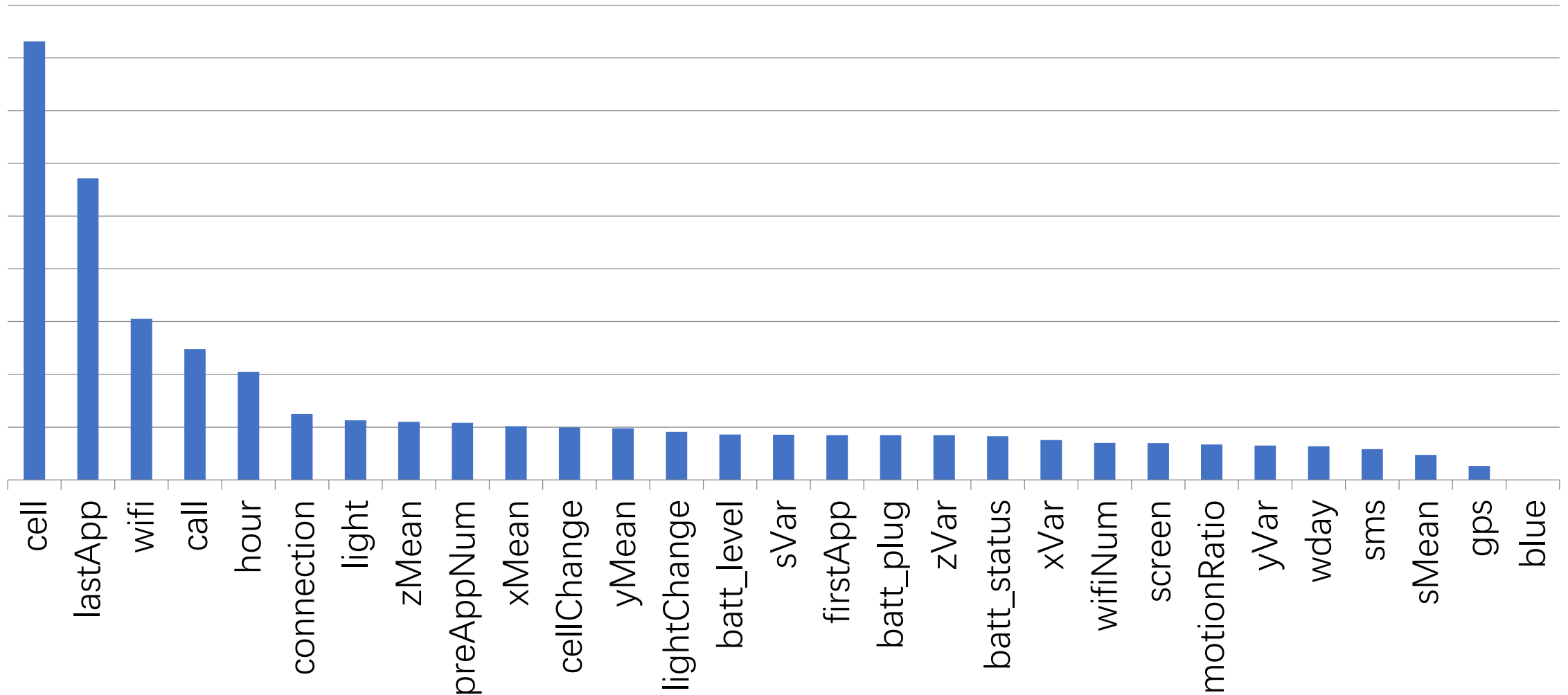


Feature engineering

Context Features
Previous used App
Cell
Battery
Network
GPS
WiFi
Accelerometer
Call/SMS log
Time
Light

- Discretization:
 - Previous App: One hot encoding
 - Popular Apps: Multi hot encoding
- Clustering:
 - GPS: distance
 - WiFi+time
- Transformation:
 - ✓ Accelerometer: mean, variance, energy, FFT
 - ✓ GPS: point of interest (POI)

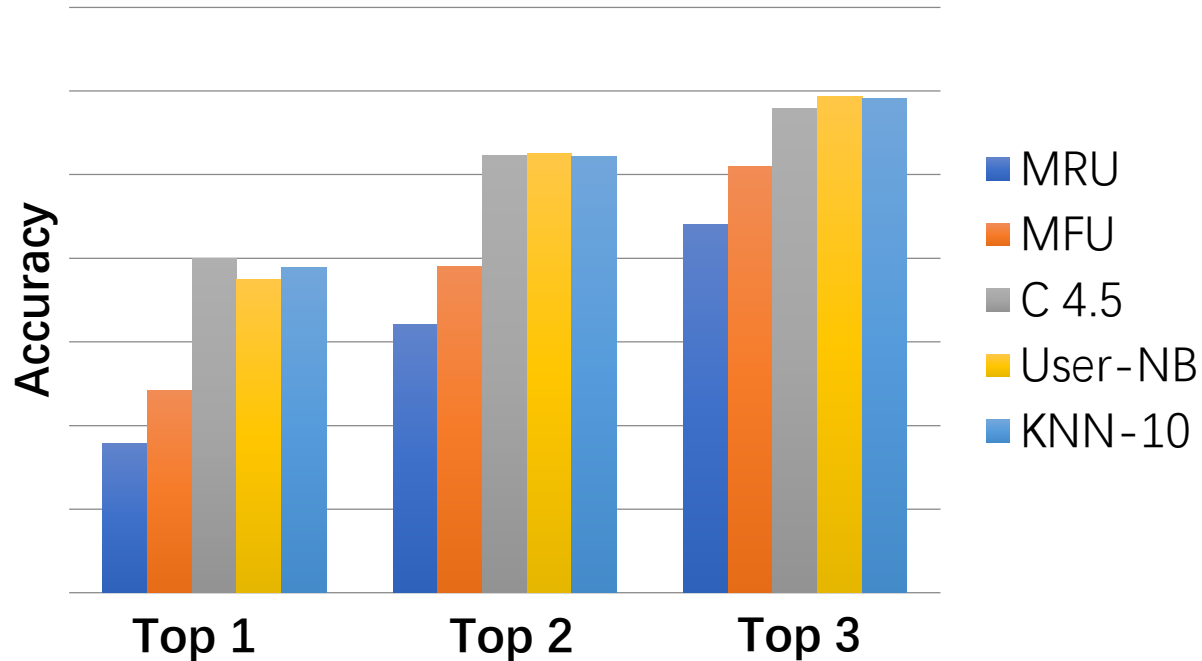
Feature importance (Information gain ratio)



Experiment: model selection

- Recruit 50 subjects with their consent
- Each subject had more than 400 services usage records in 30 consecutive days
- Collects data and generate features (see in last slide)
- Test on each user
 - ✓ Training data set: first $\frac{3}{4}$ records
 - ✓ Test data set: last $\frac{1}{4}$ records
- Model & Rules
 - ML models: Navie Bayes, C4.5, KNN
 - Rules: most recently usage (MRU), most frequency usage (MFU)

Avg. Accuracy

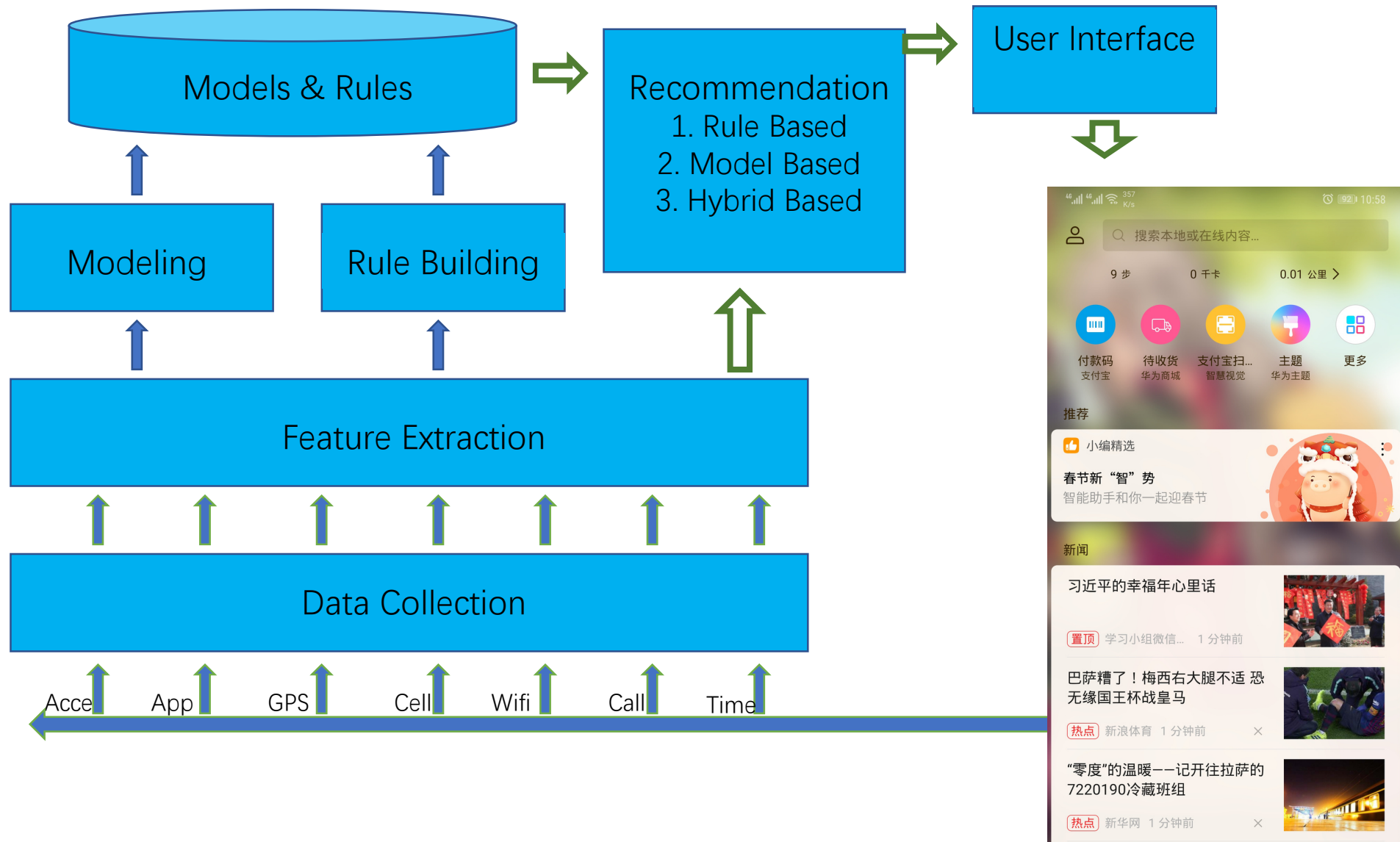


The number of the Best prediction model

TopN	MRU	MFU	C 4.5	User-NB	KNN-10
1	0	0	20	25	5
2	0	0	0	45	5
3	0	0	0	35	15
4	0	11	0	34	5
5	0	27	0	18	5
6	0	32	0	18	0
7	9	36	0	5	0
8	14	32	0	4	0

- Top 4 : NB performs best
- All the ML models have similar results
- MFU performs best above Top 4

Architecture

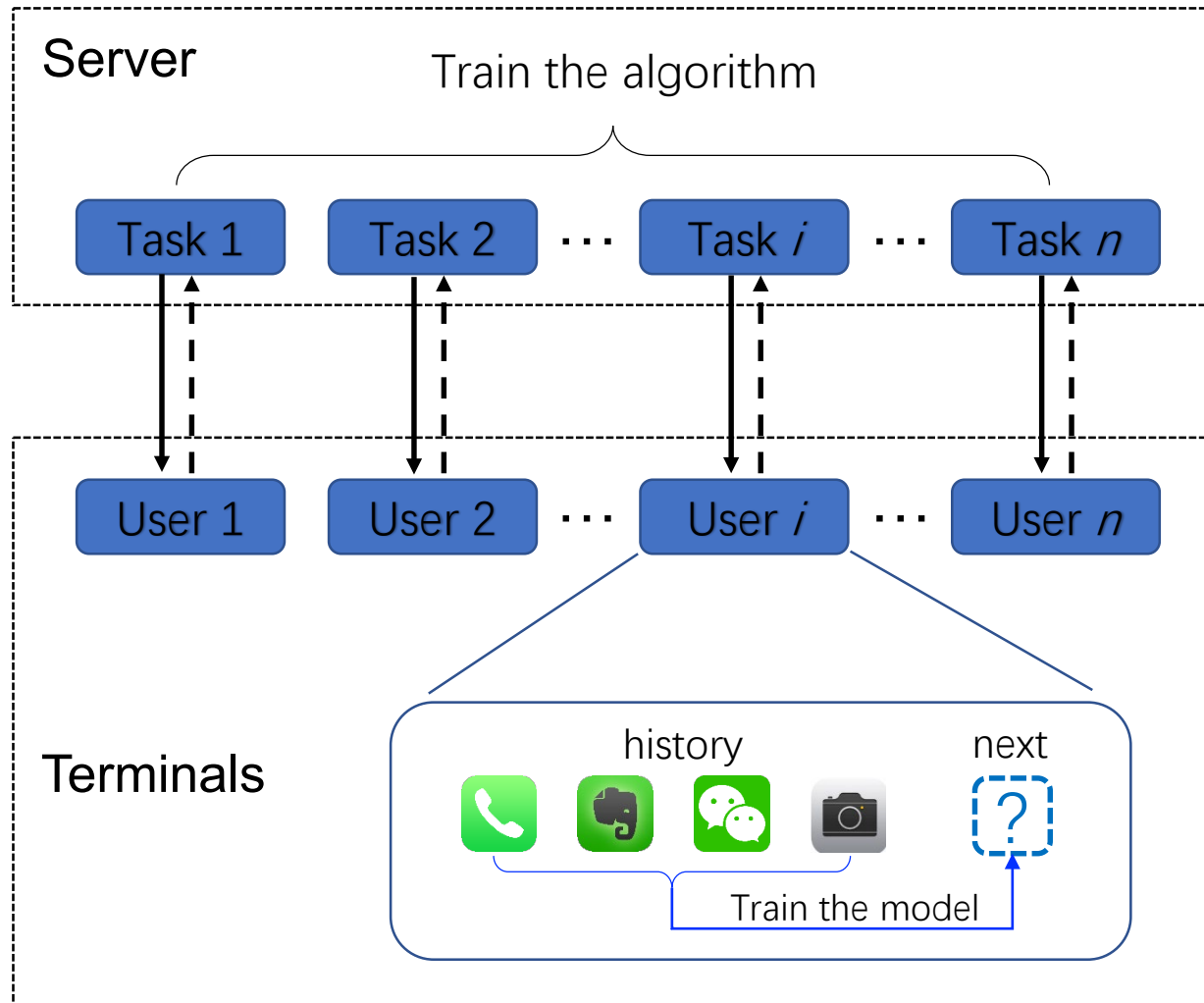


Cloud & terminal collaboration: federated meta learning

- Meta-learning is not just designed for few-shot learning, but more importantly, it provides an approach to learn shared knowledge within a group, e.g., smartphone users.
- Share data?
 - Privacy issues
- Share model?
 - (Possibly) unnecessarily large model
- Share algorithm.
 - Local model with local training
 - Trough [federated meta-learning](#)

Approaches	Sharing	Privacy	Small
Traditional learning	Data: sample	×	×
Federated learning	Model: CNN, LR, NB	√ ×	×
Federated meta-learning	Algorithm: SGD, LSTM	√	√

Example: next App suggestion



Server: train the algorithm using SGD with **test loss gradient**

→ algorithm
 - - -> loss gradient

Each terminal: train the model using the algorithm with **local data**

Table 6. Accuracies on the Production Dataset

			80% Support		5% Support	
			Top 1	Top 4	Top 1	Top 4
MIXED	NN-unified		76.72%	89.13%	66.47%	79.88%
SELF	MFU		42.92%	81.49%	42.18%	72.87%
	MRU		70.44%	81.43%	70.44%	81.43%
	NB		78.18%	92.57%	59.16%	72.83%
	LR	100 steps	58.30%	86.52%	52.53%	75.25%
		10000 steps	78.31%	93.70%	65.35%	77.11%
	NN	100 steps	57.20%	88.37%	49.89%	75.26%
		10000 steps	83.79%	94.56%	68.87%	77.66%
META	MAML + LR		47.69%	71.60%	46.75%	66.26%
	Meta-SGD + LR		81.70%	93.56%	72.32%	77.94%
	MAML + NN		83.87%	94.88%	73.08%	78.02%
	Meta-SGD + NN		86.23%	96.46%	72.98%	78.17%

Federated meta-learning for recommendation. arXiv preprint arXiv:1802.07876. 2018 Feb 22.

Take away:

(1) Real time is the industry standard technology for RecSys

- Update model: catch the trend of all users' requirements
- Update user feature: catch the change of one user's requirement

(2) Model selection

- Primary stages: LR is a good choice, simple, robust and easy to debug
- AutoML: select models, features, parameters automatically

(3) Recommender system with constrains

- Privacy constrain: GDPR in Europe → Federated learning, modeling in terminal
- Data quality constrain: data loss, noisy data → PU learning, data cleaning
- Computing resource constrain → Flexible automatic scaling system

(4) Data > feature > model

- Claudia Perlich: “40% of web click behaviors come from Bot, 36% of mobile phone click behaviors came from the users’ unintentionally clicks. The model learned from the above data can only predict the Bot’s behaviors well, not the user’s.”
- Always doubt the “data quality”: presumption of guilt
- Iterate the data cleaning loop:

(5) Beyond accuracy

- Joe Konstan: “CTR is just click behavior, why click? What is the decision mechanism behind it? We need to answer the 2 questions?” “Recommender system should be end-to-end systematic research, not just algorithm”
- User centric evaluation:



Thank you for your listening!