User-based real-time product recommendations leveraging deep learning using Analytics Zoo on Apache Spark and BigDL

Luyang Wang (Office Depot), Guoqiong Song (Intel), Jing (Nicole) Kong (Office Depot), Maneesha Bhalla(Office Depot)

Strata
DATA CONFERENCE

Agenda

- Office Depot Overview
- What is BigDL & Analytic Zoo?
- Recommendation Use Case
- Takeaways

Office DEPOT

Office Depot, Inc. (NASDAQ:ODP) is a leading B2B integrated distribution company providing business services and supplies, products and technology solutions through its fully integrated omni-channel platform of approximately 1,350 stores, online presence, and dedicated sales professionals and technicians to small, medium and enterprise businesses. Through its banner brands Office Depot®, OfficeMax®, CompuCom® and Grand&Toy®, the company offers its customers the tools and resources they need to focus on their passion of starting, growing and running their business.

Big Data Journey for Recommendation

Stage I :

We tried to build machine learning models to do product recommendation using Python/SAS/R

…

Challenge:

We can not process this amount of data on a single machine

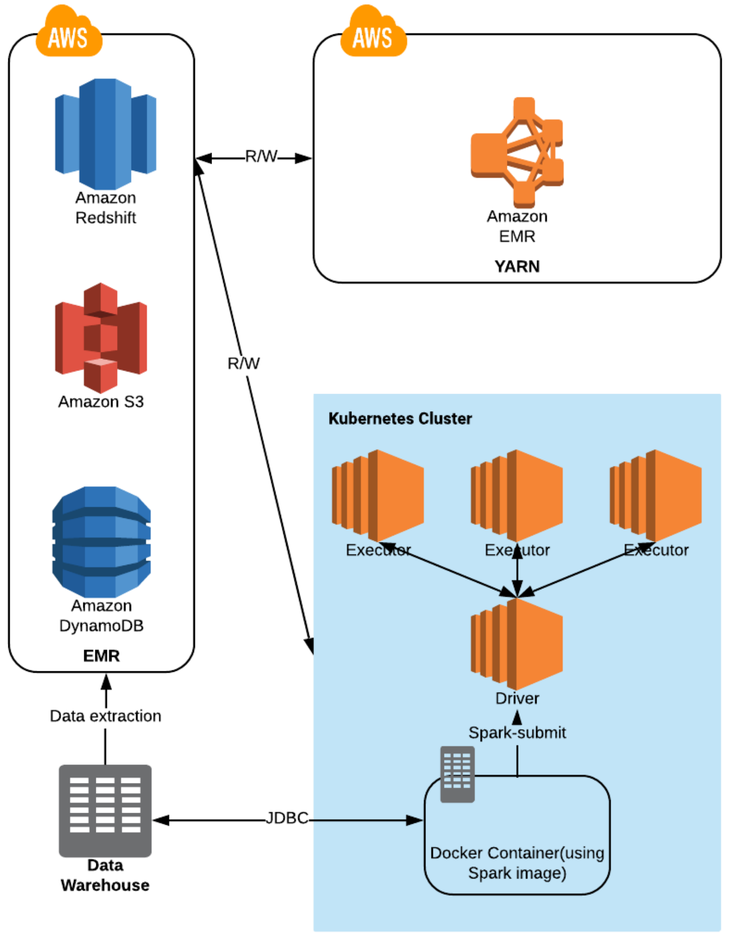
- Over 100,000,000 distinct sessions
- More than 300,000 active products selling online
- Training data often exceed 10G

Big Data Journey for Recommendation

Stage II :

We incorporated Spark and AWS cloud into our workflow

Challenge:
Deep learning libraries such as Tensorflow/Keras/PyTorch cannot run directly on Spark cluster

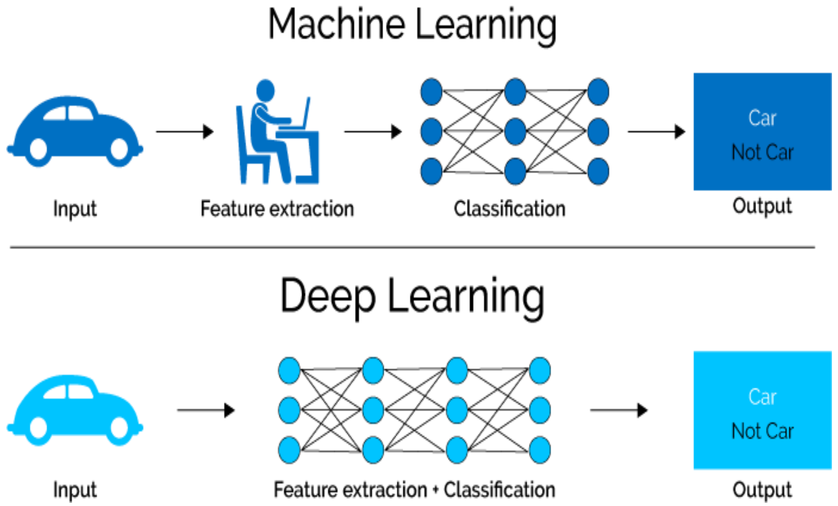


Big Data Journey for Recommendation

Stage III :

Why do we need deep learning ?

- Better performance with larger data
- Less manual feature engineering needed, feature generation and classification are tied within one system
- Easier to compute complex functions
- Flexibility to combine different architectures and output formats



AI on Apache Spark

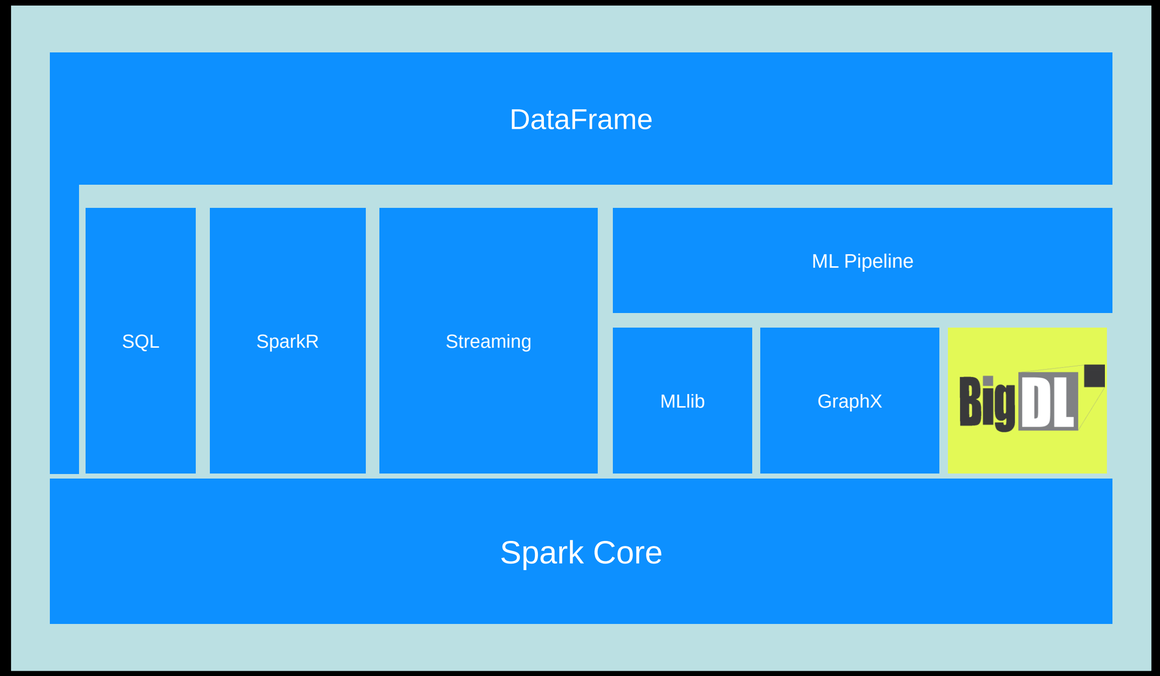
Distributed, High-Performance
Deep Learning Framework
for Apache Spark
https://github.com/intel-analytics/bigdl

Distributed TensoRflow, Keras and BigDL on Spark
Reference Use Cases, AI Models,
High-level APIs, Feature Engineering, etc.
https://github.com/intel-analytics/analytics-zoo

Unifying Analytics + AI on Apache Spark

BigDL: Bringing Deep Learning To Big Data Platform

- Distributed deep learning framework for Apache Spark*
- Make deep learning more accessible to big data users and data scientists
  - Write deep learning applications as standard Spark programs
  - Run on existing Spark/Hadoop clusters (no changes needed)
- Feature parity with popular deep learning frameworks
  - E.g., Caffe, Torch, Tensorflow, etc.
- High performance (on CPU)
  - Powered by Intel MKL and multi-threaded programming
- Efficient scale-out
  - Leveraging Spark for distributed training & inference



*Other names and brands may be claimed as the property of others

Analytics Zoo

Unified Analytics + AI Platform for Big Data

Distributed TensorFlow, Keras and BigDL on Spark

Reference Use Cases	• Anomaly detection, sentiment analysis, fraud detection, image generation, chatbot, etc.
Built-In Deep Learning Models	• Image classification, object detection, text classification, text matching, recommendations, sequence-to-sequence, anomaly detection, etc.
Feature Engineering	Feature transformations for • Image, text, 3D imaging, time series, speech, etc.
High-Level Pipeline APIs	• Distributed TensorFlow and Keras on Spark • Native support for transfer learning, Spark DataFrame and ML Pipelines • Model serving API for model serving/inference pipelines
Backbends	Spark, TensorFlow, Keras, BigDL, OpenVINO, MKL-DNN, etc.

Analytics Zoo

Build end-to-end deep learning applications for big data

- Distributed TensorFlow on Spark
- Keras-style APIs (with autograd & transfer learning support)
- nnframes: native DL support for Spark DataFrames and ML Pipelines
- Built-in feature engineering operations for data preprocessing

Productionize deep learning applications for big data at scale

- POJO model serving APIs (w/ OpenVINO support)
- Support Web Services, Spark, Storm, Flink, Kafka, etc.

Out-of-the-box solutions

- Built-in deep learning models and reference use cases

What Can you do with Analytic Zoo?

Anomaly Detection

- Using LSTM network to detect anomalies in time series data

Fraud Detection

- Using feed-forward neural network to detect frauds in credit card transaction data

Recommendation

- Use Analytics Zoo Recommendation API (i.e., Neural Collaborative Filtering, Wide and Deep Learning) for recommendations on data with explicit feedback.

Sentiment Analysis

- Sentiment analysis using neural network models (e.g. CNN, LSTM, GRU, Bi-LSTM)

Variational Autoencoder (VAE)

- Use VAE to generate faces and digital numbers

https://github.com/intel-analytics/analytics-zoo/tree/master/apps

Building and Deploying with BigDL/Analytics Zoo

http://software.intel.com/bigdl/build

Not a Full List

*Other names and brands may be claimed as the property of others

End to End Deep Learning Examples

Consumer

[Sentiment Analysis]

[Image Similarity And Search]

[Image Transfer Learning]

[Image Generation]

Health

[3D Image Support]

Finance

[Fraud Detection]

[Anomaly Detection]

Retail

Recommendation NCF

[Wide n Deep]

Manufacturing

[Object Detection]

Infrastructure

[Tensorflow support]

[Low latency serving]

Follow links above to sample code for various Deep Learning use cases

Online Real-time Recommendation

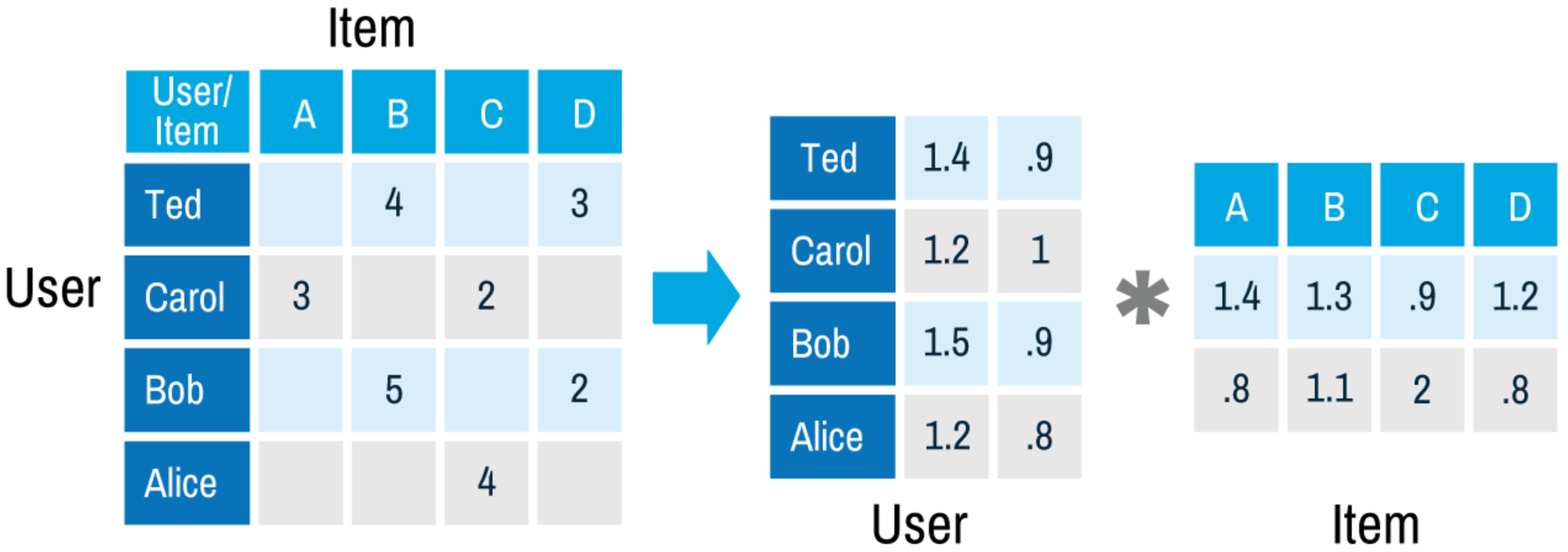
Goal:

A system able to recommend products to online customer from an active product backlog of more than 300,000 products

Challenge:

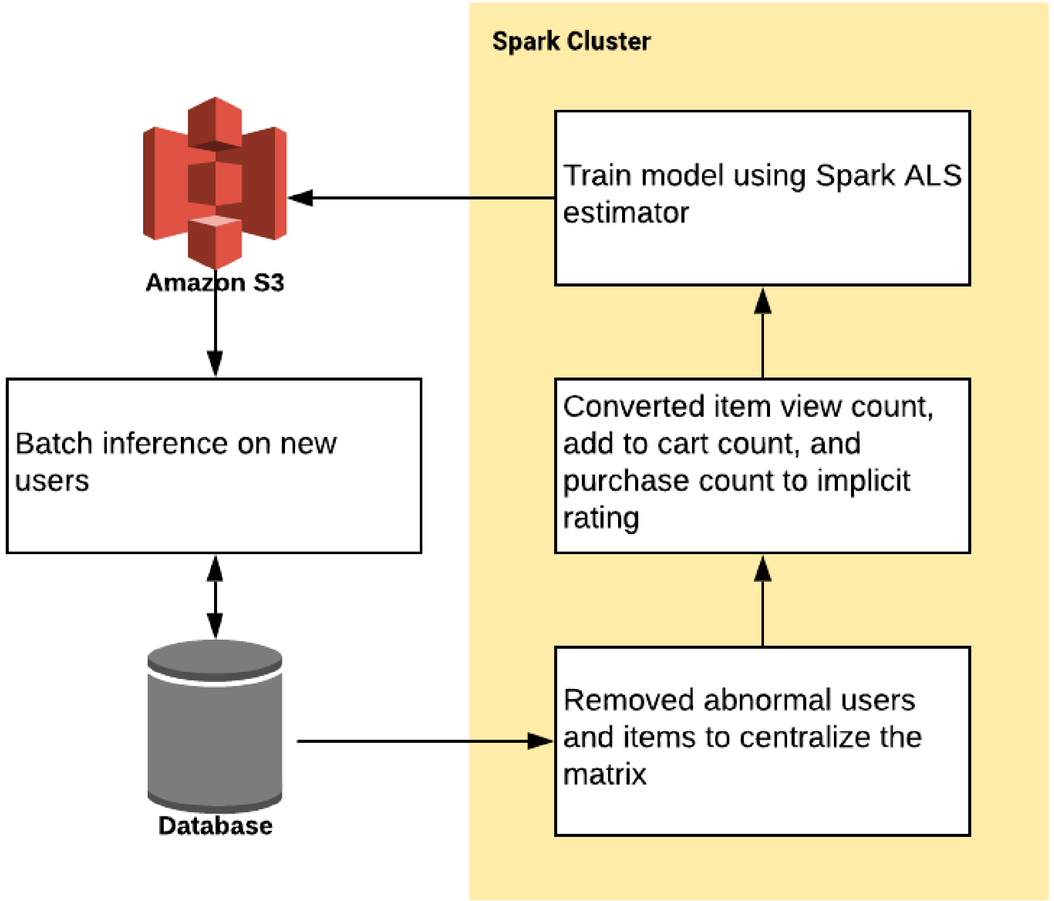
- Large amount of user and item data to process
- User's purchase intent can change frequently
- Deploy model to production with high throughput and low latency

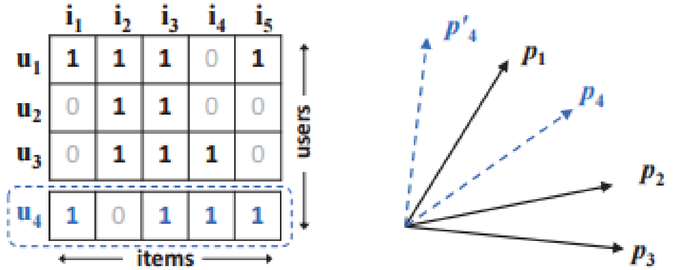
Collaborative Filtering (ALS)



The Collaborative filtering approach work by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrices. Spark ALS implementation runs matrix factorization in a parallel fashion and therefore has a pretty good scalability and sparseness dealing with large datasets

Collaborative Filtering (ALS)





(a) user–item matrix (b) user latent space

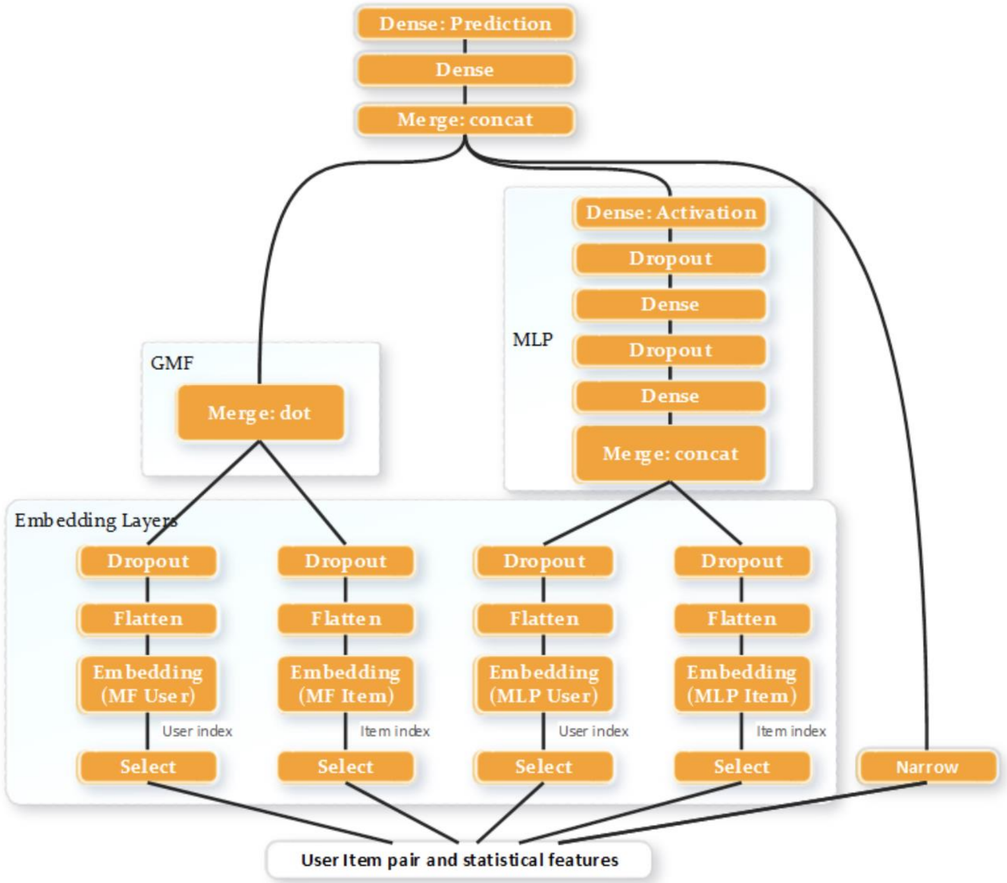
Figure 1: An example illustrates MF's limitation. From data matrix (a), u_4 is most similar to u_1, followed by u_3, and lastly u_2. However in the latent space (b), placing p_4 closest to p_1 makes p_4 closer to p_2 than p_3, incurring a large ranking loss.

Limitations:

- Not able to do incremental training
- Not able to capture the latest purchase intent
- Data sparse problem

…

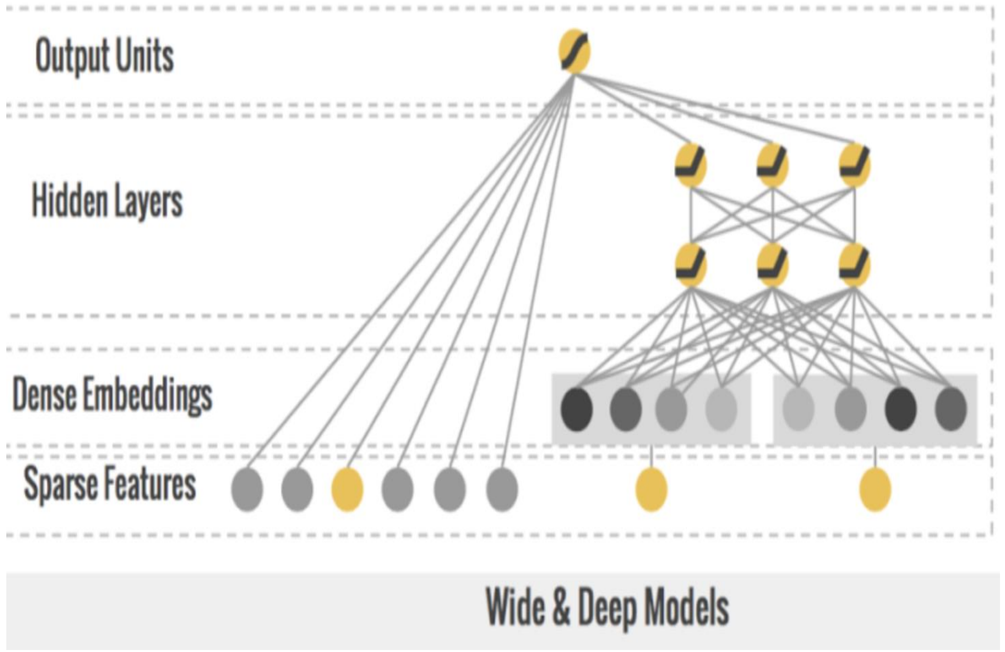
Neural Collaborative Filtering



NCF stimulates matrix factorization using DNN approach and is severed as a guideline for deep learning methods for recommendation services. It combines MF with MLP to model user-item interactions. With the same user item input, NCF outperforms traditional shallow collaborative filtering models on our dataset.

```
val ncf = NeuralCF[Float] (userCount, itemCount, numClasses,
userEmbed, itemEmbed, mfEmbed)

val optimizer = Optimizer(model = ncf, sampleRDD = trainRdds,
  criterion = ClassNLLCriterion[Float](), batchSize = batchSize)

val ncf_model = optimizer
  .setOptimMethod(new Adam[Float](learningRate = 1e-2,
learningRateDecay = 1e-5))
  .setEndWhen(Trigger.maxEpoch(maxEpoch))
  .optimize()
```

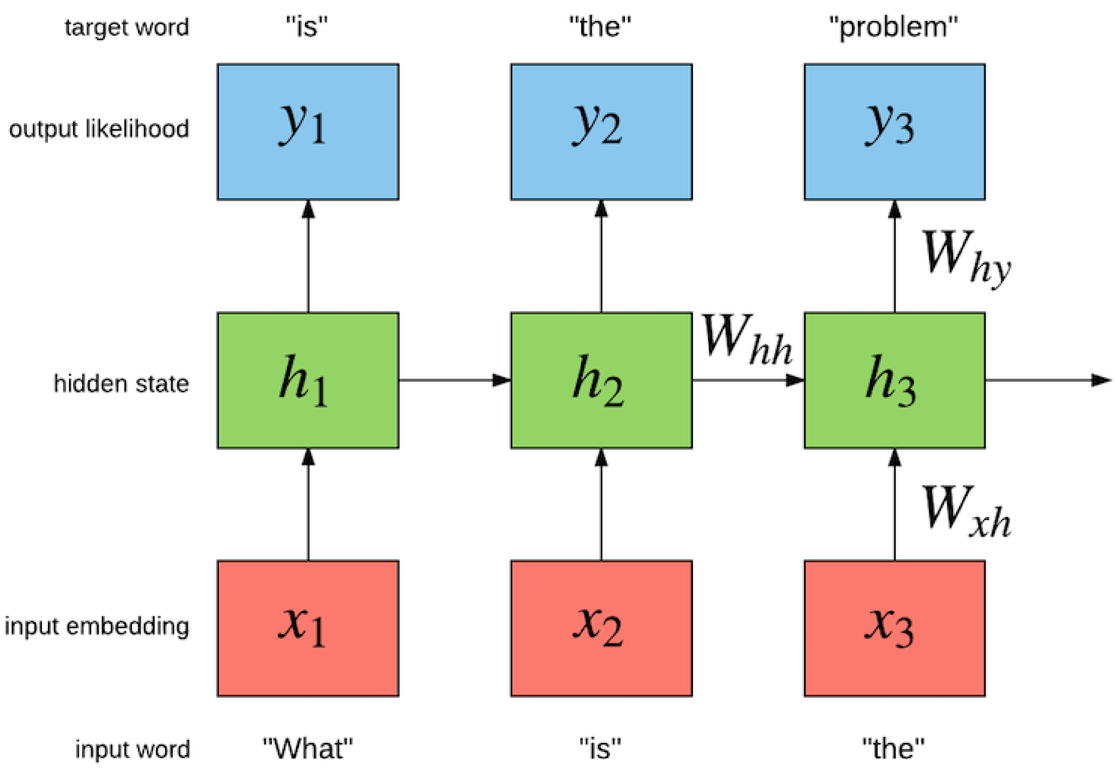
Wide and Deep Recommender

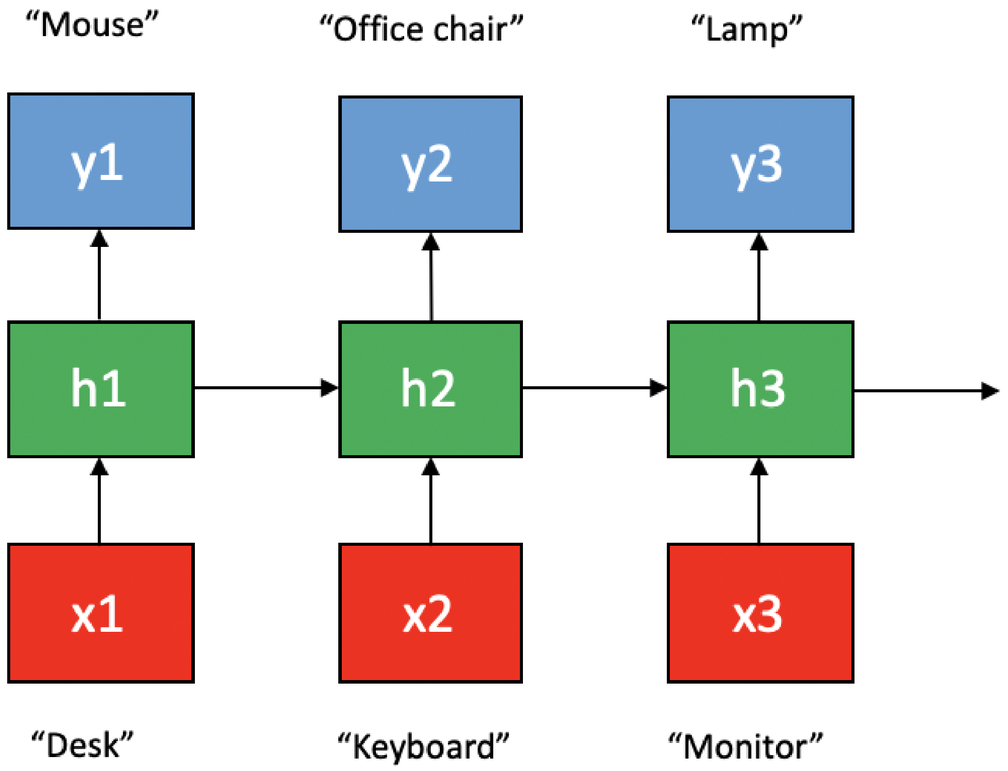


Wide and Deep Model can take rich data as input. The wide part can effectively memorize sparse feature interactions using cross-product feature transformations such as smb_flg – paper_flg, while deep part can generalize to previously unseen feature interactions through low dimensional user / item embeddings similar to NCF.

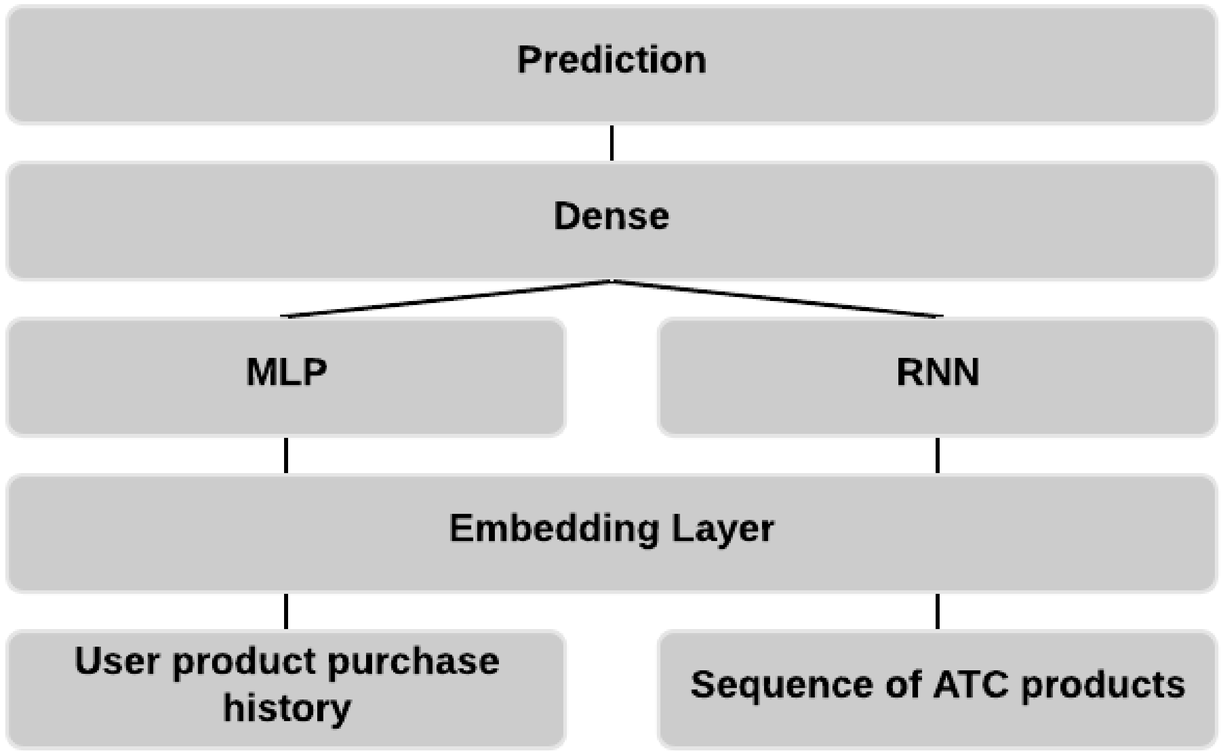
```
val wideAndDeep: = WideAndDeep[Float](modelType, numClasses, columnInfo)

val optimizer = Optimizer(model = wideAndDeep, sampleRDD = trainRdds, criterion = ClassNLLCriterion[Float](), batchSize = batchSize)

val wnd_model = optimizer
  .setOptimMethod(new Adam[Float](learningRate = 1e-2, learningRateDecay = 1e-5))
  .setEndWhen(Trigger.maxEpoch(maxEpoch)).optimize()
```

Sequence Recommender





Sequence Recommender



The Good:
Can catch the latest purchase intent from current session behavior and adjust its product recommendation in real time
Work with both anonymous / identified customers
No pre-filtering mechanism required, simpler serving architect

The Bad:
Sequence window size is hard to set
Online inference requires lots of resources

Performance Comparison

Offline measurement:

Method	Top 5 Accuracy
Sequence	52.3%
Wide & Deep	45.2%
NCF	46.7%
ALS	16.2%

Online measurement:

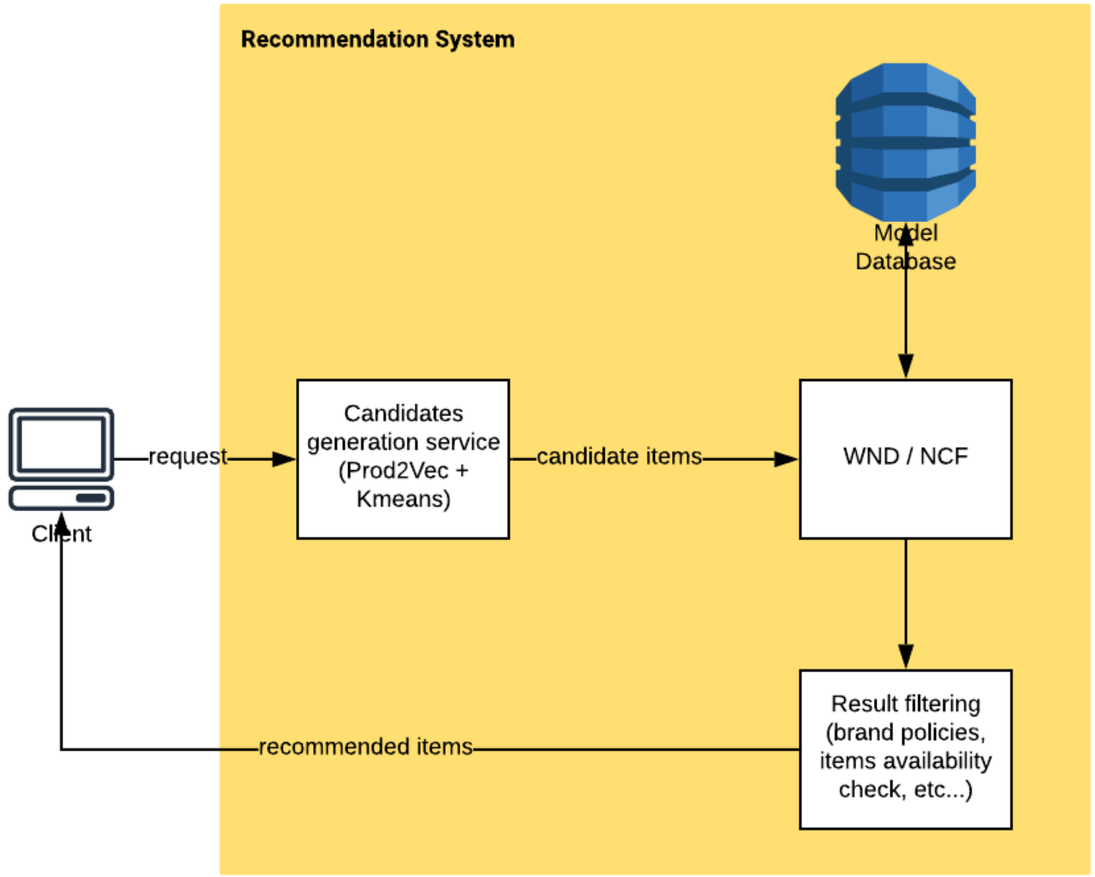
Online A/B testing shows the test group using sequence model lifted sales by 1% and average order value by 1.6% compared to control group

*Testing by Office Depot

Note: test data provided by Office Depot

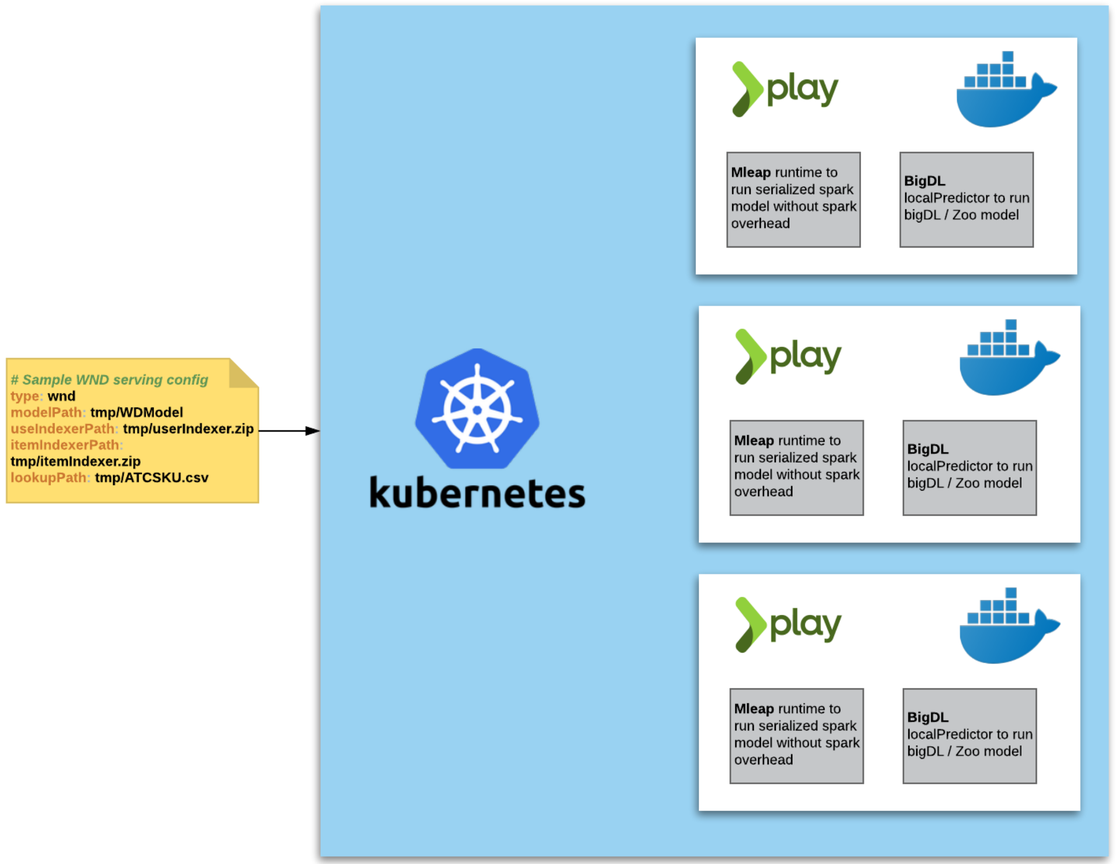
Recommendation System Deployment

- Neural embedding to generate candidate items
- Recommendation service ranking candidate items by probabilities
- Apply post filtering rules according to business needs



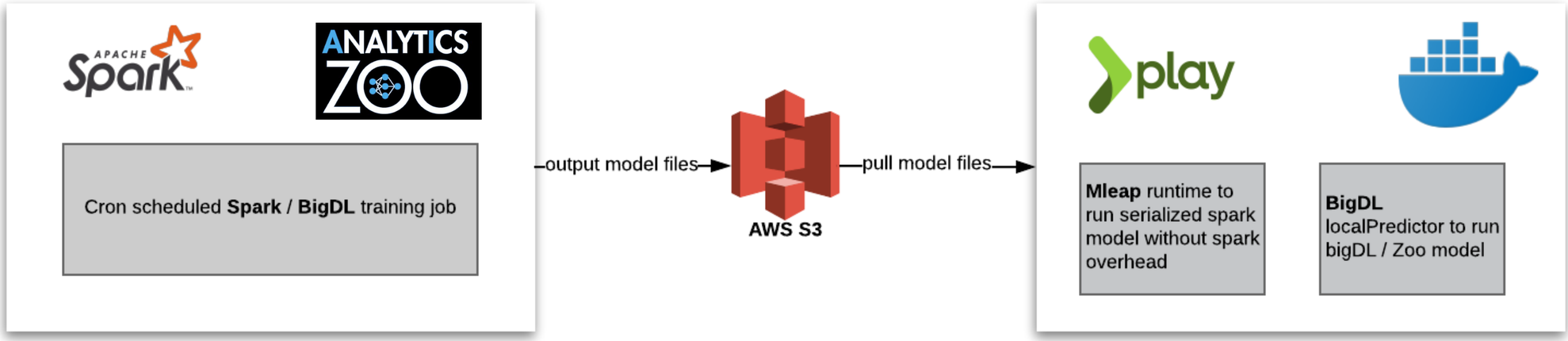
Recommendation System Deployment

- Use model configuration file to deploy different models to production
- Ability to scale up / down according to the current workload using Kubernetes

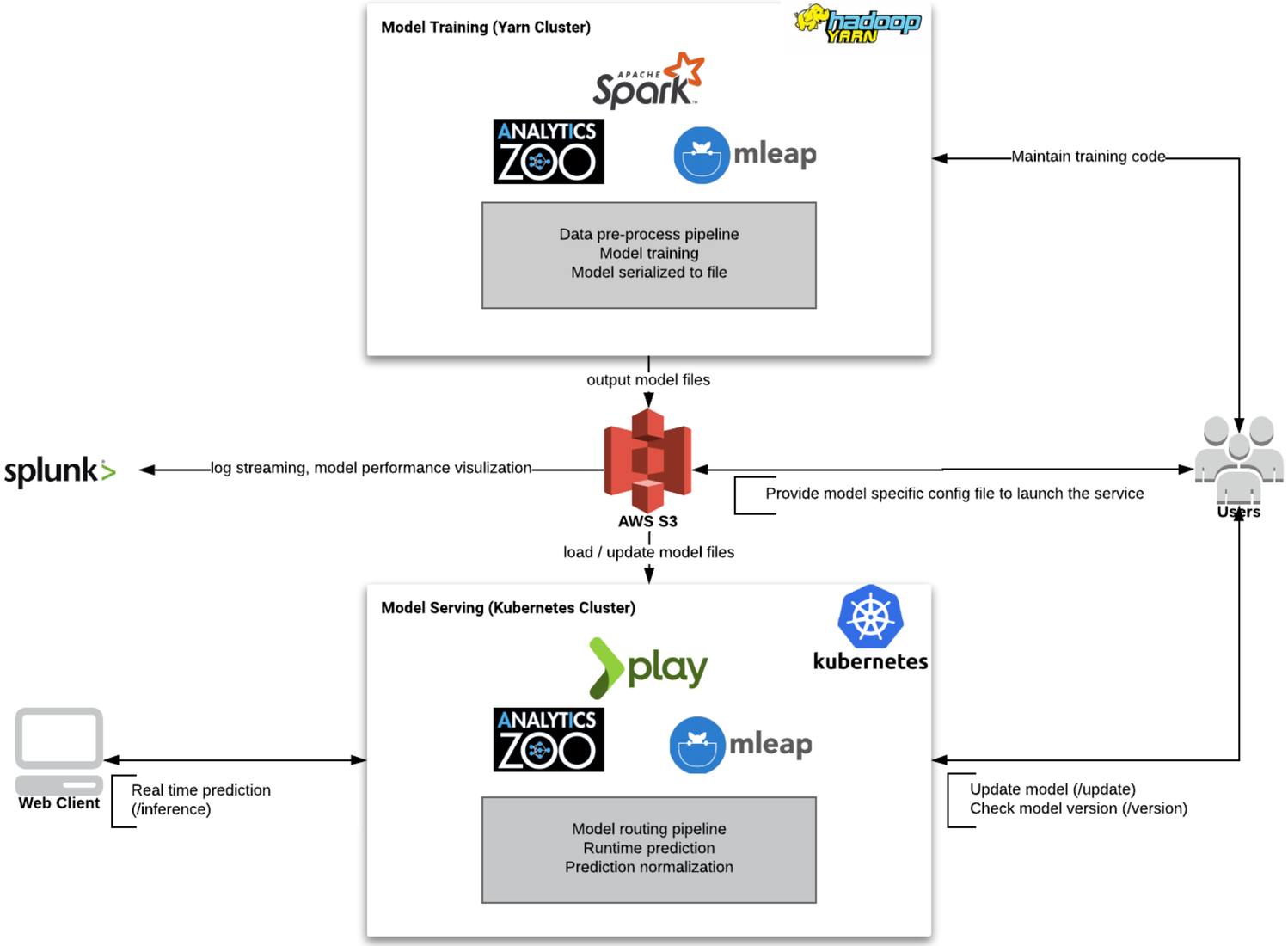


Recommendation System Deployment

- Self-updated / automated model deployment pipeline
- No down time when update model in production



Deploy Zoo Models to Production



Takeaways

- Analytics Zoo/BigDL integrates well into existing AWS Spark ETL and machine learning platform
- Analytics Zoo/BigDL provides inference/local predictor for high performance real-time inference
- Deep learning can drastically reduce manual feature engineer workload compare to traditional machine learning approach
- Deep learning based recommendation provides more flexibility to combine different model architectures for different use cases
- By mapping item to word and session to document, lots of NLP algorithms can be utilized for recommendation
- More information available at https://analytics-zoo.github.io

References

- Yehuda Koren. (2009). Matrix Factorization Techniques for Recommender Systems. Retrieved from https://dl.acm.org/citation.cfm?id=1608614
- Xiangnan He. (2017). Neural Collaborative Filtering. Retrieved from https://arxiv.org/abs/1708.05031
- Heng-Tze Cheng. (2016). Wide & Deep Learning for Recommender Systems. Retrieved from https://arxiv.org/abs/1606.07792
- Sai Wu. (2016). Personal recommendation using deep recurrent neural networks in NetEase. Retrieved from http://cfm.uestc.edu.cn/~zhangdongxiang/papers/ICDE16_industry_231.pdf

Questions?