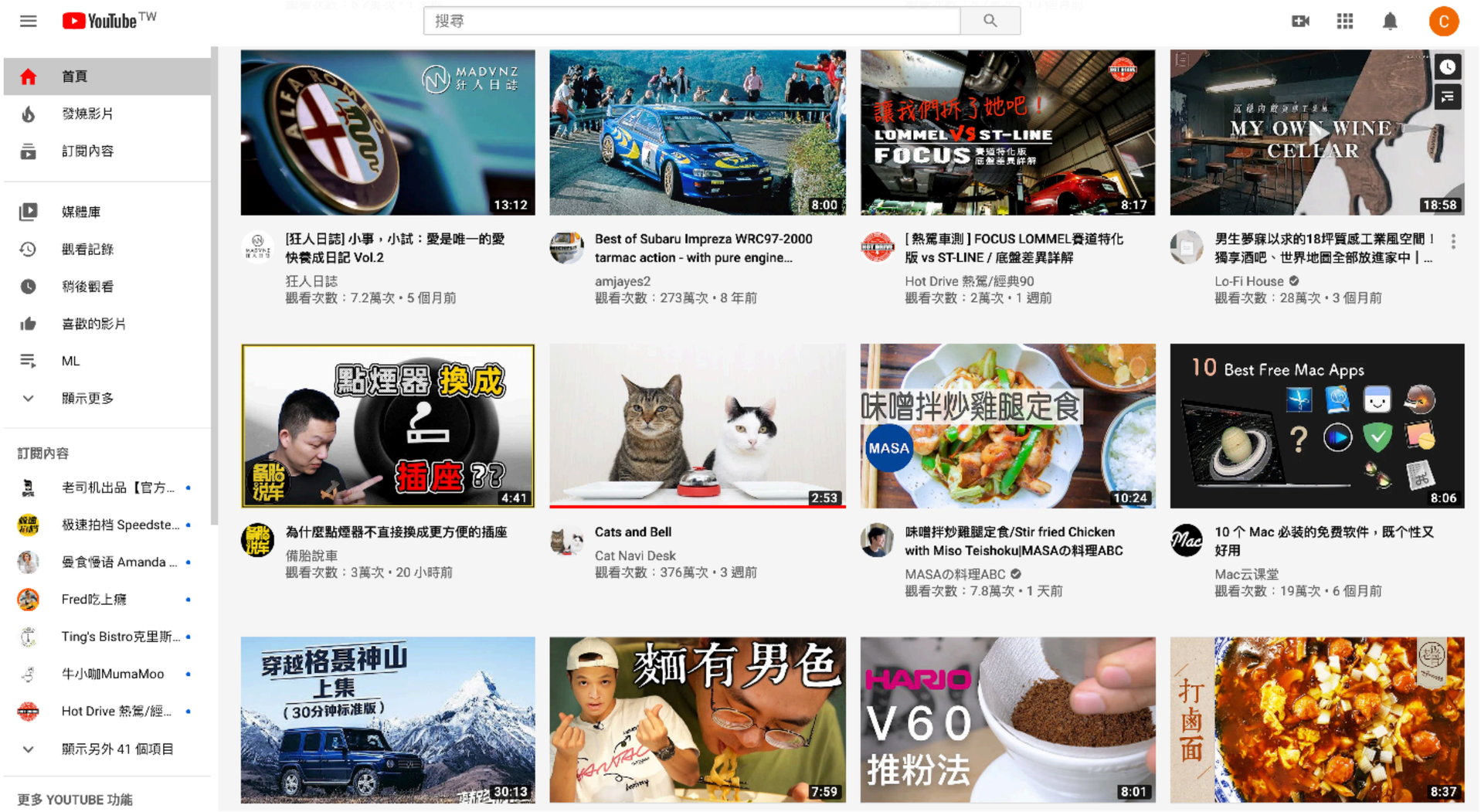
Modern recommender system in large content website

Cyrus Chiu @ PyConTW 2020

Youtube - Landing Page



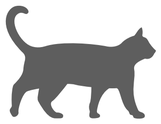
About

Cyrus Chiu

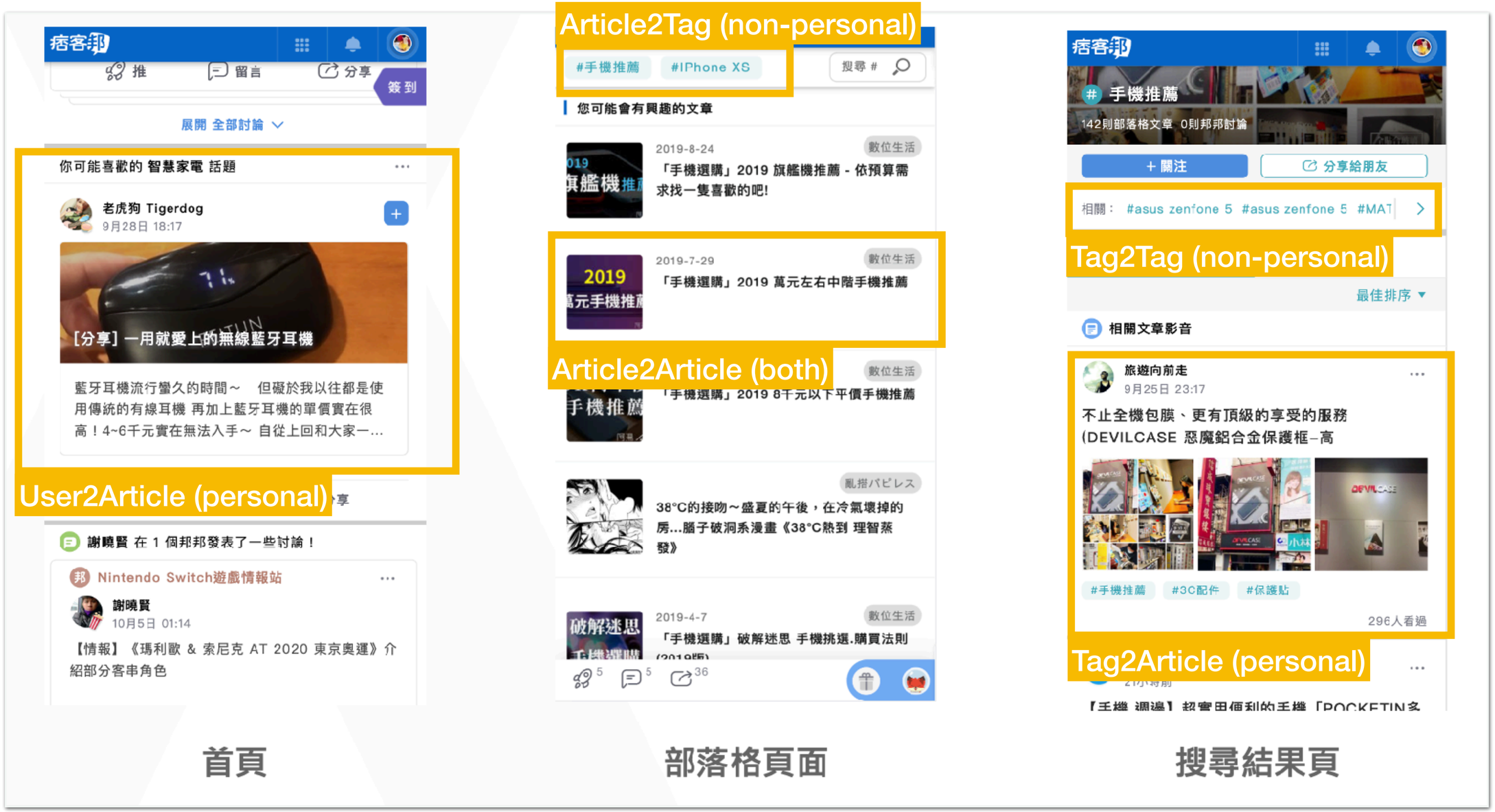
www.linkedin.com/in/iamcyruschiu

a ML engineer at a financial (technology) company





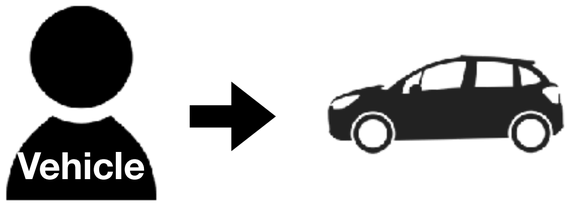

Use case in PIXNET

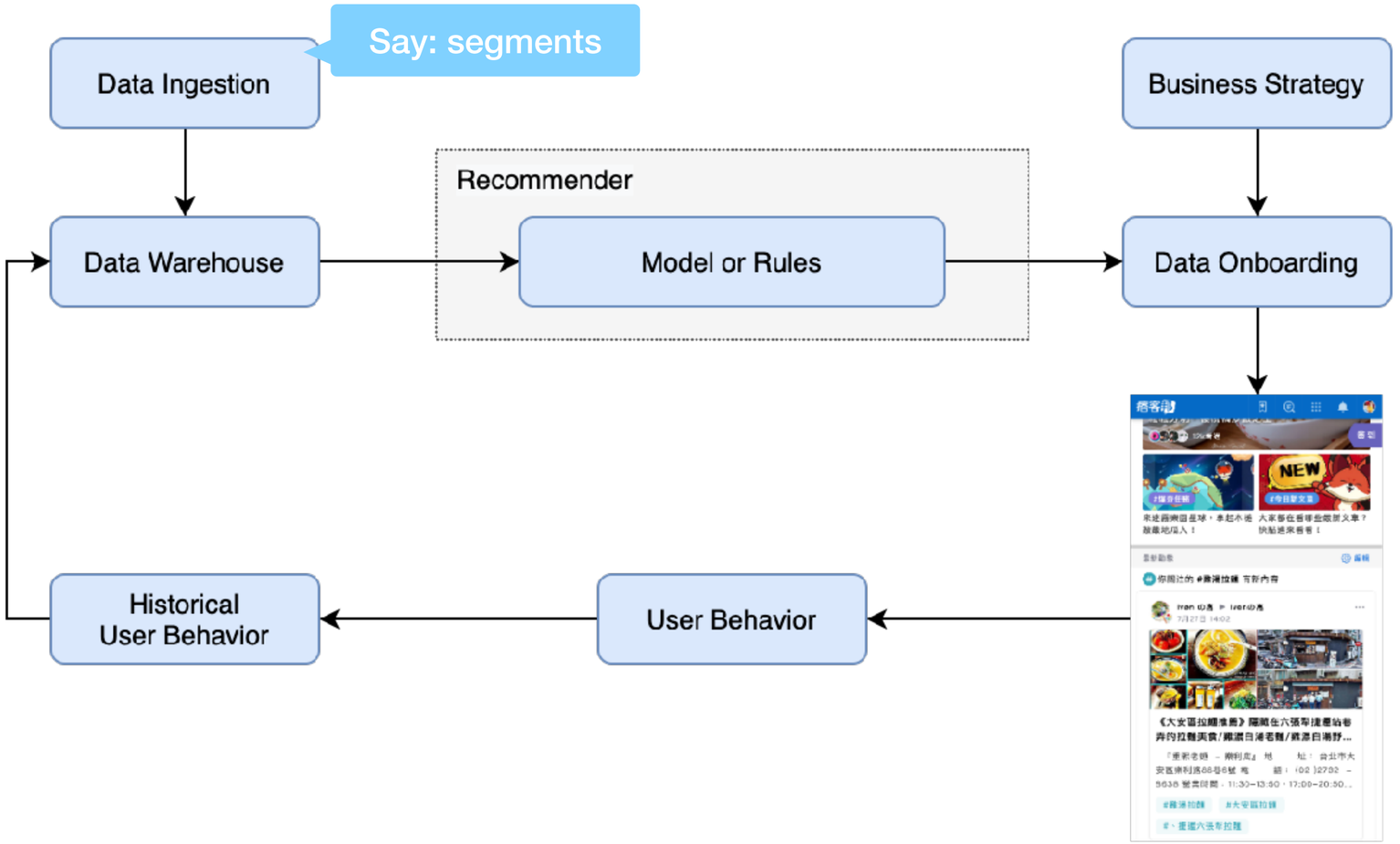


How we building a recommender system

Basic

- Data infrastructure
- Data collection
- User tracking
- Some rules or model
- Position on the webpage

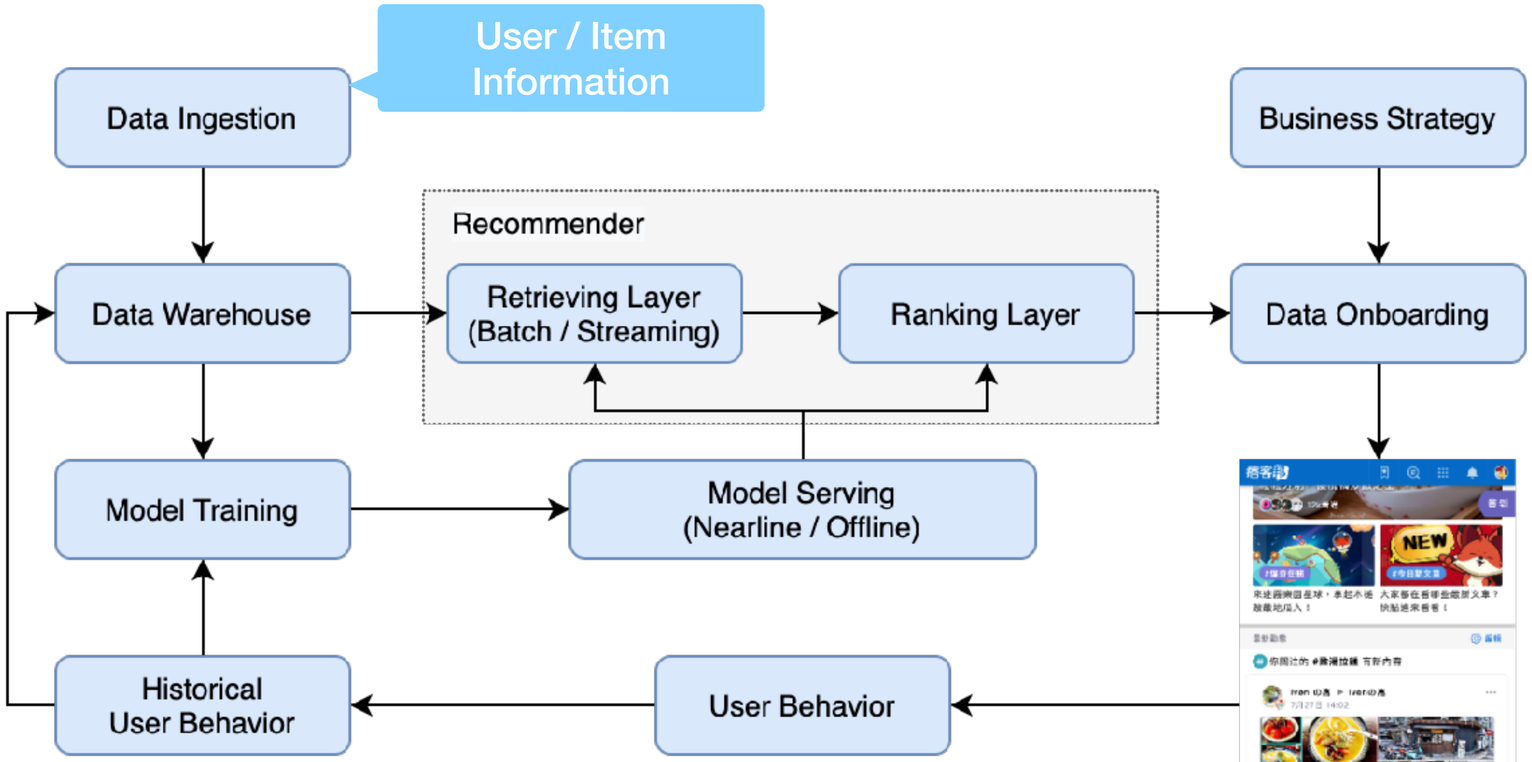




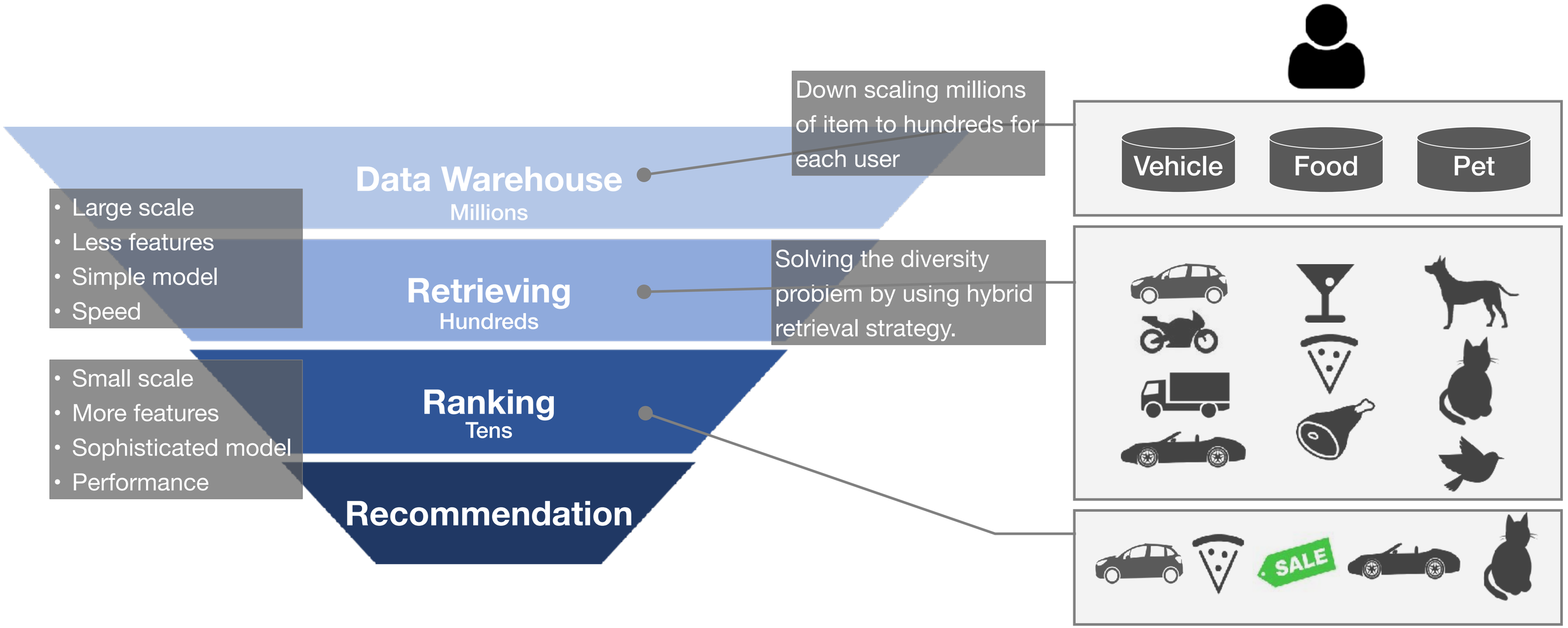
How we building a recommender system

Advanced

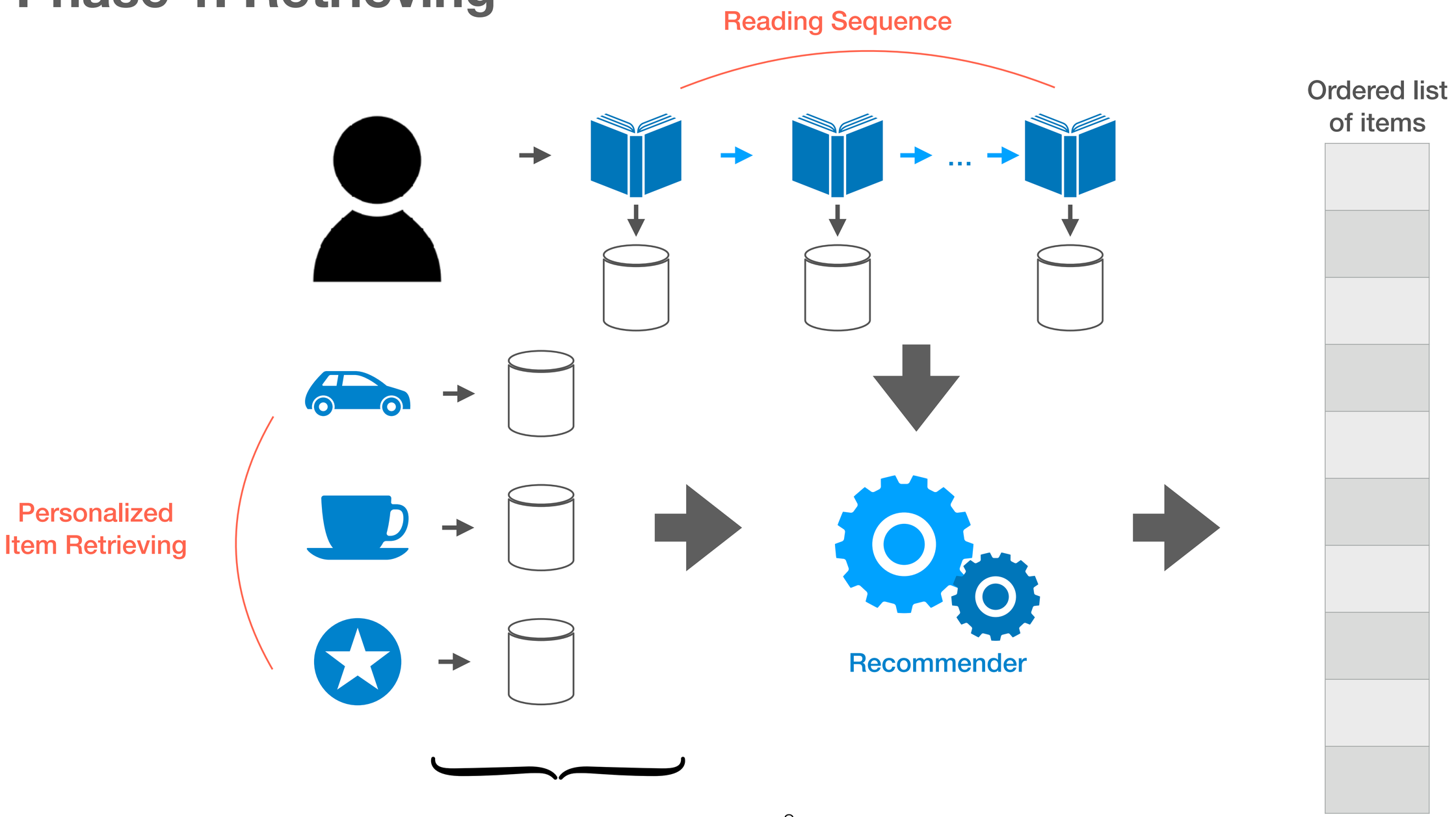
- Data lifecycle
- Scalability
- User profiling
- Item profiling
- Item retrieving
- Item ranking



What is 2–phase recommender system



Phase 1: Retrieving



Define the similarity between item A and B

According to the content

According to the user behavior

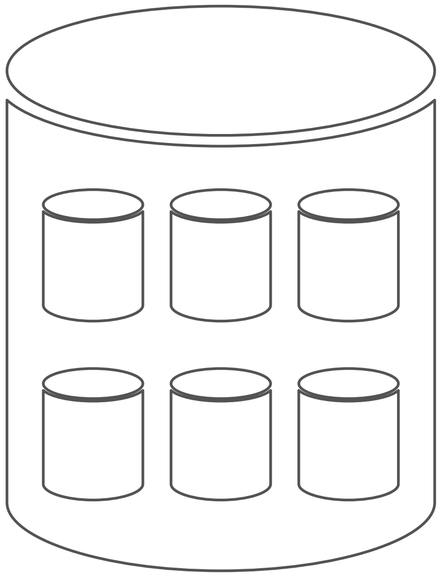
Some Retrieving Methods

Strategy	Search Engine	Word Embedding	Item Embedding	Frequently Occurring Together	Hot
Features	• Easy to use • Multiple query type • Another cost to maintain your Elasticsearch cluster	• Unexplainable • Vector space • High computational cost	• Cold start problem • Data sparsity issue • Vector space • High computational cost	• Cold start problem • Data sparsity issue • Explainable • Easy to maintain	• Explainable • Easy to use • Suitable for any situations
Recommended	★★★★	★★	★★★★	★★★★★	★★★

Phase 2: Ranking

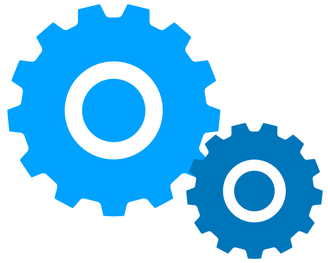
User Information	Item Information	Context Information

Ordered list of items



Candidate of items

$$f\,(\,user,\,item,\,context\,)$$

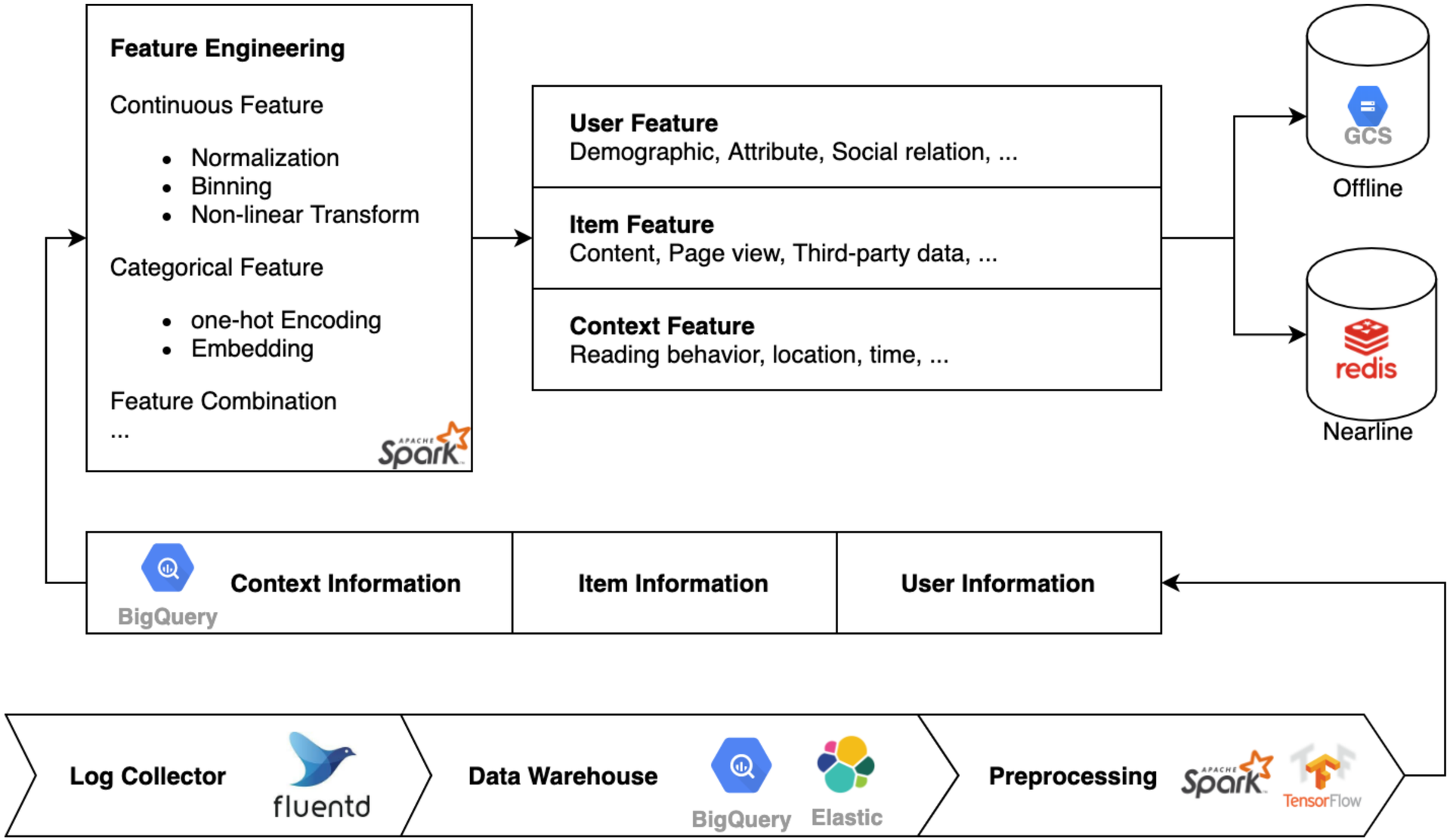


Recommender

Evolution of the prediction model

Model	Logsitc Regression	GBDT + LR	Factorization Machine	DNN
Features	• Fast • Highly explainable • Manually crafting features • Limited effective	• Fast • Highly explainable • Capability to generate feature interaction	• Slow • Explainable • Good at learning feature interactions • Exhausting resources	• More slowly • Unexplainable • Learning sophisticated feature interactions automatically • Exhausting more resources • Wide & Deep, Deep & Cross, DeepFM, xDeepFM, …

RecSys on DMP



Example: Find the pair-wise distance


```
array([[0.34723291, 0.80343977, 0.60506014, 0.23067618, 0.79965735],
       [0.43726941, 0.04598019, 0.19007782, 0.85376095, 0.25351579],
       [0.24234896, 0.22349041, 0.63439333, 0.40484165, 0.57103437],
       [0.38264449, 0.93276329, 0.73085271, 0.98900326, 0.52048102],
       [0.99288529, 0.31834113, 0.97249914, 0.08776034, 0.90290914],
       [0.90180205, 0.63893861, 0.22106869, 0.09200385, 0.86880432],
       [0.04775797, 0.44545171, 0.9665554 , 0.27558683, 0.13279217],
       [0.40203503, 0.53100031, 0.67645917, 0.17551579, 0.37494095],
       [0.77076934, 0.12681486, 0.1971458 , 0.79065758, 0.4628704 ],
       [0.20363516, 0.72497933, 0.92195548, 0.96016578, 0.340993  ]])
```

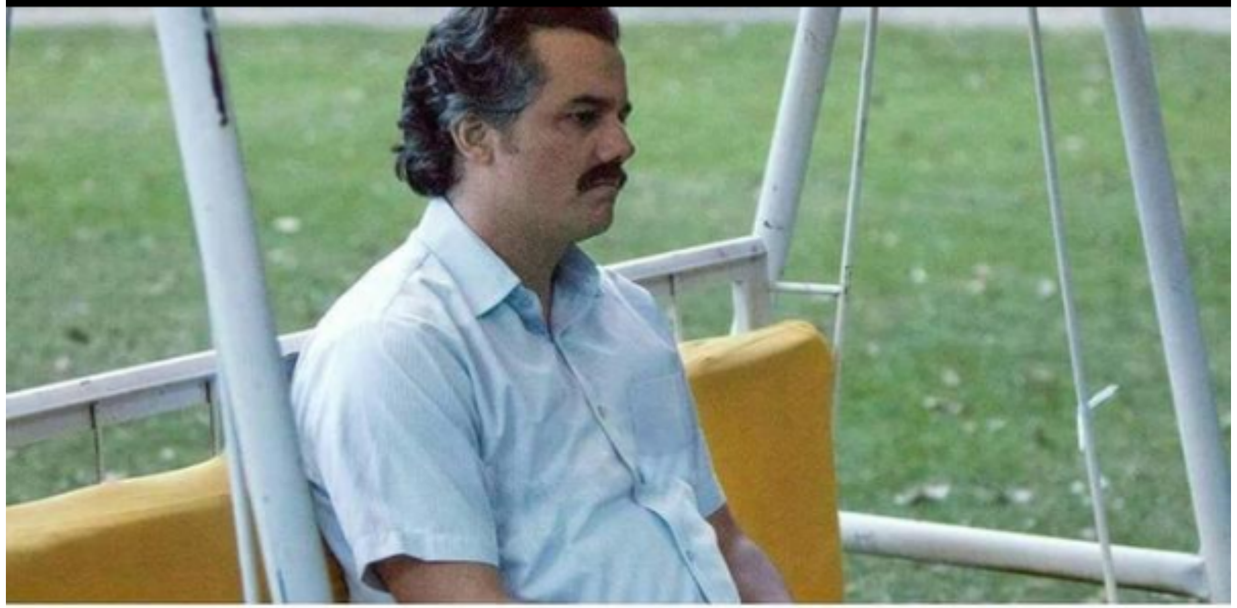
Solution 1

```
from sklearn.metrics.pairwise import cosine_similarity
cosine_similarity(X)
```

Solution 2

```
from sklearn.metrics.pairwise import pairwise_distances
pairwise_distances(X, metric="cosine")
```

When you submit the job without parallel computing..



OUT OF MEMORY ...

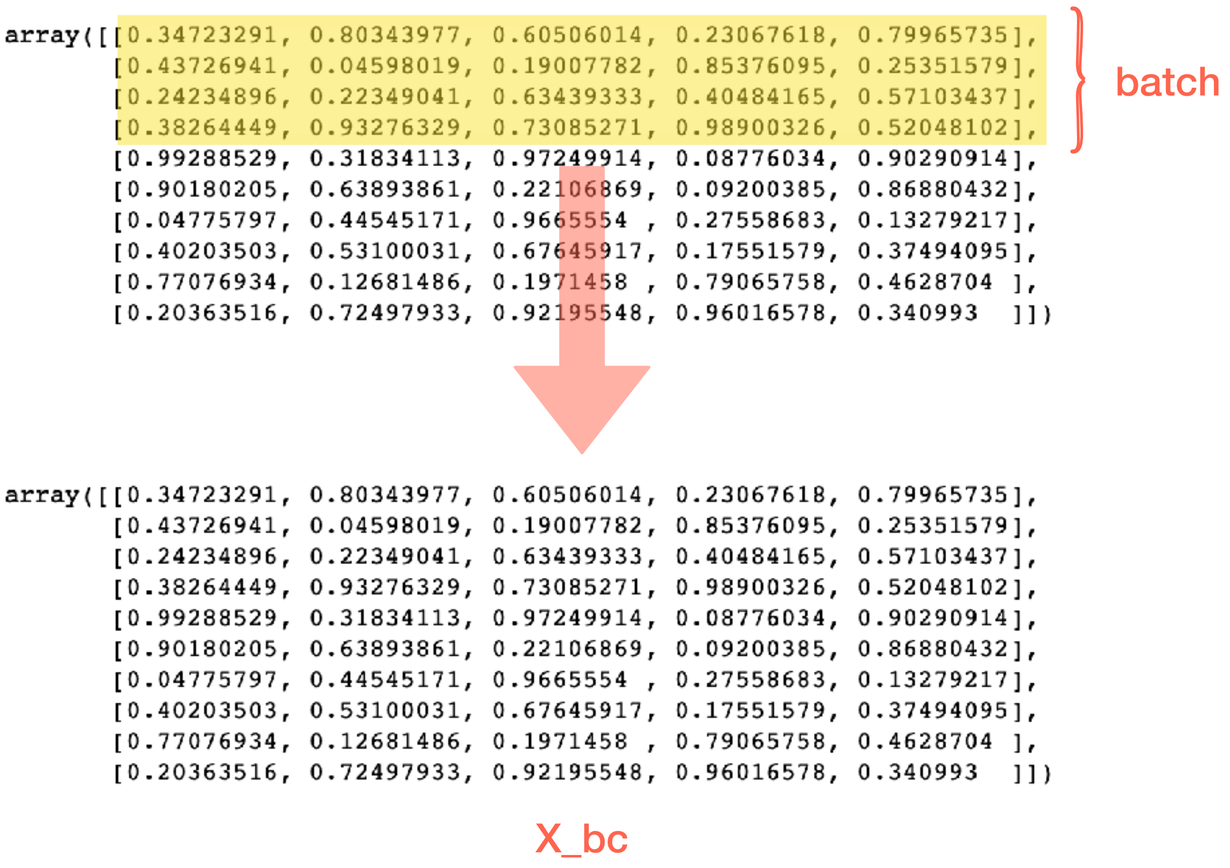
import pyspark (1)

Solution: RDD

```
X_bc = sc.broadcast(X)
n_part = 100
batch_size = 64
data_size = X.shape[0]

batch_vectors_rdd = sc.parallelize(
    [X[i: i+batch_size]
        for i in range(0, data_size, batch_size)
    ],
n_part)

batch_vectors_rdd.mapPartitions(
    lambda x: cosine_similarity(x, X_bc.value))
```



import pyspark (2)

Solution: DataFrame

```
@F.udf(returnType=DoubleType())
def get_distance(i, j):
    sim = cosine_similarity(i, j)
    return sim

vector_sdf.alias("i").join(
    vector_sdf.alias("j"),
    col("i.id") < col("j.id")).select(
        F.col("i.id").alias("i"),
        F.col("j.id").alias("j"),
        get_distance("i.vector", "j.vector").alias(
            "cosine_similarity")
    )
```

I

	vector	id
0	[0.01227569580078125, -0.07232666015625, 0.036...	59358
1	[-0.00469970703125, -0.0190277099609375, 0.011...	59427
2	[0.04229736328125, -0.0195465087890625, -0.058...	59389
3	[0.0528564453125, -0.0261688232421875, 0.00719...	59373
4	[0.043609619140625, -0.041107177734375, -0.022...	59330
...	...	...
83118	[0.034881591796875, -0.0396728515625, 0.002809...	38903
83119	[0.08929443359375, -0.0239410400390625, 0.0203...	38873
83120	[1.0478515625, 0.6865234375, -1.03125, 0.82080...	38651
83121	[-0.01203155517578125, 0.0203857421875, -0.020...	38807
83122	[0.025238037109375, -0.056640625, 0.0049400329...	38987

J = I

	vector	id
0	[0.01227569580078125, -0.07232666015625, 0.036...	59358
1	[-0.00469970703125, -0.0190277099609375, 0.011...	59427
2	[0.04229736328125, -0.0195465087890625, -0.058...	59389
3	[0.0528564453125, -0.0261688232421875, 0.00719...	59373
4	[0.043609619140625, -0.041107177734375, -0.022...	59330
...	...	...
83118	[0.034881591796875, -0.0396728515625, 0.002809...	38903
83119	[0.08929443359375, -0.0239410400390625, 0.0203...	38873
83120	[1.0478515625, 0.6865234375, -1.03125, 0.82080...	38651
83121	[-0.01203155517578125, 0.0203857421875, -0.020...	38807
83122	[0.025238037109375, -0.056640625, 0.0049400329...	38987

CROSS JOIN

CosineSimilarity (
Column I.vector,
Column J.vector
)

Comparison

Calculating distances between coordinates from a 32*1k/10k/100k matrix (32 core, 32 GB memory)

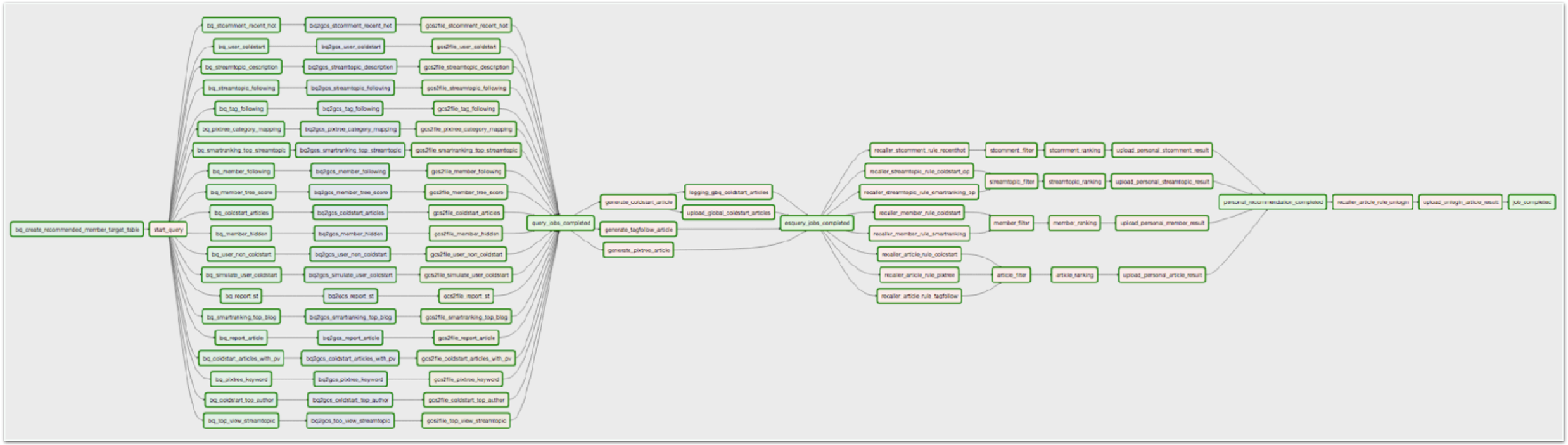
	1K	10K	100K
sklearn	0.1	1.5	OOM
pyspark	2	5.1	70
Faiss	0.05	0.6	21

https://github.com/facebookresearch/faiss

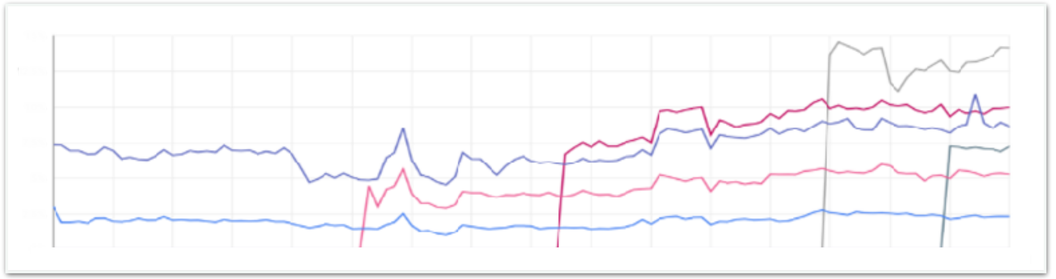
Workflows of the data project

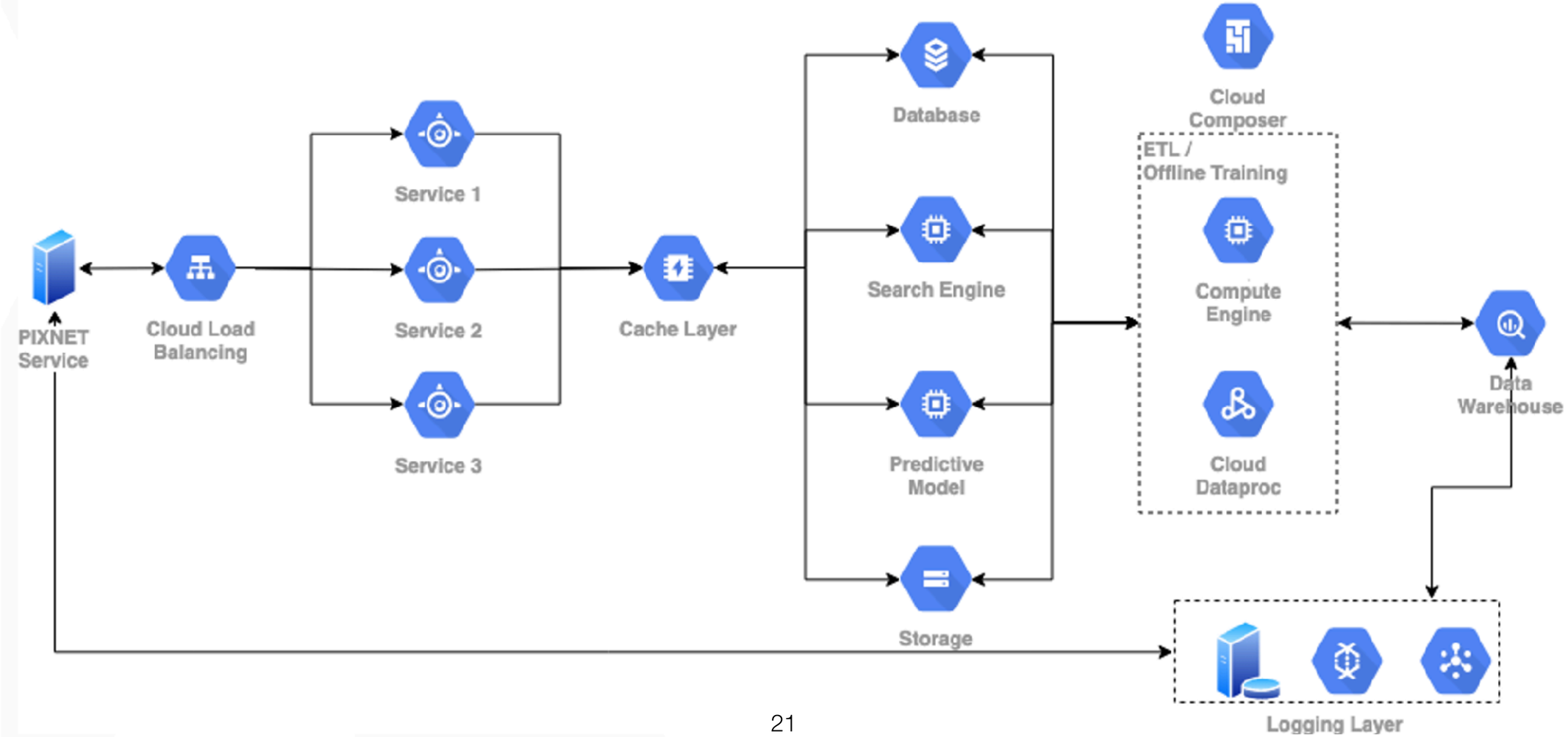
Data Management	Programming	Project Management
• ETL pipeline design • Data infra design • Metalayer design • Data toolkit	• SQL • Framework design • Library development • Parallel computing • Machine learning	• Git flow • Review

Using Apache Airflow to schedule and monitor workflows

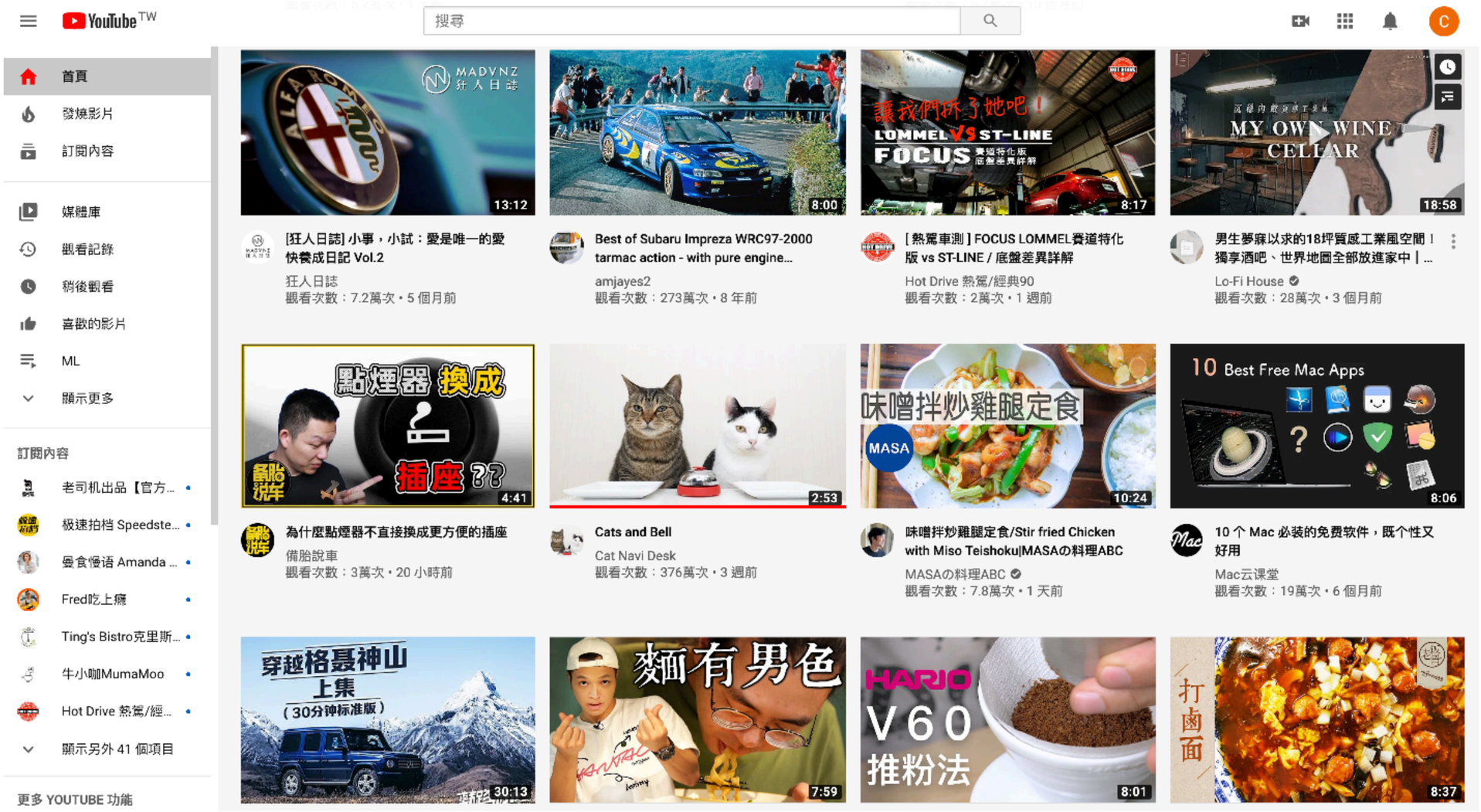


Online A/B Testing

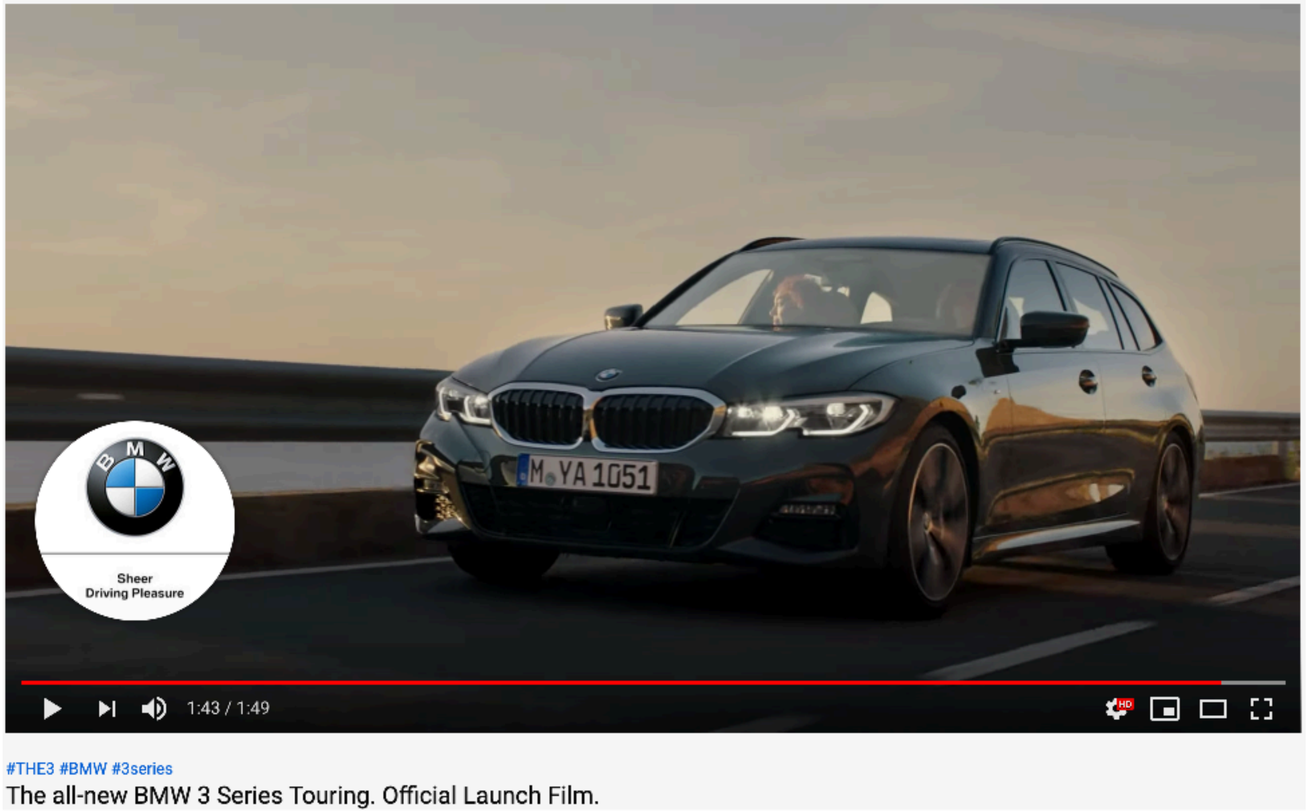




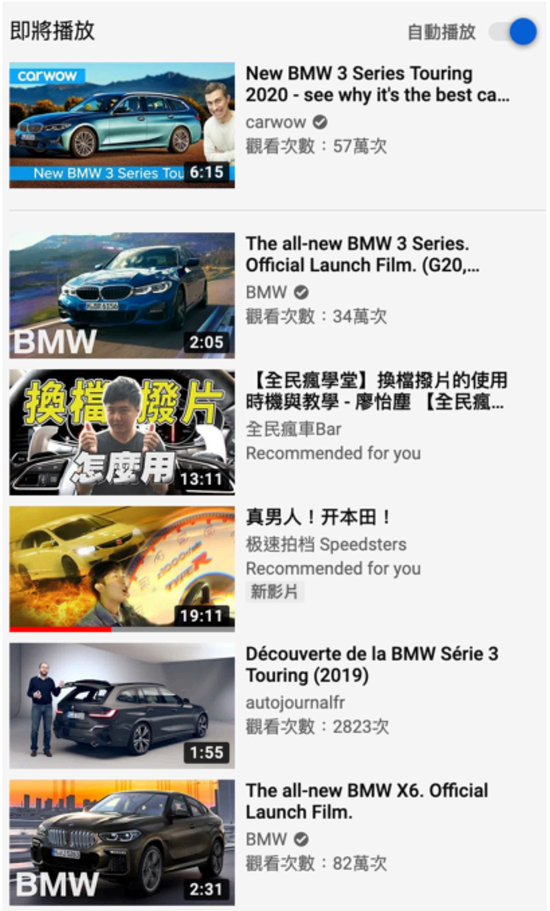
Youtube - Landing Page



Youtube - Video Page



#THE3 #BMW #3series
The all-new BMW 3 Series Touring. Official Launch Film.



Thank You