Search & Recommendations:
3 Sides of the Same Coin

Nick Pentreath
Principal Engineer

@MLnick

IBM CODE

About

@MLnick on Twitter & Github

Principal Engineer, IBM

CODAIT - Center for Open-Source Data & AI Technologies

Machine Learning & AI

Apache Spark committer & PMC

Author of *Machine Learning with Spark*

Various conferences & meetups

Center for Open Source Data and AI Technologies

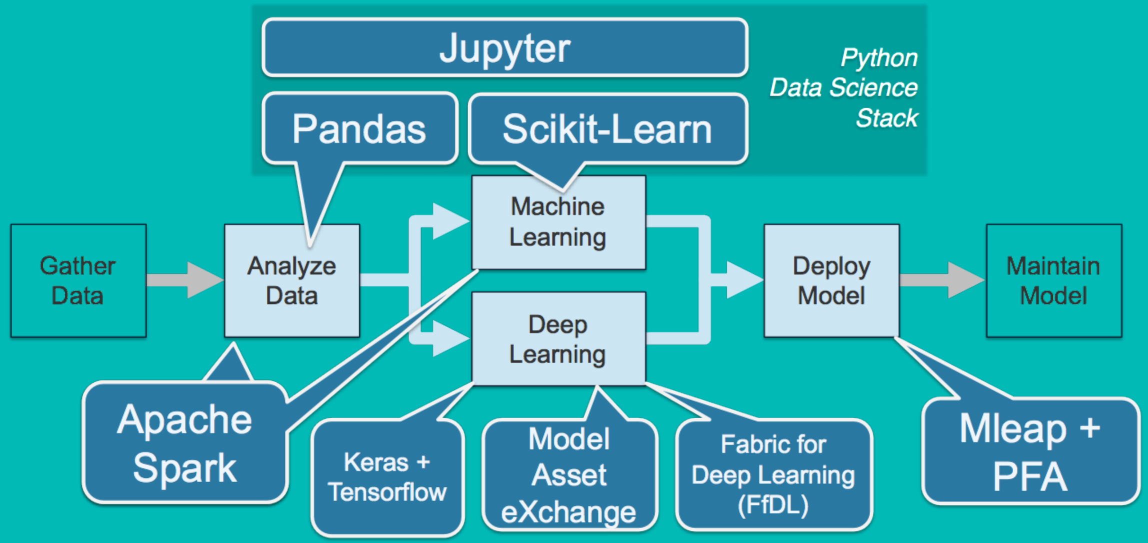
CODAIT

codait.org

CODAIT aims to make AI solutions dramatically easier to create, deploy, and manage in the enterprise

Relaunch of the Spark Technology Center (STC) to reflect expanded mission

Improving Enterprise AI Lifecycle in Open Source



Agenda

Recommender systems overview

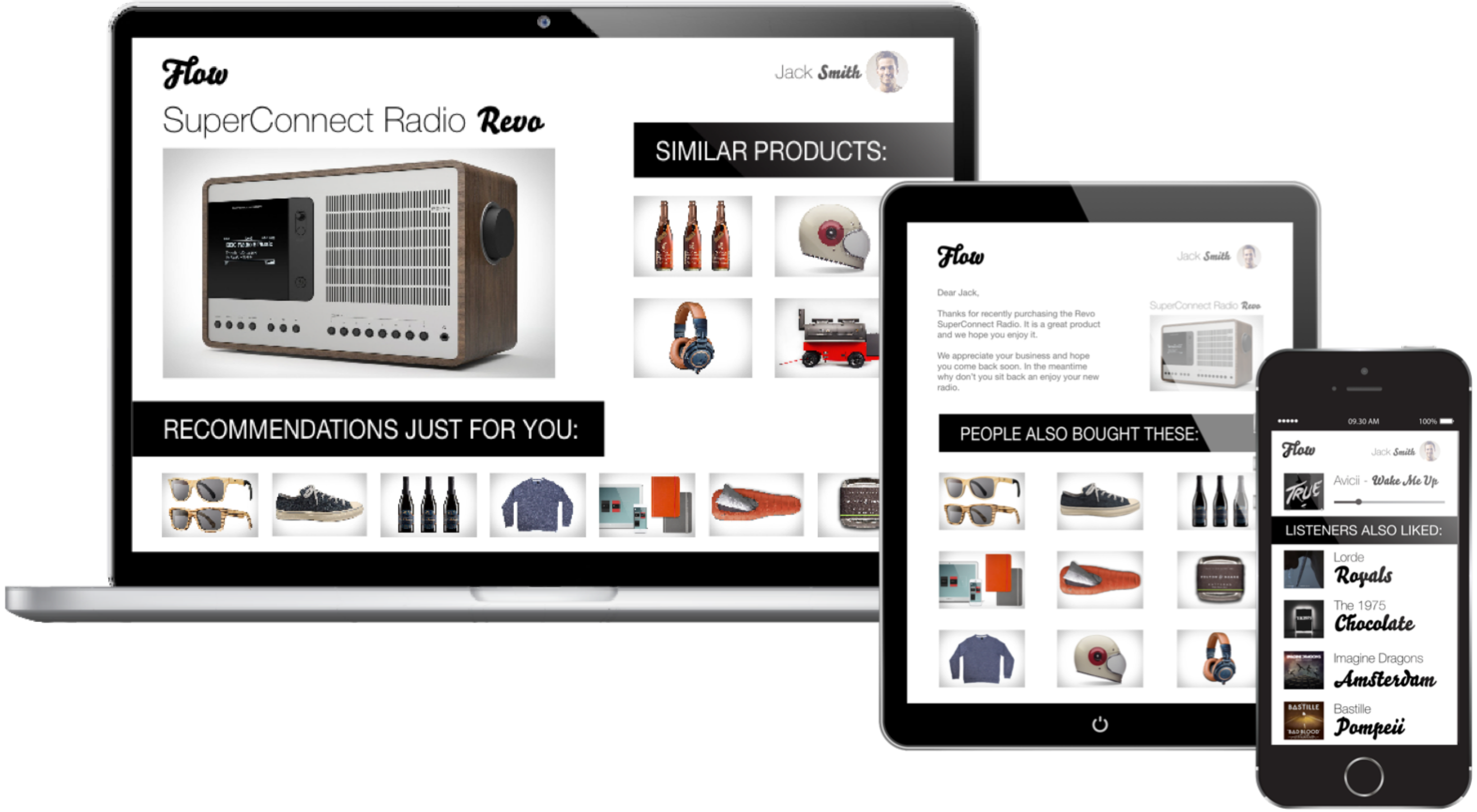
Search & Recommendations

3 Sides of the Coin

Performance

Summary

Recommender Systems



Users and Items

```
{
  "user_id": "1",
  "name": "Joe Bloggs",
  "created_date": 1476884080,
  "updated_date": 1476946916,
  "last_active_date": 1476946962,
  "age": 32,
  "country": "DE",
  "city": "Berlin",
  ...
}
```

```
{
  "item_id": "10",
  "name": "LOL Cats",
  "description": "catscatscats",
  "category": ["Cat Videos", "Humour", "Animals"],
  "tags": ["cat", "lol", "funny", "cats", "felines"],
  "created_date": 1476884080,
  "updated_date": 1476884080,
  "last_played_date": 1476946962,
  "likes": 100000,
  "author_id": "321",
  "author_name": "ilikecats",
  "channel_id": "CatVideoCentral",
  ...
}
```

System Requirements

Filtering & Grouping


```
{
  "user_id": "1",
  "name": "Joe Bloggs",
  "created_date": 1476884080,
  "updated_date": 1476946916,
  "last_active_date": 1476946962,
  "age": 32,
  "country": "DE",
  "city": "Berlin",
  ...
}
```

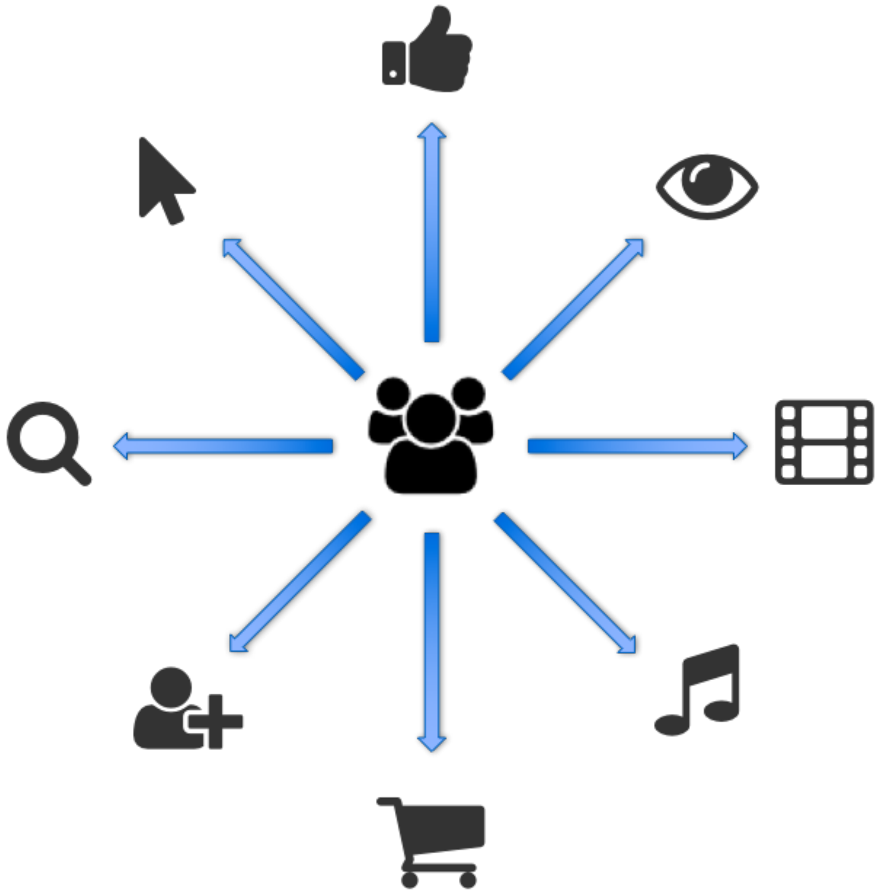


```
{
  "item_id": "10",
  "name": "LOL Cats",
  "description": "catscatscats",
  "category": ["Cat Videos", "Humour", "Animals"],
  "tags": ["cat", "lol", "funny", "cats", "felines"],
  "created_date": 1476884080,
  "updated_date": 1476884080,
  "last_played_date": 1476946962,
  "likes": 100000,
  "author_id": "321",
  "author_name": "ilikecats",
  "channel_id": "CatVideoCentral",
  ...
}
```


Business Rules

Events

– Implicit preference data
  - Online – page view, click
  - Commerce – add-to-cart, purchase, return
– Explicit preference data
  - Ratings, reviews
– Intent
  - Search query
– Social
  - Like, share, follow, unfollow, block



Context

```json
{
  "user_id": "1",
  "item_id": "10",
  "event_type": "page_view",
  "timestamp": 1476884080,
  "referrer": "http://codait.org",
  "ip": "123.12.12.12",
  "device_type": "Smartphone",
  "user_agent_os": "Android",
  "user_agent_type": "Mobile Browser",
  "user_agent_family": "Chrome Mobile",
  "geo":"52.520", "13.405"
  ...
}
```

How to handle implicit feedback?


```
{
  "user_id": "1",
  "item_id": "10",
  "event_type": "page_view",
  "weight": 1.0,
  "timestamp": 1476884080,
  "referrer": "http://codait.org",
  "ip": "123.12.12.12",
  "device_type": "Smartphone",
  "user_agent_os": "Android",
  "user_agent_type": "Mobile Browser",
  "user_agent_family": "Chrome Mobile",
  "geo":"52.520", "13.405"
  ...
}
```

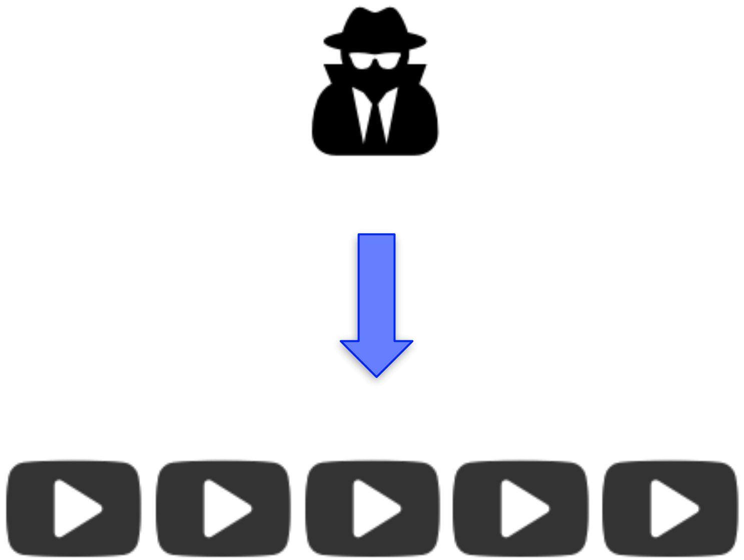
Cold Start

New items

- No historical interaction data
- Typically use baselines or item content

New (or unknown) users

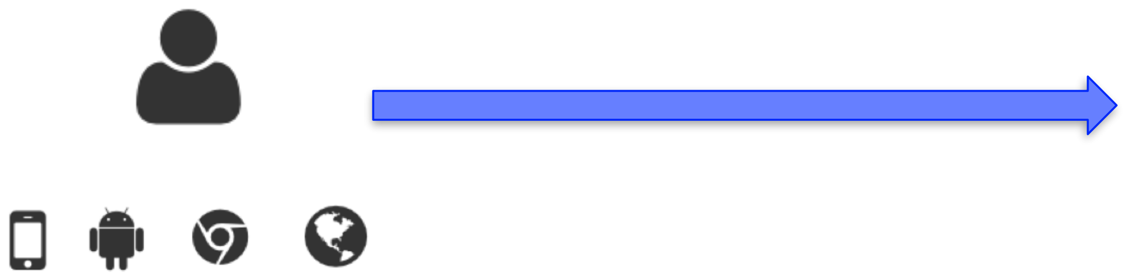
- Previously unseen or anonymous users have no user profile or historical interactions
- Have context data (but possibly very limited)
- Cannot directly use collaborative filtering models
  - Item-similarity for current item
  - Represent user as aggregation of items
  - Contextual models can incorporate short-term history

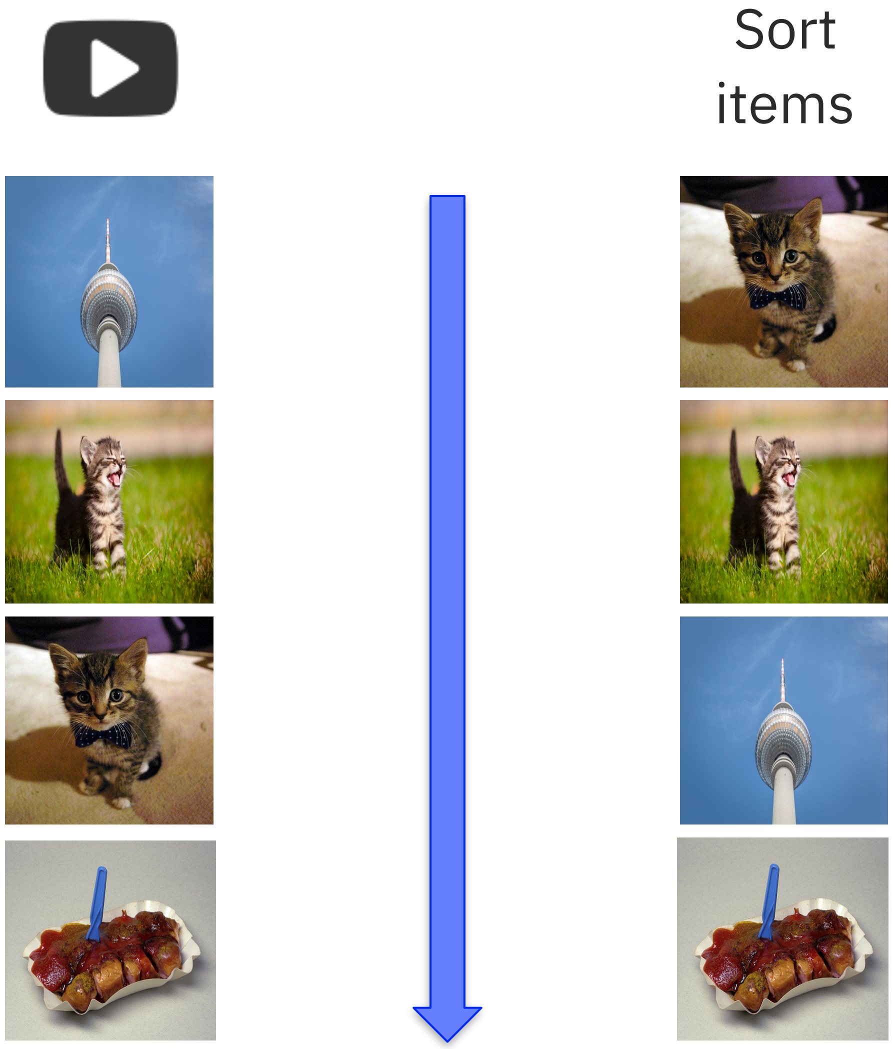


Prediction

Recommendation is ranking

- Given a user and context, rank the available items in order of likelihood that the user will interact with them





Serving Requirements

– Serving => ranking large # items

– Often need to filter

  - categories, popularity, price, geo, time

– Use all data at prediction time

  - content, context, preference, session

– Need to scale with item set and feature set

– Handle cold start

  - content-based models or fallbacks, item-aggregates

– Easily incorporate new preference data

  - model re-training, fold in

Search &
Recommendations

Prelude: Ratings Matrix

$$
\begin{bmatrix}
3 & & & \\
 & 1 & & \\
 & 5 & & 2 \\
1 & & & \\
 & & 2 & 1
\end{bmatrix}
$$

Prelude: Item-item Co-occurrence

One of the earliest approaches

- Compute item co-occurrence matrix from ratings matrix, i.e. $X^t X$
- Scoring can be online or pre-computed
- Similarity metric between items

$$\begin{bmatrix} 3 & & & \\ & 1 & & \\ & 5 & & 2 \\ 1 & & & \\ & & 2 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & & & 1 & \\ & 1 & 1 & & \\ & 1 & 1 & & 1 \\ 1 & & & 1 & \\ & & 1 & & 1 \end{bmatrix}$$

Prelude: Matrix Factorization

One of the de facto standard models

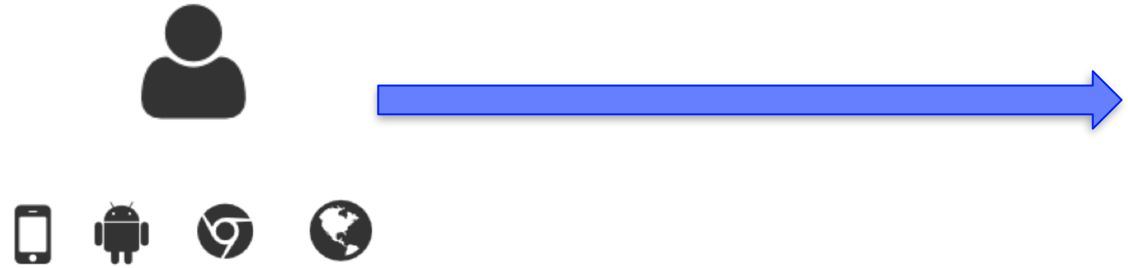
- Find two smaller matrices (called the factor matrices) that approximate the ratings matrix
- Minimize the reconstruction error (i.e. rating prediction / completion)
- Efficient, scalable algorithms
  - Gradient Descent; Alternating Least Squares (ALS)
- Prediction is simple
- Can handle implicit data through weighting

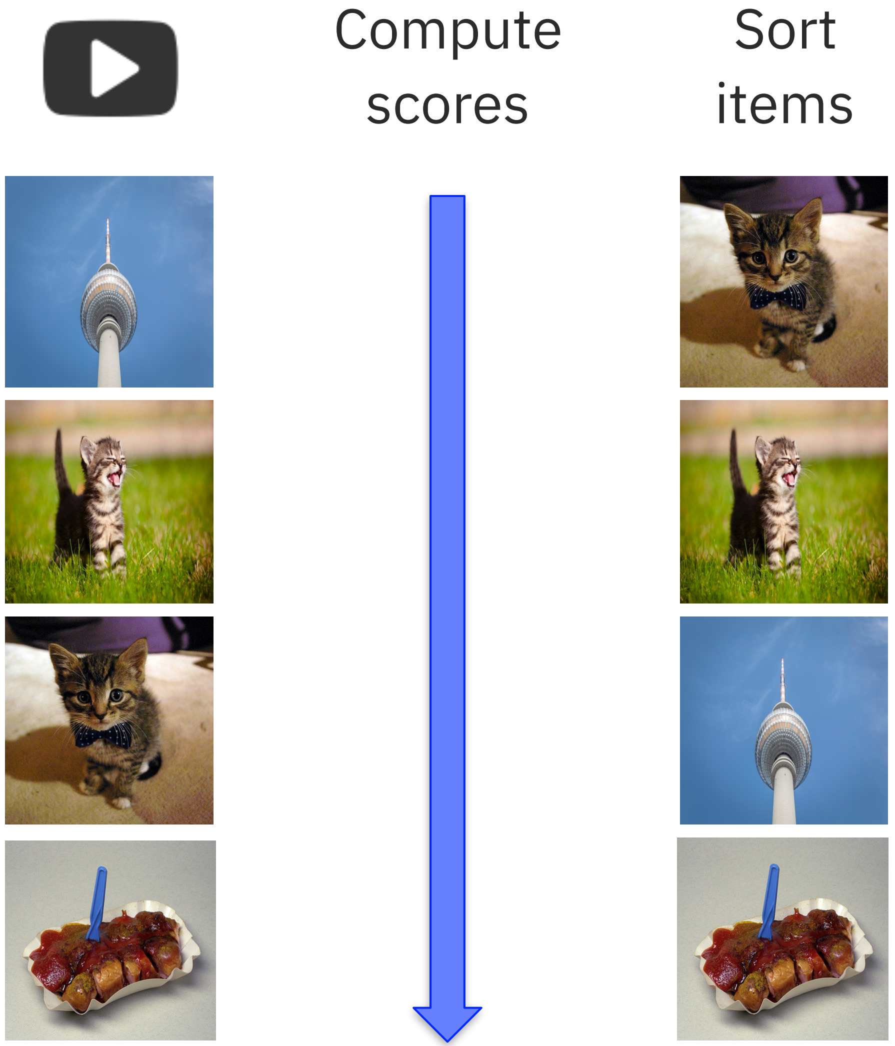
$$\begin{bmatrix} 3 & & & & 4 \\ & 1 & & & \\ & 5 & & 2 & \\ 1 & & & & 3 \\ & & 2 & 1 & \end{bmatrix} \Rightarrow \begin{bmatrix} -1.1 & 3.2 & 4.3 \\ 0.2 & 1.4 & 3.1 \\ 2.5 & 0.3 & 2.3 \\ 4.3 & -2.4 & 0.5 \\ 3.6 & 0.3 & 1.2 \end{bmatrix} \times \begin{bmatrix} 0.2 & 1.7 & 2.3 \\ 1.9 & 0.4 & 0.8 \\ 1.5 & -1.2 & 0.3 \\ -0.4 & 2.1 & 0.6 \\ 2.7 & 0.8 & 1.4 \end{bmatrix}$$

Recommendation engines

Recommendation is ranking

- Given a user and context, rank the available items in order of likelihood that the user will interact with them

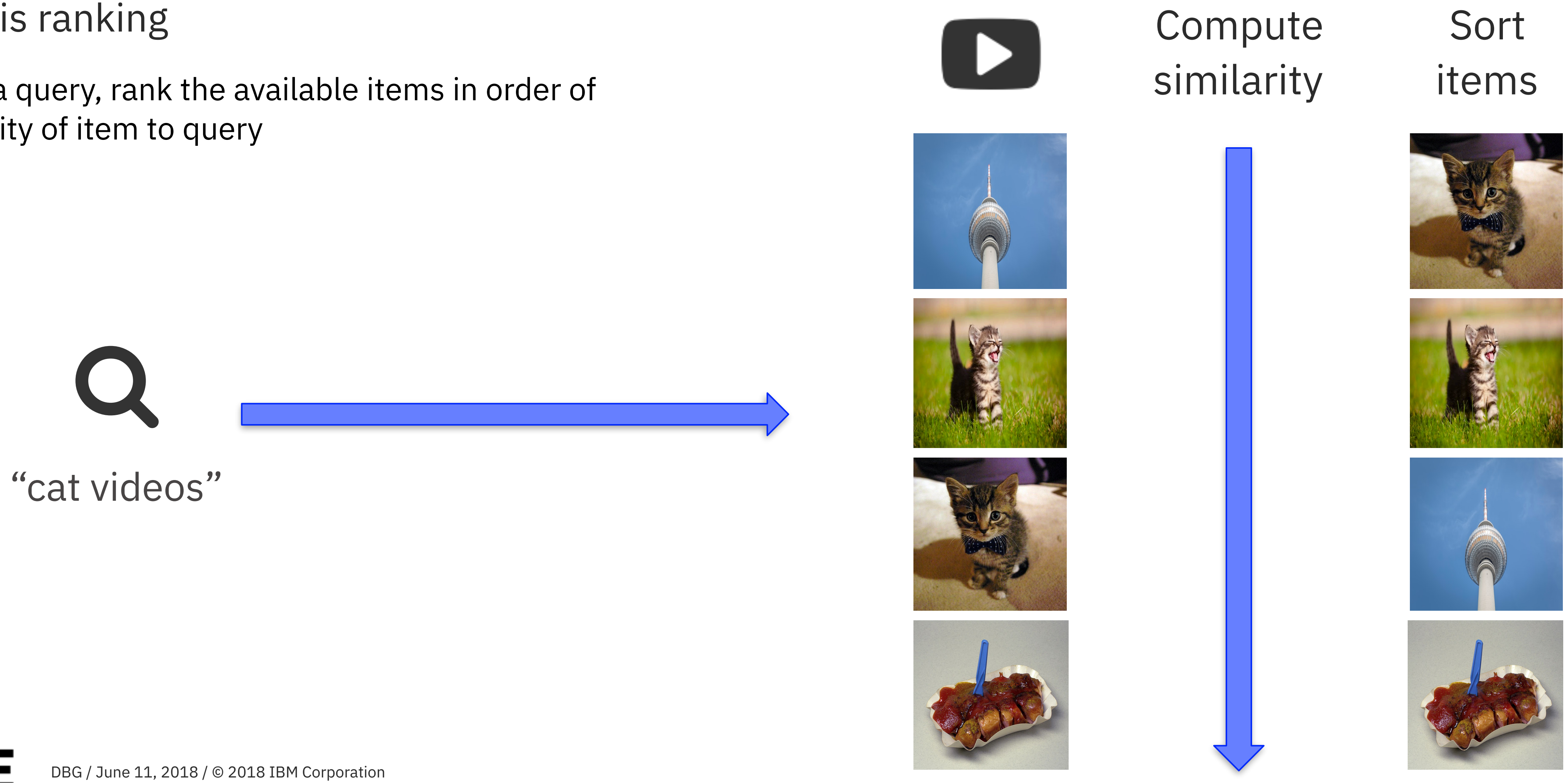




Search engines

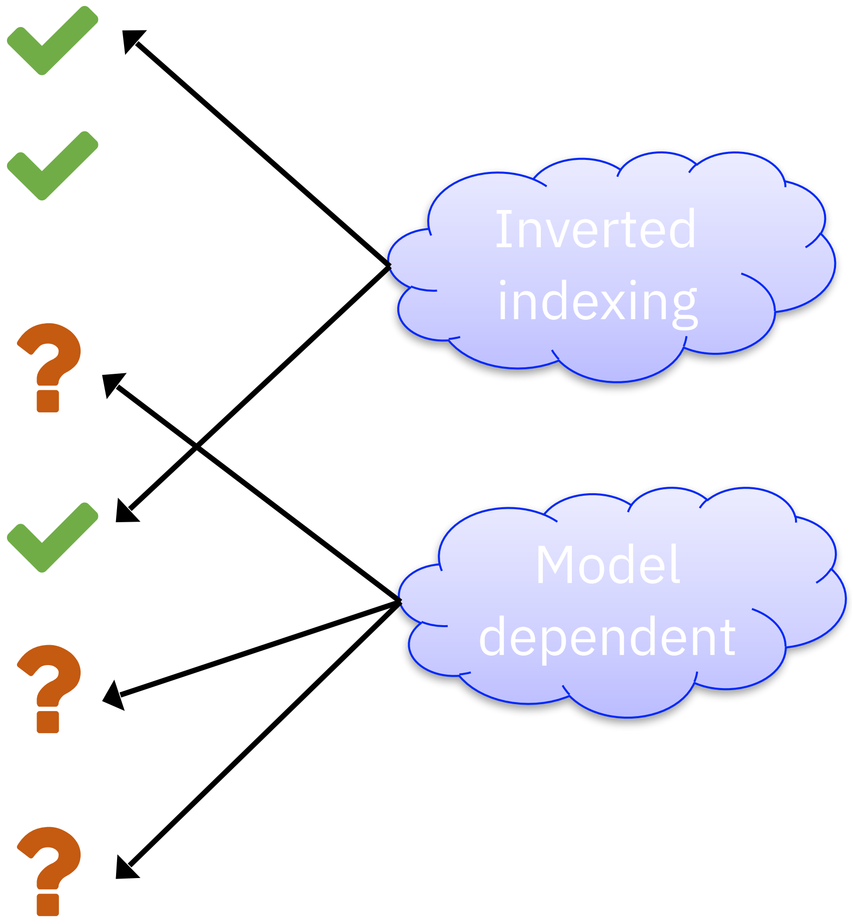
Search is ranking

– Given a query, rank the available items in order of similarity of item to query



Meeting our Requirements?

- Serving => ranking large # items
- Often need to filter
  - categories, popularity, price, geo, time
- Use all data at prediction time
  - content, context, preference, session
- Need to scale with item set and feature set
- Handle cold start
  - content-based models or fallbacks, item-aggregates
- Easily incorporate new preference data
  - model re-training, fold in



3 Sides of the Coin

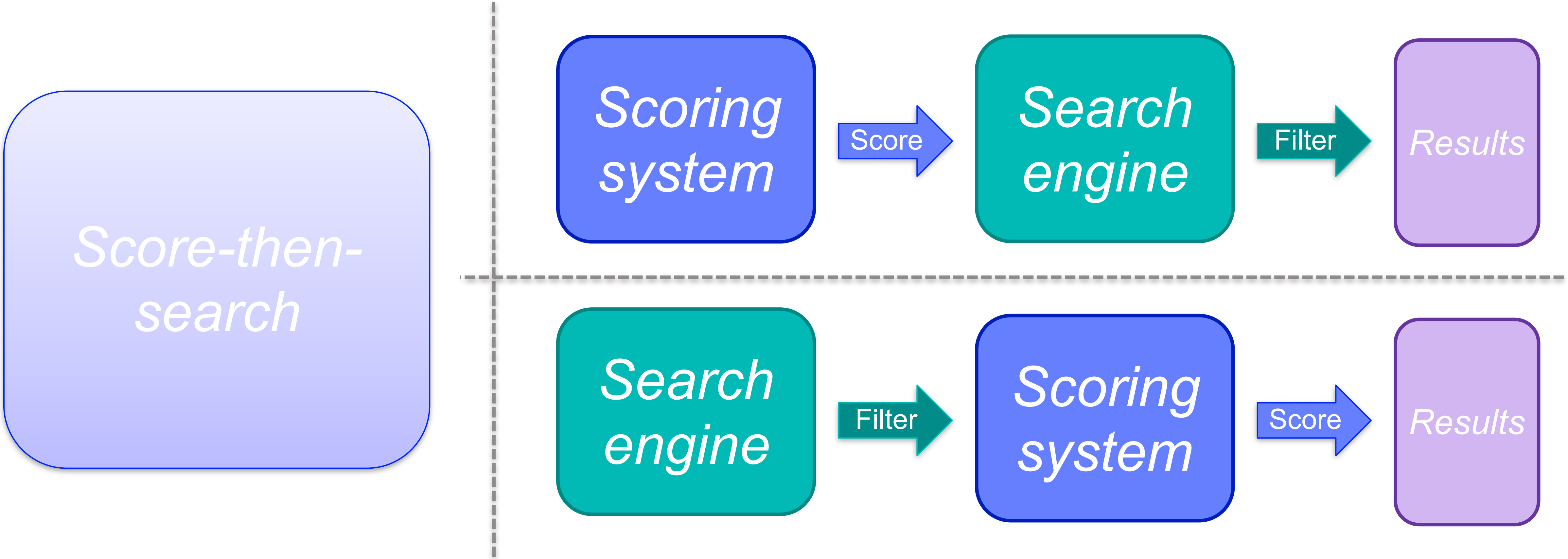
Broad approaches

Score-then-search

Native search

Custom ranking

Score-then-search



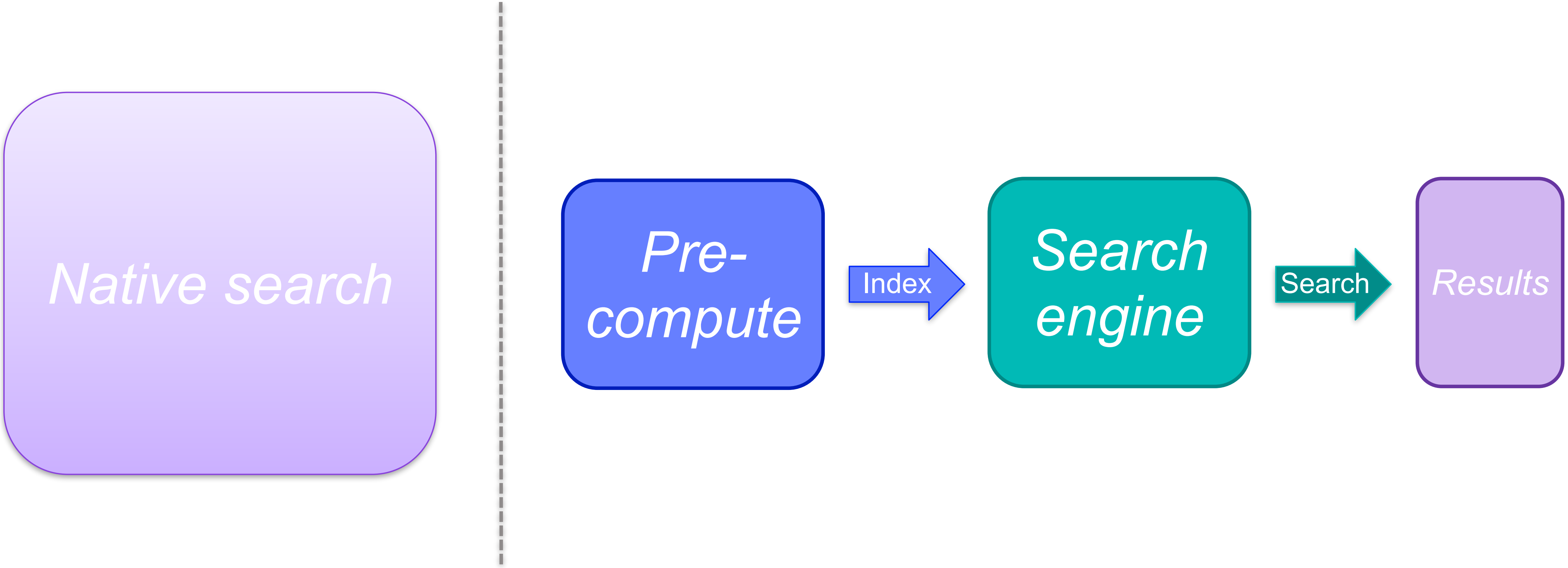
Score-then-search

Score-then-search

- Complete flexibility in model
- Easier to incorporate richer content features
- Can optimize scoring component

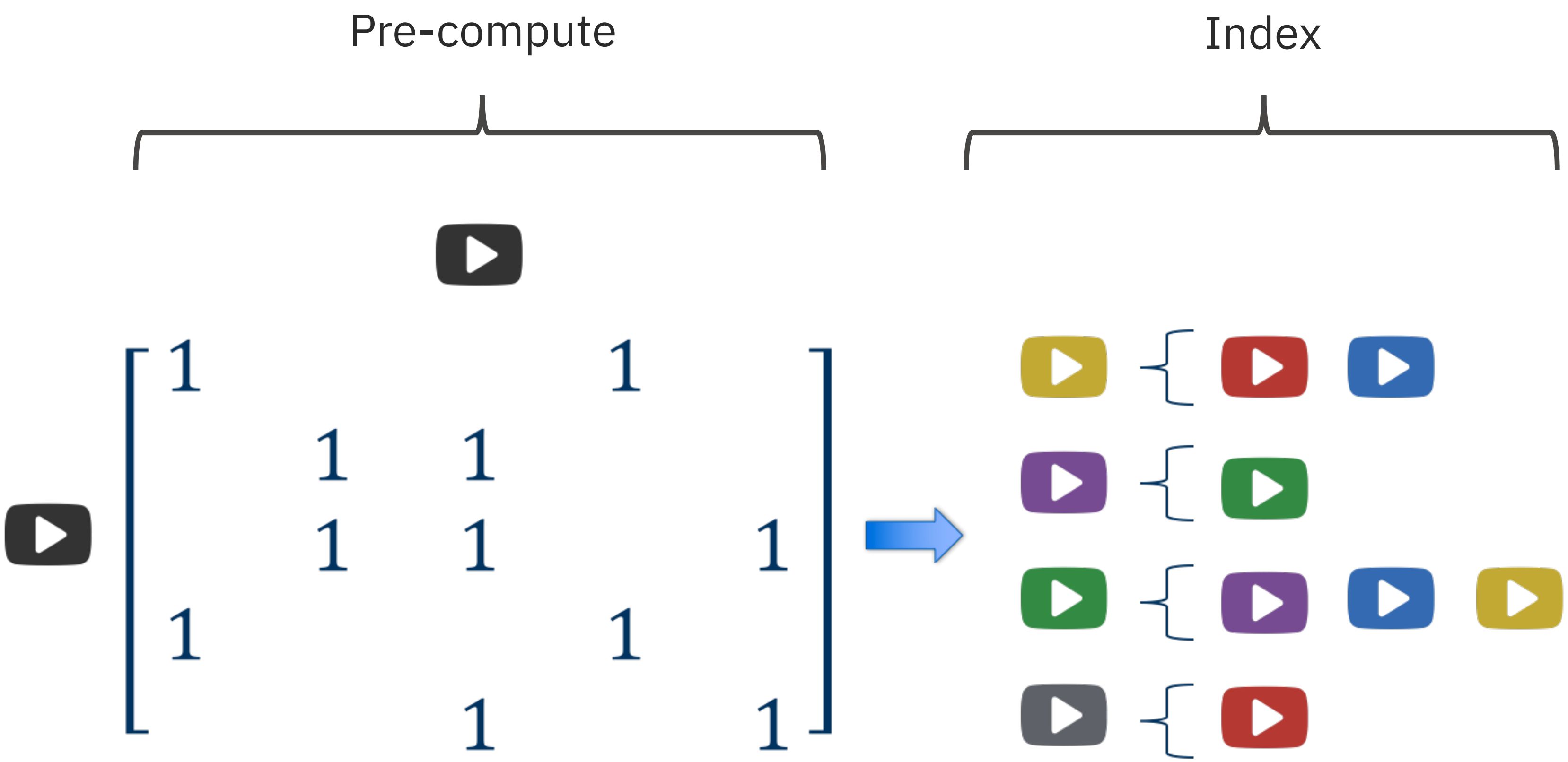
- Maintain (at least) 2 systems
- Filtering challenges
- Round trips between systems

Native Search

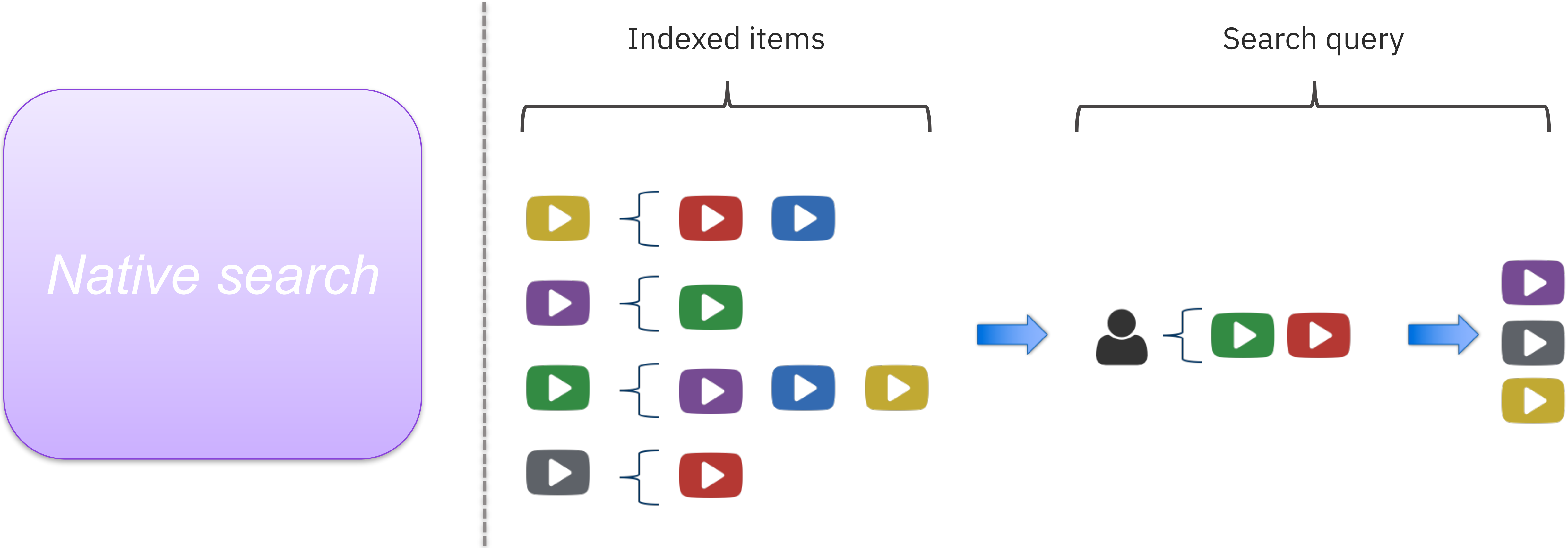


Native Search

Native search



Native Search



Native Search

Native search

- No changes to search engine required
- Very fast & flexible at query time
- Scalable - pre-compute + thresholding
- Can use (almost) all data

- Must decide what to pre-compute
- No ordering retained in indexed terms
- More difficult to include richer content data (image, audio)

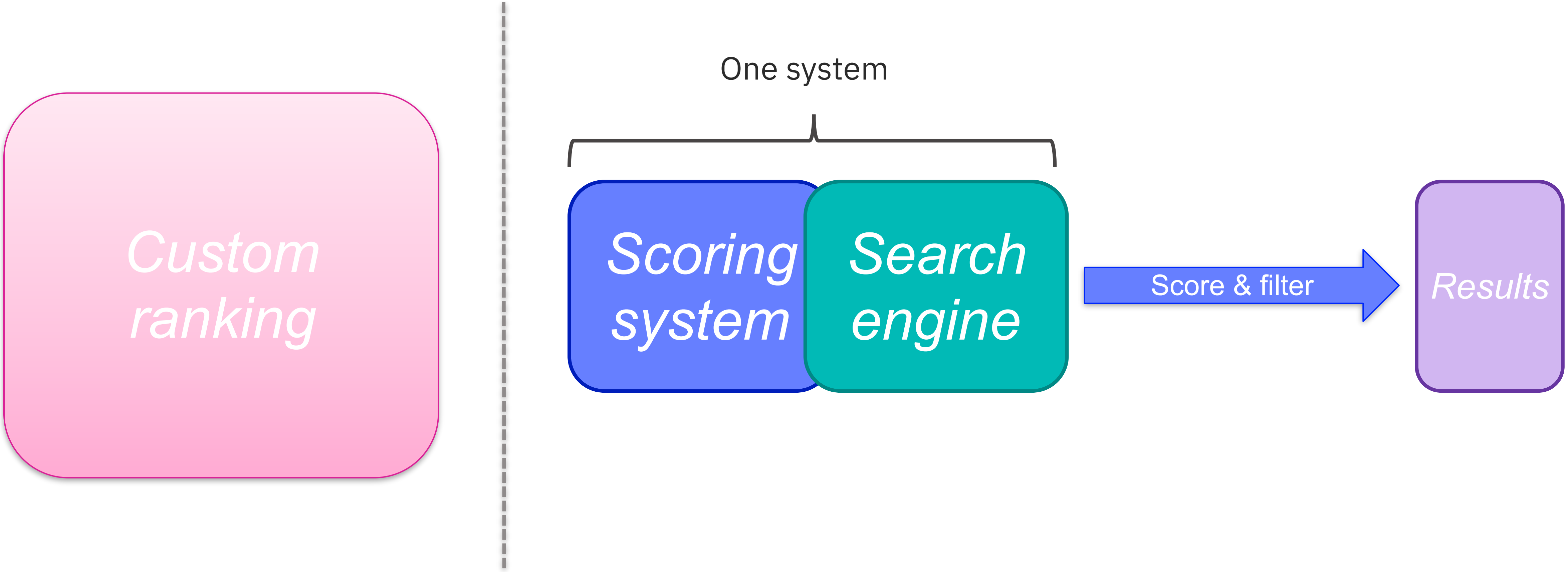
Native Search

Native search

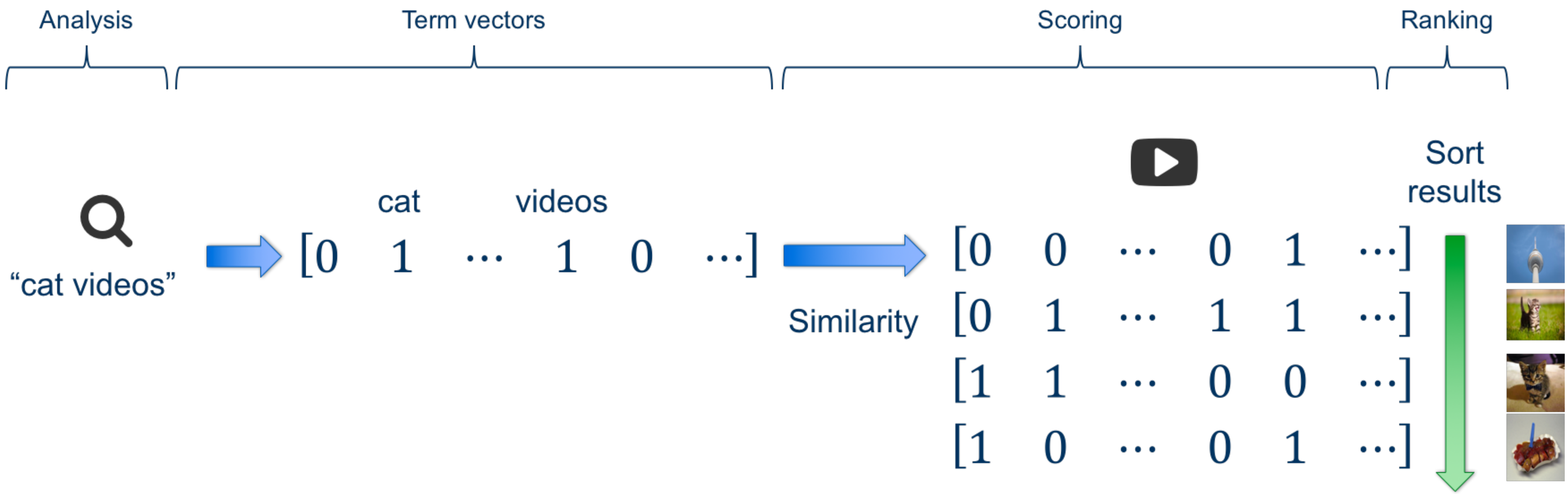
[Universal recommender]()

[The Mahout Correlated Cross-Occurrence Algorithm]()

Custom Ranking



Search Ranking



Can we use the same machinery?

Analysis

Term vectors

Scoring

Ranking

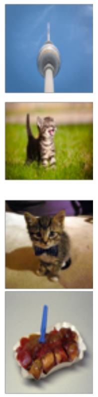
User
(or item)
vector

$$[1.2 \quad \cdots \quad -0.2 \quad 0.3]$$

Similarity

$$\begin{matrix} [0 & 0 & \cdots & 0 & 1 & \cdots] \\ [0 & 1 & \cdots & 1 & 1 & \cdots] \\ [1 & 1 & \cdots & 0 & 0 & \cdots] \\ [1 & 0 & \cdots & 0 & 1 & \cdots] \end{matrix}$$

Sort
results



Dot product & cosine similarity
… the same as we need for recommendations!

Delimited Payload Filter

Raw vector

$$[1.2 \quad \cdots \quad -0.2 \quad 0.3]$$

Custom analyzer

0|1.2 ··· 3|-0.2 4|0.3

Term vector with payloads

```
'terms': {
  '0': {
    'term_freq': 1,
     'tokens': [
       {
         'payload': 'P5mZmg==',
         'position': 0
       }
     ]
  },
```

Custom Scoring Function

– Native script (Java), compiled for speed

– Scoring function computes dot product by:

  - For each document vector index (“term”), retrieve payload
  - Score += payload * query(i)

– Normalizes with query vector norm and document vector norm for cosine similarity

```
{
  "function_score": {
    "query" : {
      ...
    },
    "script_score": {
      "script": "payload_vector_score",
      "lang": "native",
      "params": {
        "field": "@model.factor",
        "vector": [1.2,...,-0.2,0.3],
        "cosine" : True
      }
    },
    "boost_mode": "replace"
  }
}
```

Can we use the same machinery?

Get search for free!

```
{
  "function_score": {
    "query" : {
      ...
    },
    "script_score": {
      "script": "payload_vector_score",
      "lang": "native",
      "params": {
        "field": "@model.factor",
        "vector": [1.2,...,-0.2,0.3],
        "cosine" : True
      }
    },
    "boost_mode": "replace"
  }
}
```

```
{
  "item_id": "10",
  "name": "LOL Cats",
  "description": "catscatscats",
  "category": ["Cat Videos", "Humour", "Animals"],
  "tags": ["cat", "lol", "funny", "cats", "felines"],
  "created_date": 1476884080,
  "updated_date": 1476884080,
  "last_played_date": 1476946962,
  "likes": 100000,
  "author_id": "321",
  "author_name": "ilikecats",
  "channel_id": "CatVideoCentral",
  ...
}
```

Custom Ranking

Custom ranking

- Combine search & scoring into one system
- Potential to incorporate richer content features & contextual models
- Combine best of both worlds between pre-computing & online scoring (sort of)

- Requires custom plugin for your search engine
- Scaling limits & performance considerations
- Must decide how to blend interactions (e.g. weights)

Custom Ranking

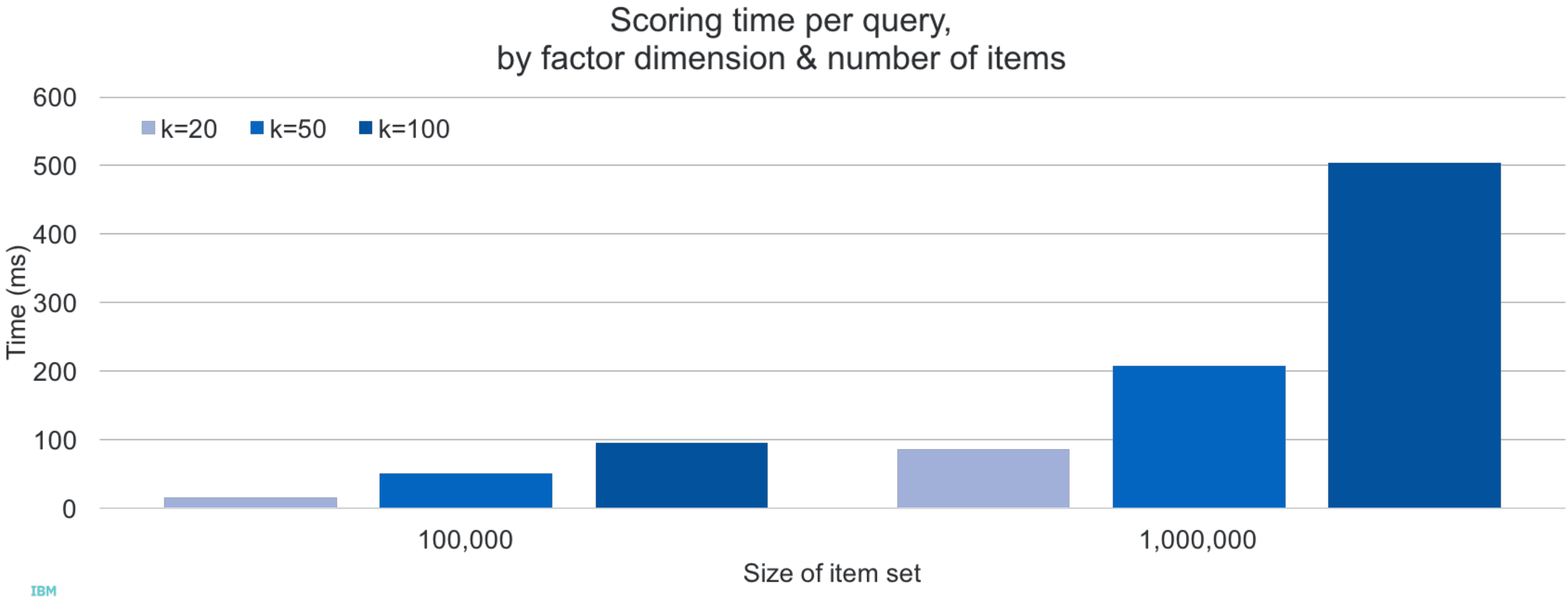
Custom ranking

[Spark & Elasticsearch Recommender on IBM Code]

[Elasticsearch vector scoring plugin]

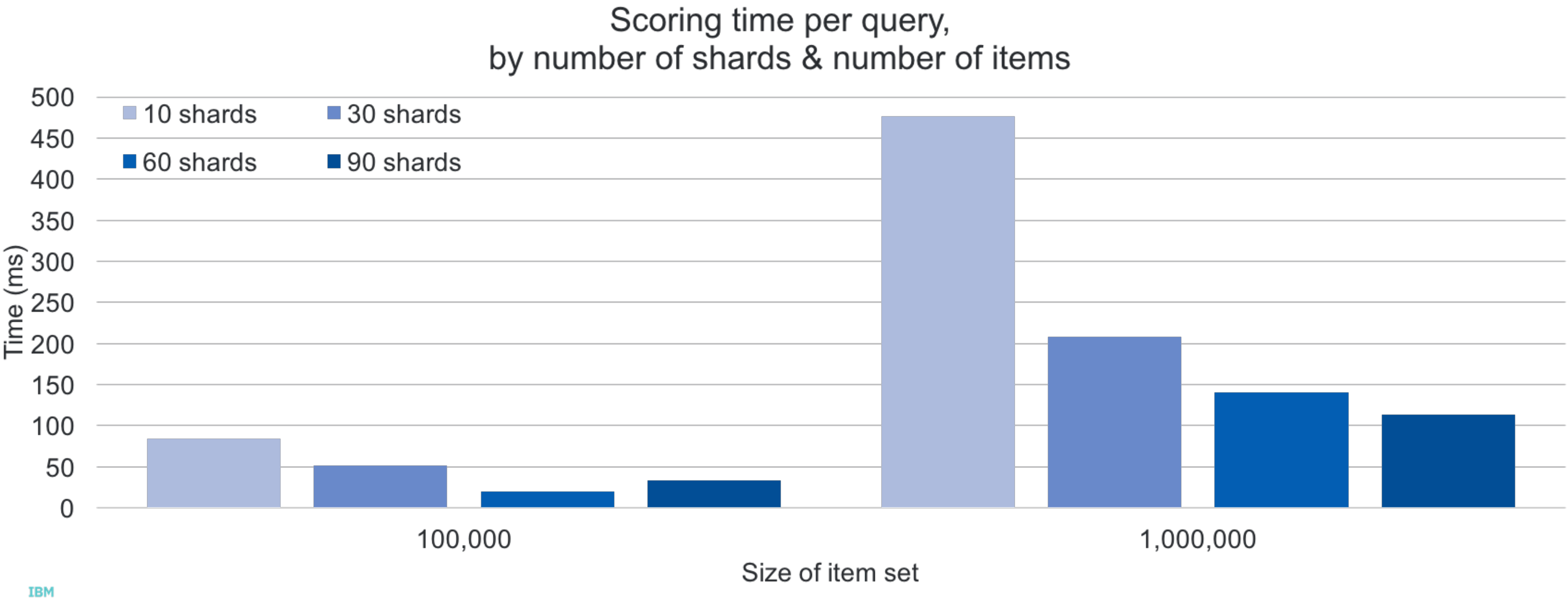
Performance

Custom Scoring Performance



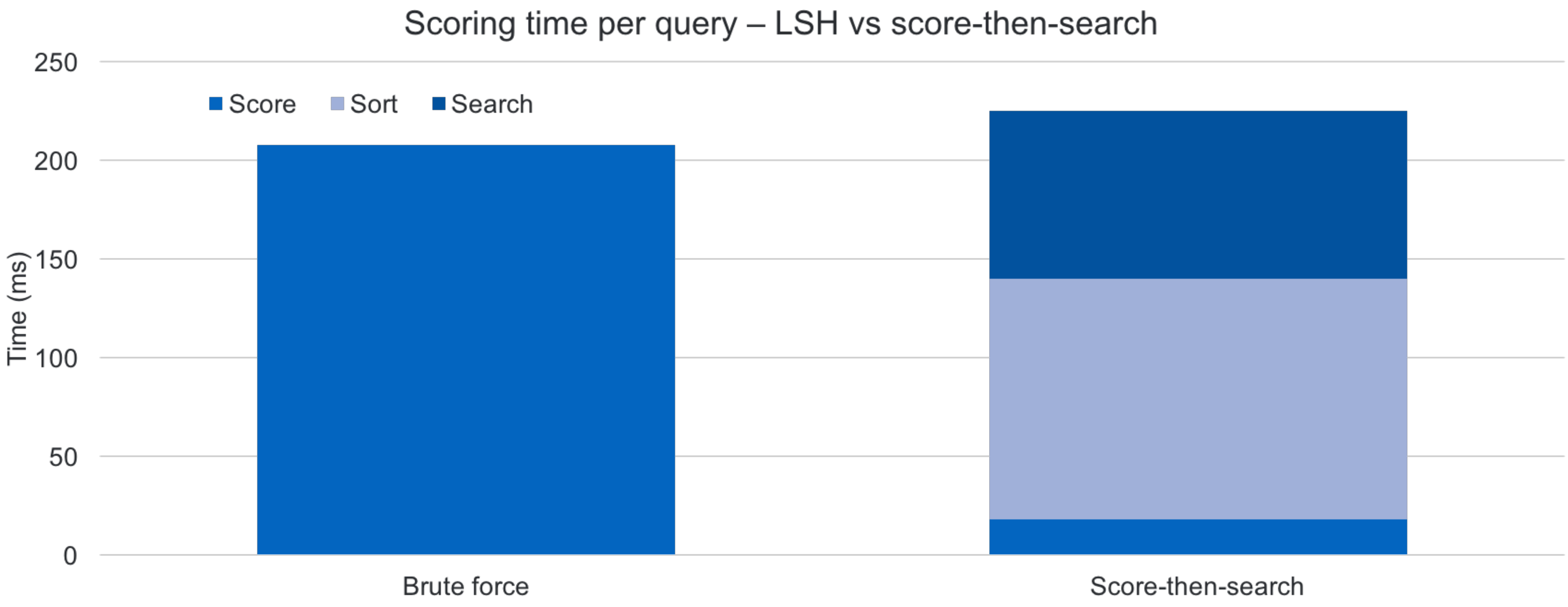
*3x nodes, 30x shards

Custom Scoring Performance



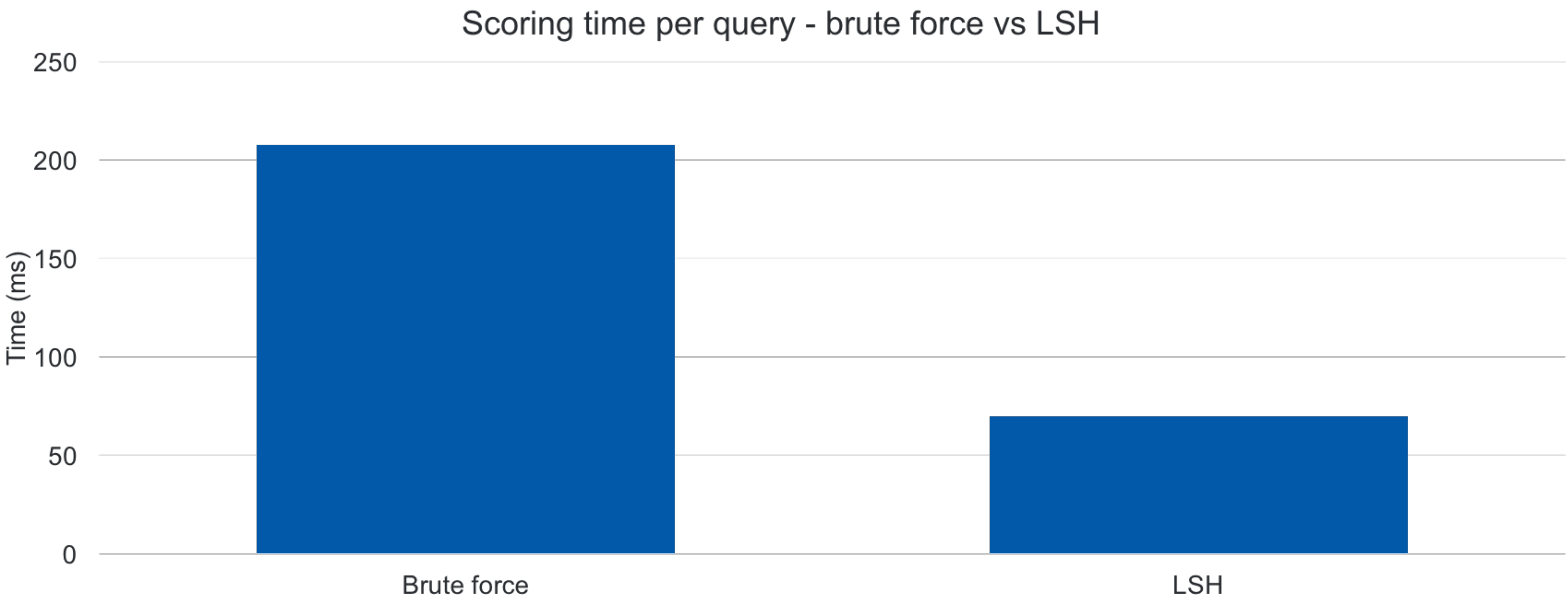
**3x nodes, k=50*

Comparison to Score-then-search



*3x nodes, 30x shards, k=50, 1,000,000 items

Scaling custom scoring - LSH



**3x nodes, 30x shards, k=50, 1,000,000 items*

Pure Search Approaches

“Pure” search engine approaches

- More like this
  - Content similarity
- Significant terms queries
  - 2 stage query:
    - 1st query interactions to get set of items for a user
    - 2nd significant terms aggregation with the item set as background
  - Result is very similar to co-occurrence approaches
  - Round trips may be slow

Conclusion

Custom Ranking – Future Directions

- Apache Solr version
  - https://github.com/saaay71/solr-vector-scoring
- Improve performance
  - Improved scoring performance for vector scoring plugin: https://github.com/lior-k/fast-elasticsearch-vector-scoring
  - Investigate performance of LSH-filtered scoring
  - Dig deeper into Lucene internals to combine matrix-vector math with search & filter?
- Investigate more complex models

Summary

All approaches have different tradeoffs

Max model flexibility | Balanced | Max search integration

Score-then-search

Custom ranking

Native search

Thank you

[codait.org]

[twitter.com/MLnick]

[github.com/MLnick]

[developer.ibm.com/code]

FfDL

MAX

Sign up for IBM Cloud and try Watson Studio!

https://ibm.biz/BdZ6qf

IBM Code Pattern

https://datascience.ibm.com/

CALL FOR CODE
GLOBAL INITIATIVE 2018

Commit to the cause. Push for change.

Call for Code inspires developers to solve **pressing global problems** with **sustainable software solutions**, delivering on their vast potential to do good.

Bringing together NGOs, academic institutions, enterprises, and startup developers to compete build effective **disaster mitigation solutions**, with a focus on health and well-being.

International Federation of Red Cross/Red Crescent, **The American Red Cross**, and the **United Nations Office of Human Rights** combine for the ***Call for Code Award*** *to* elevate the profile of developers.

Award winners will receive **long-term support** through **open source foundations**, **financial prizes**, the **opportunity to present their solution to leading VCs**, and will deploy their solution through **IBM's Corporate Service Corps**.

Developers will jump-start their project with dedicated **IBM Code Patterns**, combined with **optional enterprise technology** to build projects over the course of three months.

Judged by the world's most **renowned technologists**, the **grand prize** will be presented in **October** at an Award Event.

developer.ibm.com/callforcode

Links & References

[Spark & Elasticsearch Recommender on IBM Code]

[Elasticsearch vector scoring plugin]

[Solr vector scoring plugin]

[Improved performance Elasticsearch plugin]

[Elasticsearch More Like This Query]

[Elasticsearch significant terms aggregation]

[Universal recommender]

[The Mahout Correlated Cross-Occurrence Algorithm]