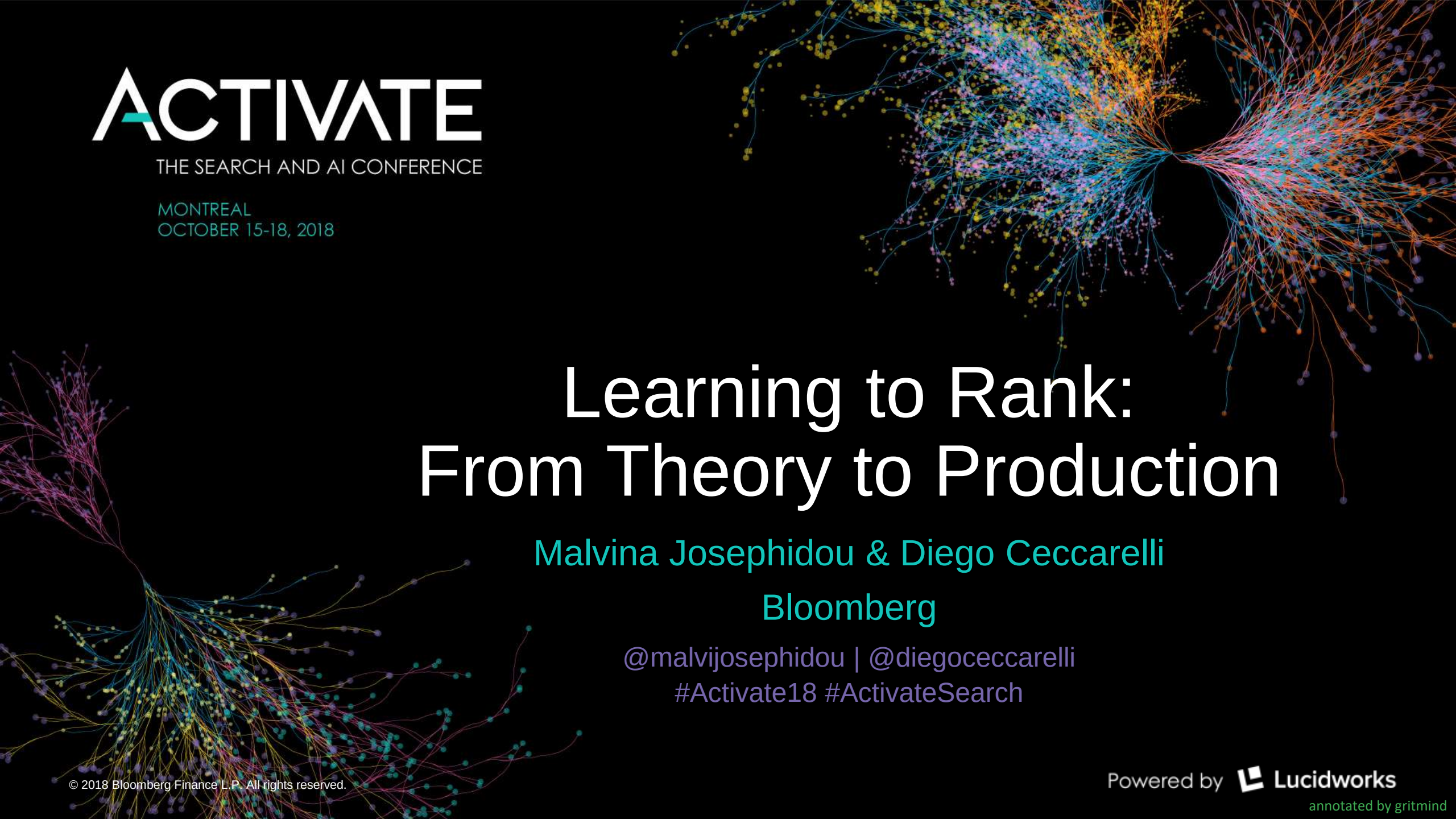


Learning to Rank: From Theory to Production

Malvina Josephidou & Diego Ceccarelli

Bloomberg

@malvijosephidou | @diegoceccarelli

#Activate18 #ActivateSearch

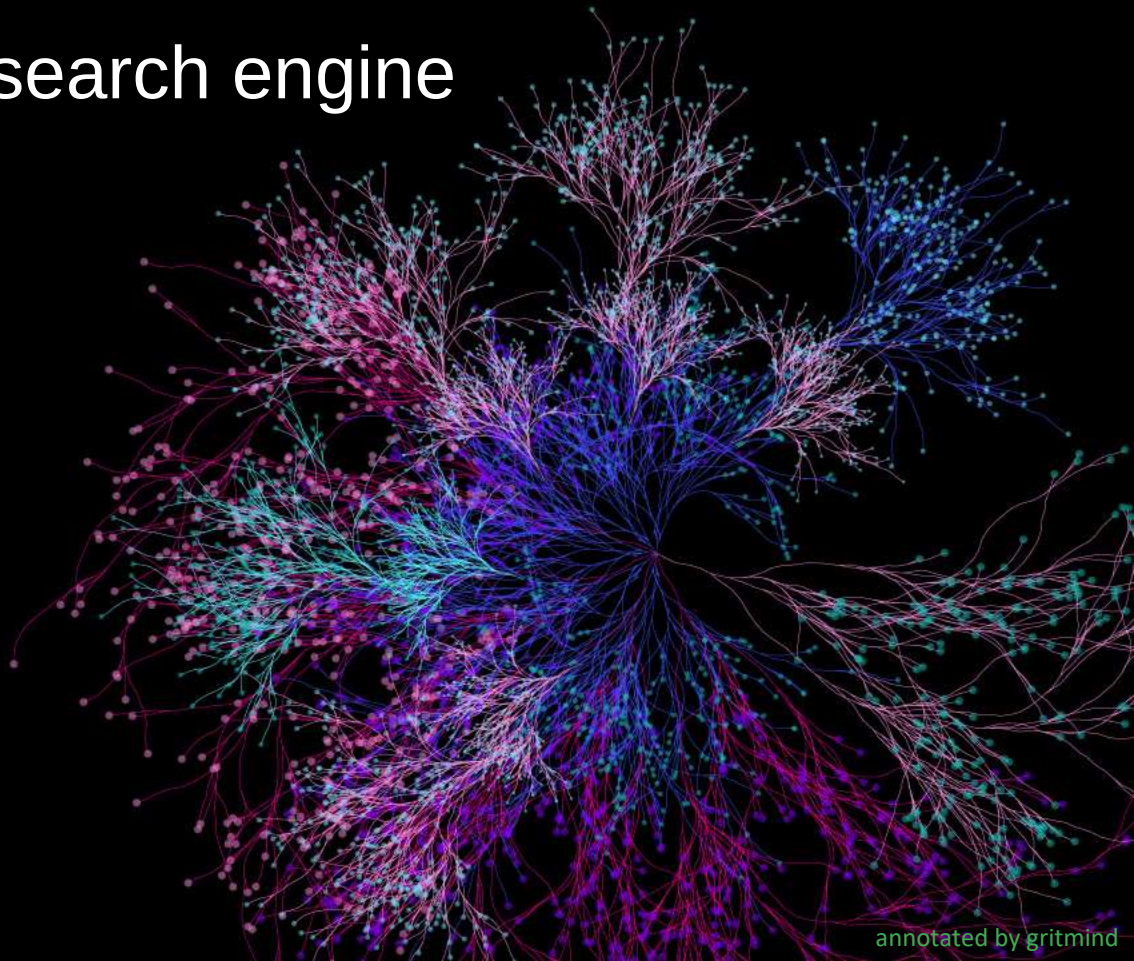
Powered by Lucidworks

annotated by gritmind

About Us

ACTIVATE

- Software Engineers at Bloomberg
- Working on relevance of the News search engine
- Before joining, PhDs in ML and IR



annotated by gritmind

Bloomberg – Who are we?

- A technology company with 5,000+ software engineers
- Financial data, analytics, communication and trading tools
- More than 325K subscribers in 170 countries

325K+

Terminal subscribers

News search

16M

queries per day

Stories available for search in

~100 ms

Average query response time

<200 ms

News volume

2M

stories per day

500

stories ingested per second

650M

stories in index

News alerting

1.5M

subscriptions

500

stories matched per second

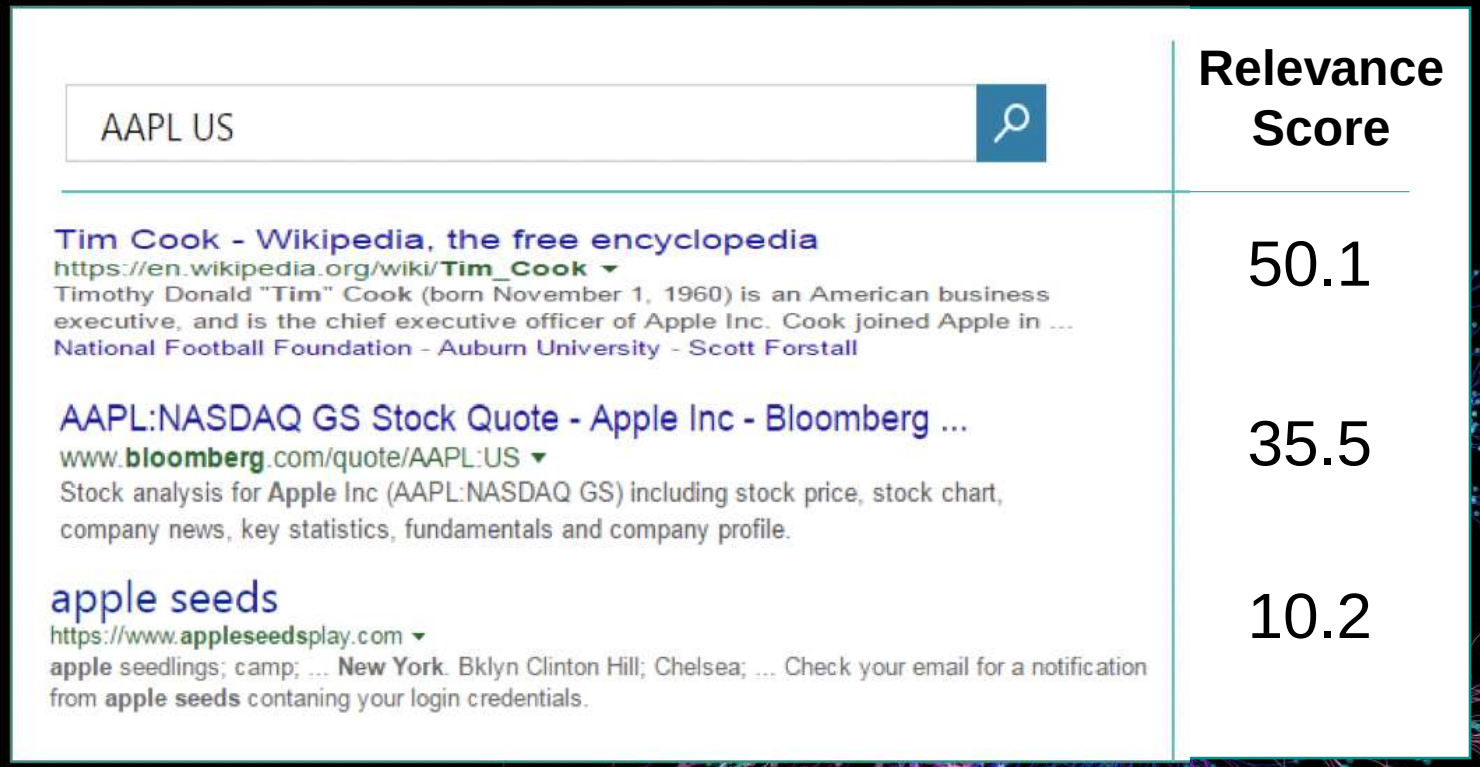
Alerts delivered in

<100ms

How do we retrieve relevant results?

ACTIVATE

- Use *relevance functions* to assign scores to each matching document
- Sort documents by relevance score

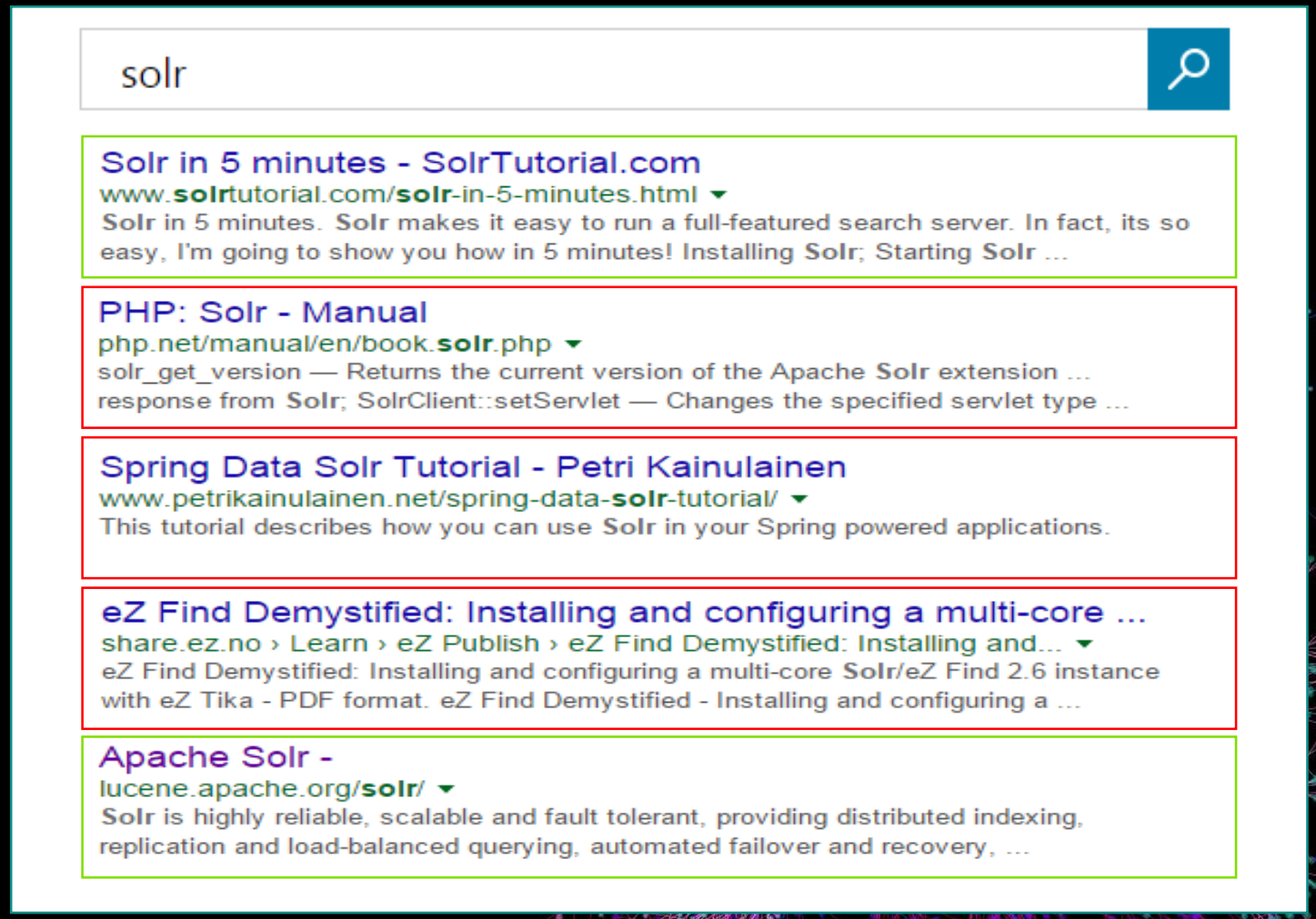


annotated by gritmind

How do we design relevance functions?

ACTIVATE

```
score = tf
      + 5.2 x tf(title)
      + 4.5 x tf(desc)
```



annotated by gritmind

Good Luck With That…

ACTIVATE

query = Solr

query = Italy

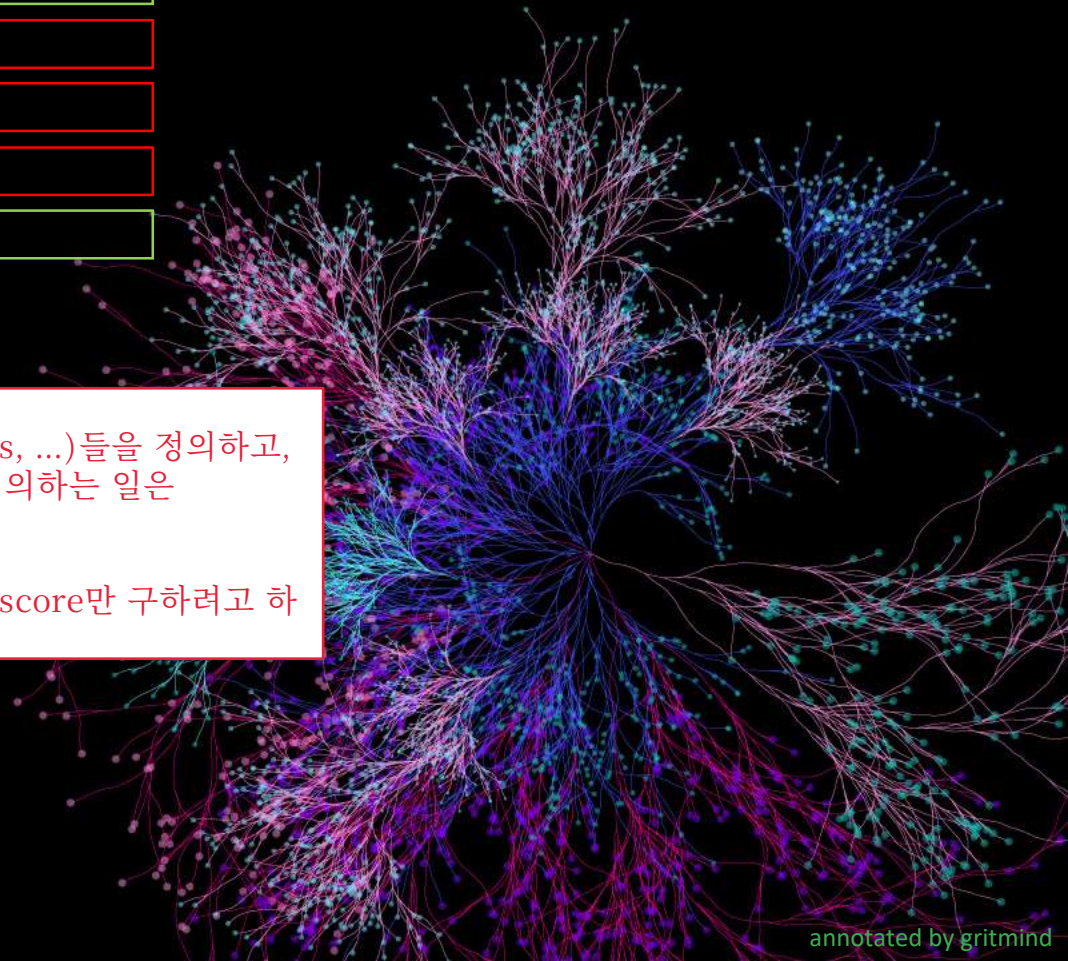
query = Facebook

query = Trump

```
score = tf(body)
      + 5.2 x tf(title)
      + 4.5 x tf(desc)
      + ??? x doc-length
      + ??? x freshness
      + ??? x popularity
      + ??? x author
      + ..... ?????
```

사람이 일일이
feature (e.g. freshness, ...)들을 정의하고,
feature마다 score를 정의하는 일은
쉽지 않은 일이다.

여기서는 기계학습으로 score만 구하려고 하는 것 같다.



annotated by gritmind

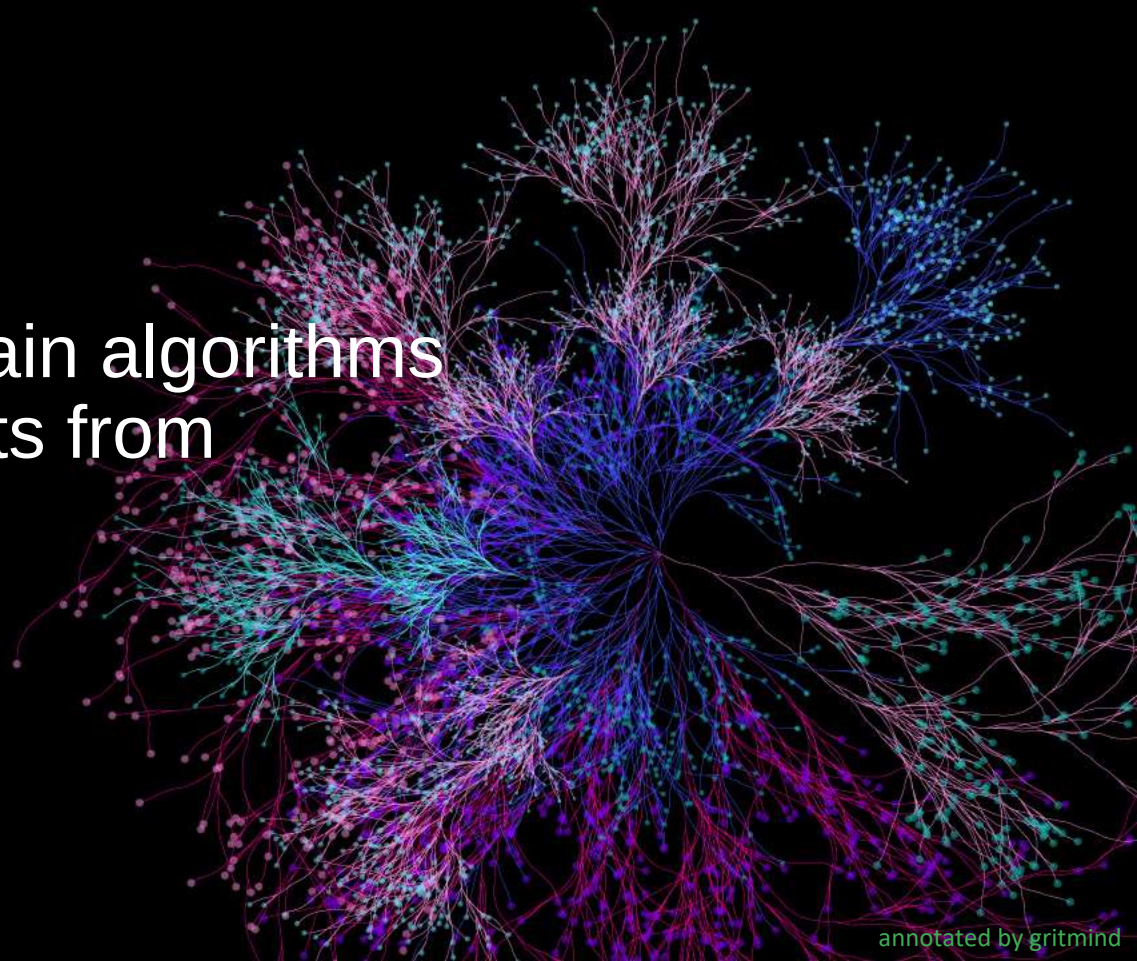
How do we come up with ranking functions?

ACTIVATE

- We don’t. Hand-tuning ranking functions is insane

전형적으로 기계학습이
잘 할 수 있는 일이다.

- ML to the rescue: Use data to train algorithms to distinguish *relevant* documents from *irrelevant* documents



annotated by gritmind

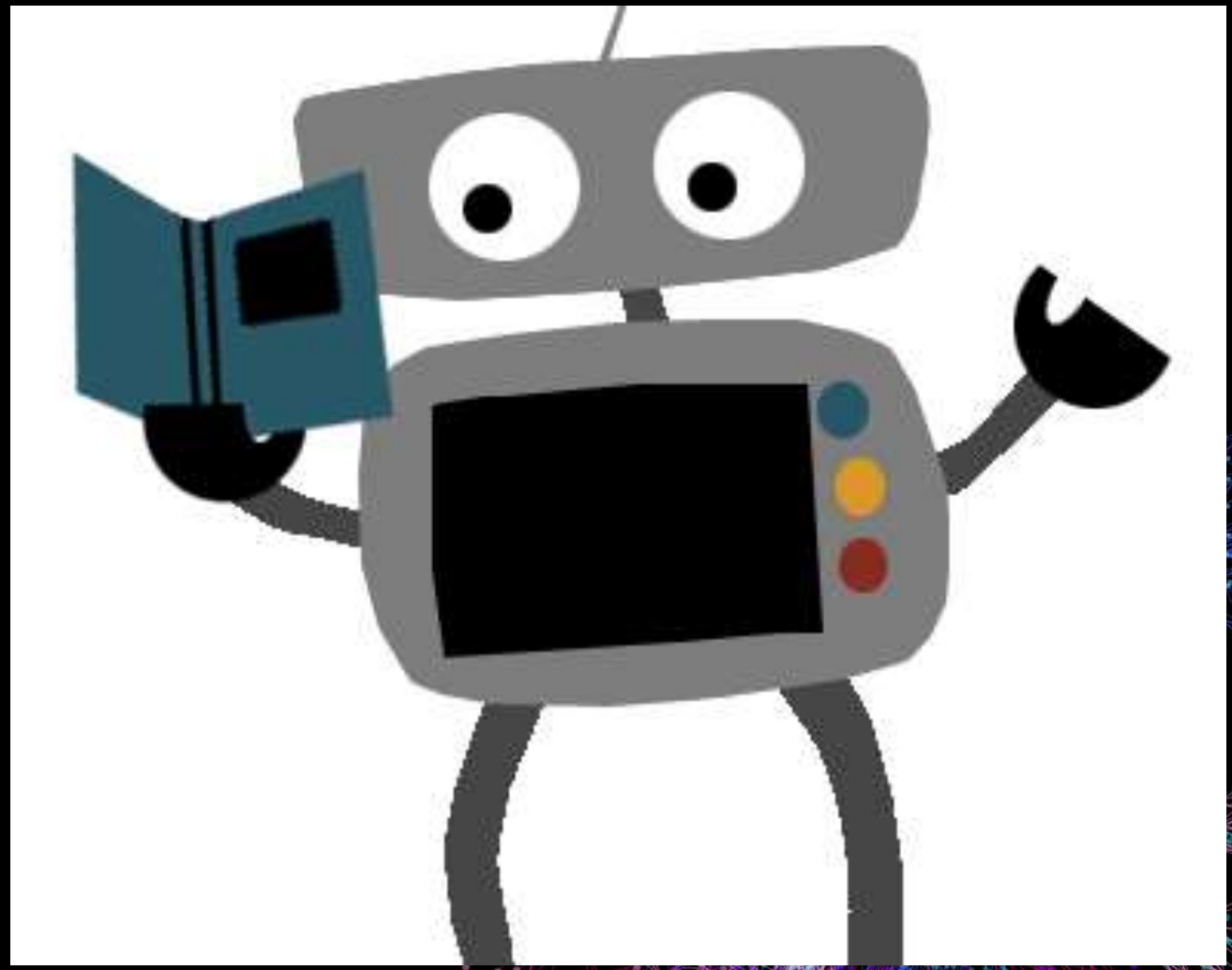
2018 achievement

Learning-to-Rank fully deployed in production

ACTIVATE

Learning-to-Rank (aka LTR)

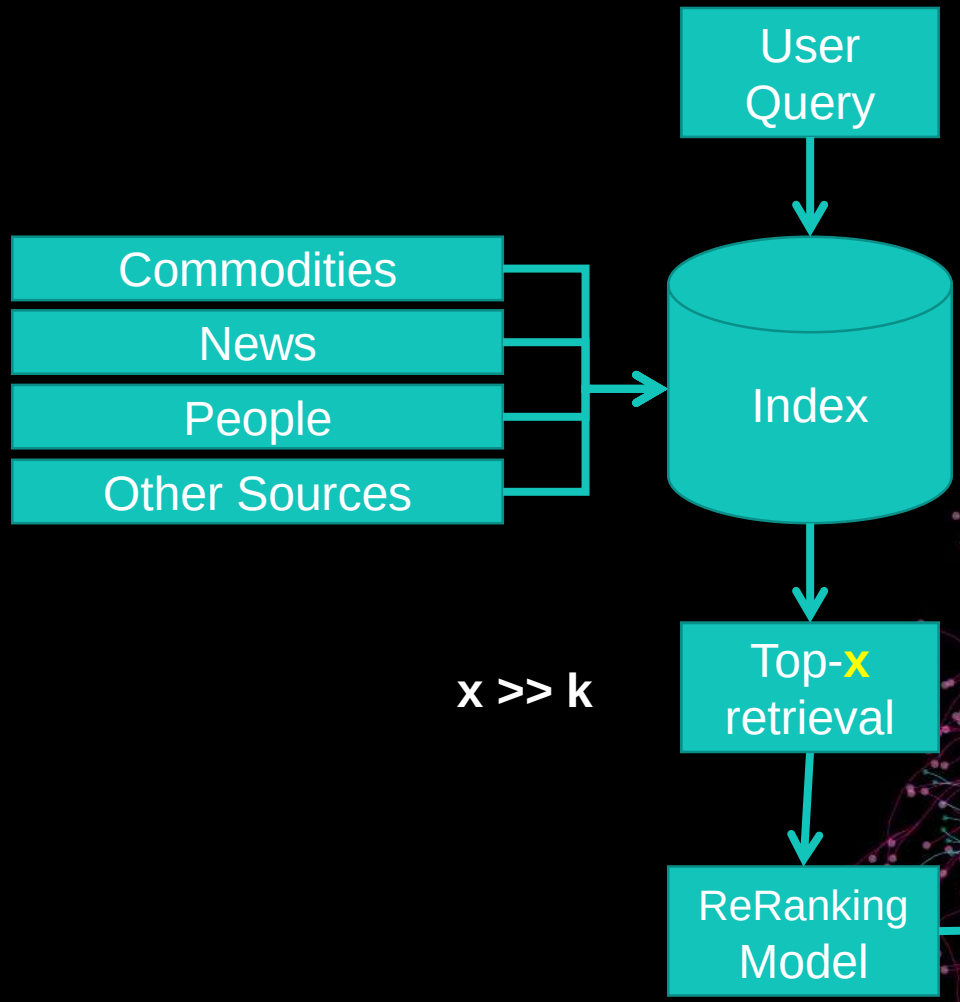
Use machine learning algorithms to rank results in a way that optimizes search relevance



annotated by gritmind

How does this work?

ACTIVATE



문제를 좀 더 세분화한다는 느낌이 있다.

Index 단계는 매칭이고,
Top - x retrieval 단계는 일반적인 랭킹,
ReRanking은 좀 더 디테일한 랭킹인 것 같다.

ReRanking 모델은 좀 더 복잡한 함수로서 뉴럴넷이나 SVM 등이 될 수 있다. 이들은 계산량이 크므로 입력이 적을수록 좋다.
그래서 Top - x 다음에 Reranking이 있는 것이 좋다. 앞에서 한 번 걸러주기 때문에 입력크기는 줄어든다.

annotated by gritmind

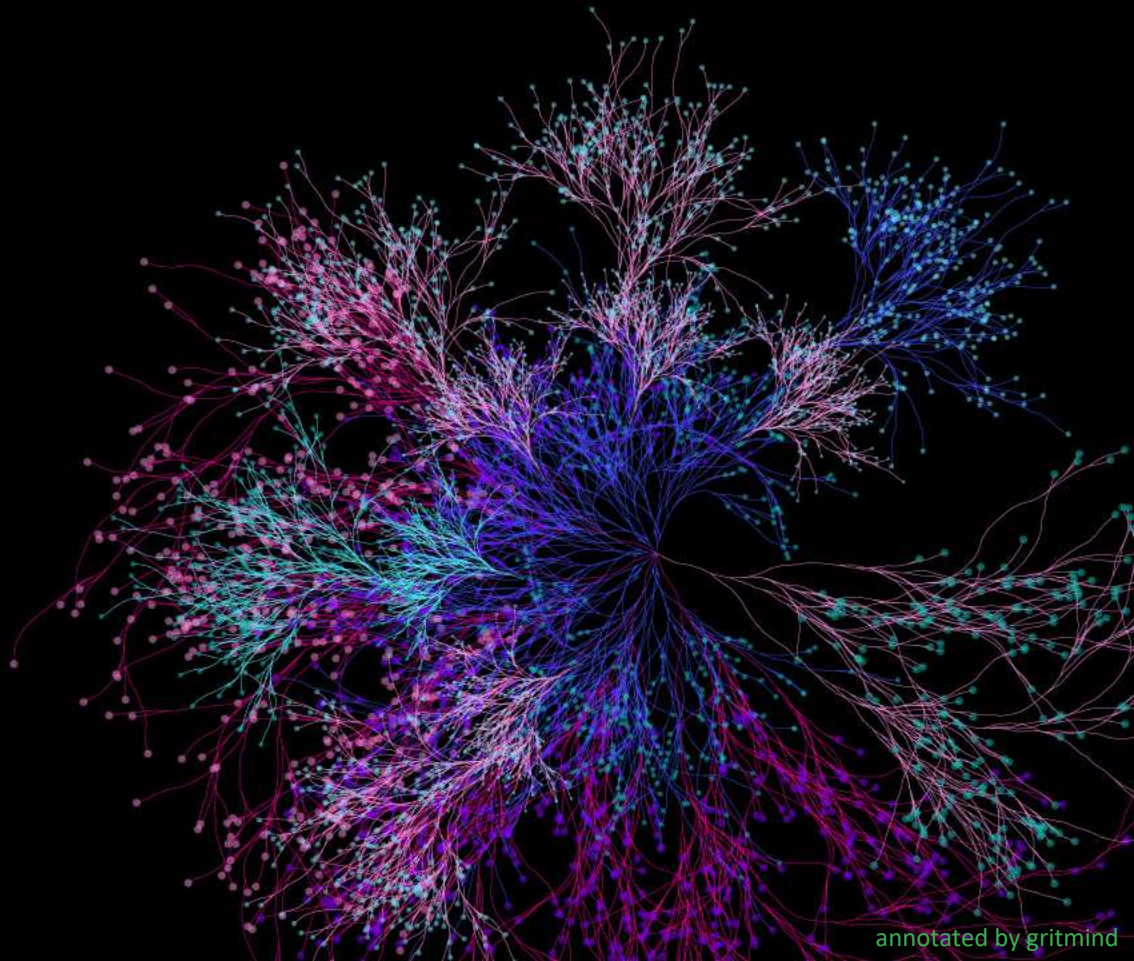
How to Ship LTR in Production in 3 Steps

ACTIVATE

Make it Work

Make it Fast

Deploy to Production



annotated by gritmind

LTR steps

ACTIVATE

I. Collect query-document judgments [Offline]

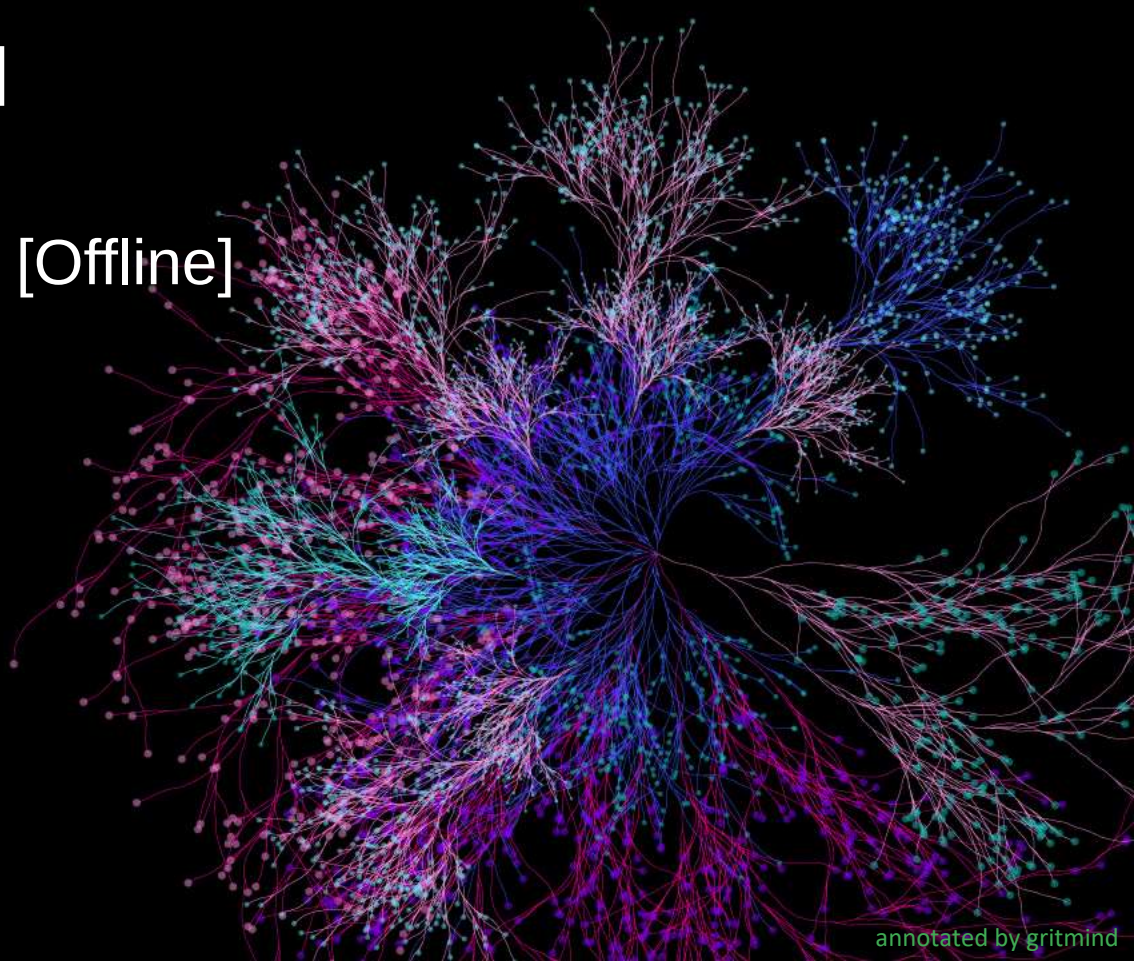
II. Extract query-document features [Solr]

III. Train model with judgments + features [Offline]

IV. Deploy model [Solr]

V. Apply model [Solr]

VI. Evaluate results [Offline]



annotated by gritmind

I. Collect judgements

첫 번째는
데이터 수집이다.

ACTIVATE

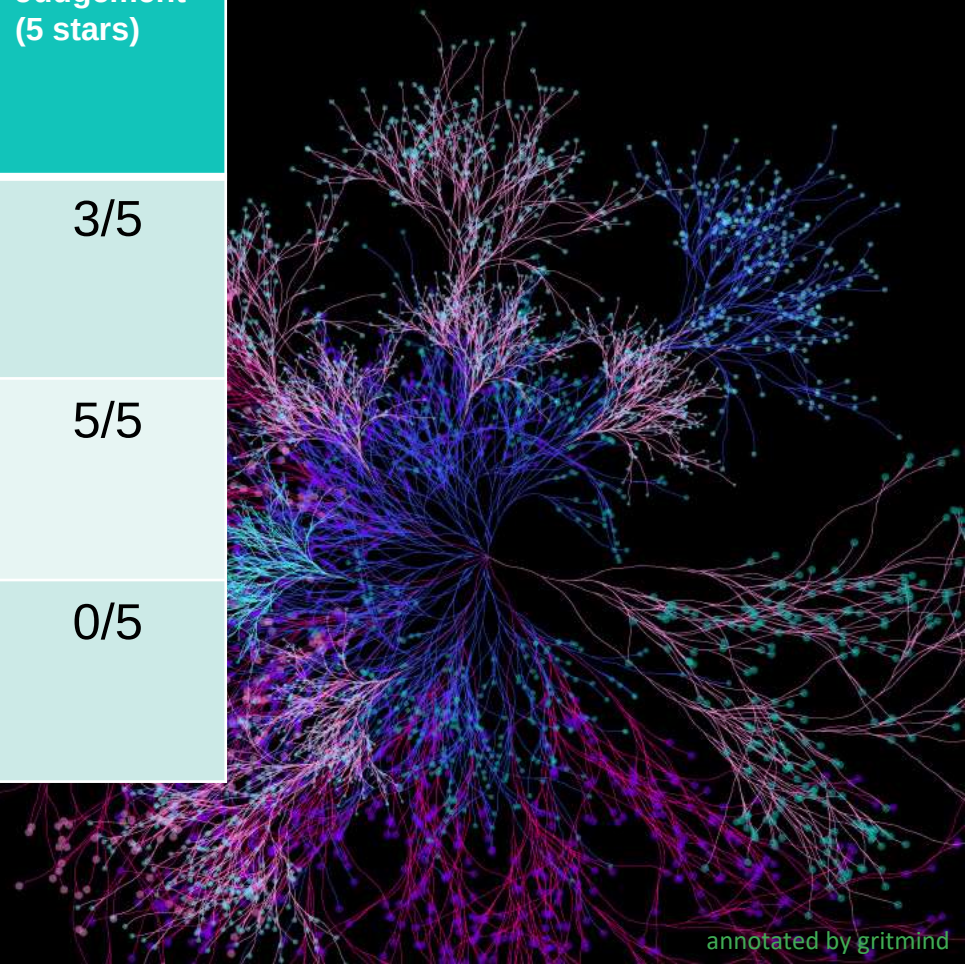
AAPL US

Tim Cook - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/**Tim_Cook**
Timothy Donald "**Tim**" **Cook** (born November 1, 1960) is an American business executive, and is the chief executive officer of Apple Inc. Cook joined Apple in ...
National Football Foundation - Auburn University - Scott Forstall

AAPL:NASDAQ GS Stock Quote - Apple Inc - Bloomberg ...
www.**bloomberg**.com/quote/AAPL:US
Stock analysis for **Apple** Inc (AAPL:NASDAQ GS) including stock price, stock chart, company news, key statistics, fundamentals and company profile.

apple seeds
https://www.**appleseeds**play.com
apple seedlings; camp; ... **New York**. Bklyn Clinton Hill; Chelsea; ... Check your email for a notification from **apple seeds** contaning your login credentials.

Judgement (good/bad)	Judgement (5 stars)
✓	3/5
✓	5/5
✗	0/5



annotated by gritmind

I. Collect judgements

ACTIVATE

Explicit – judges assess search results manually

- Experts
- Crowdsourced

비용이 비싸지만,
high quality data를 보장한다.

Implicit – infer assessments through user behavior

- Aggregated result clicks
- Query reformulation
- Dwell time

사용자들의 클릭과 같은 feedback을 data로 활용한다.

(사용자가 실수로 클릭하지 않는 이상) 사용자 클릭이 relevance를 보증할 수 있다. (쿼리와 클릭한 문서 사이의 relevance)

단점은, 데이터가 노이지하다.

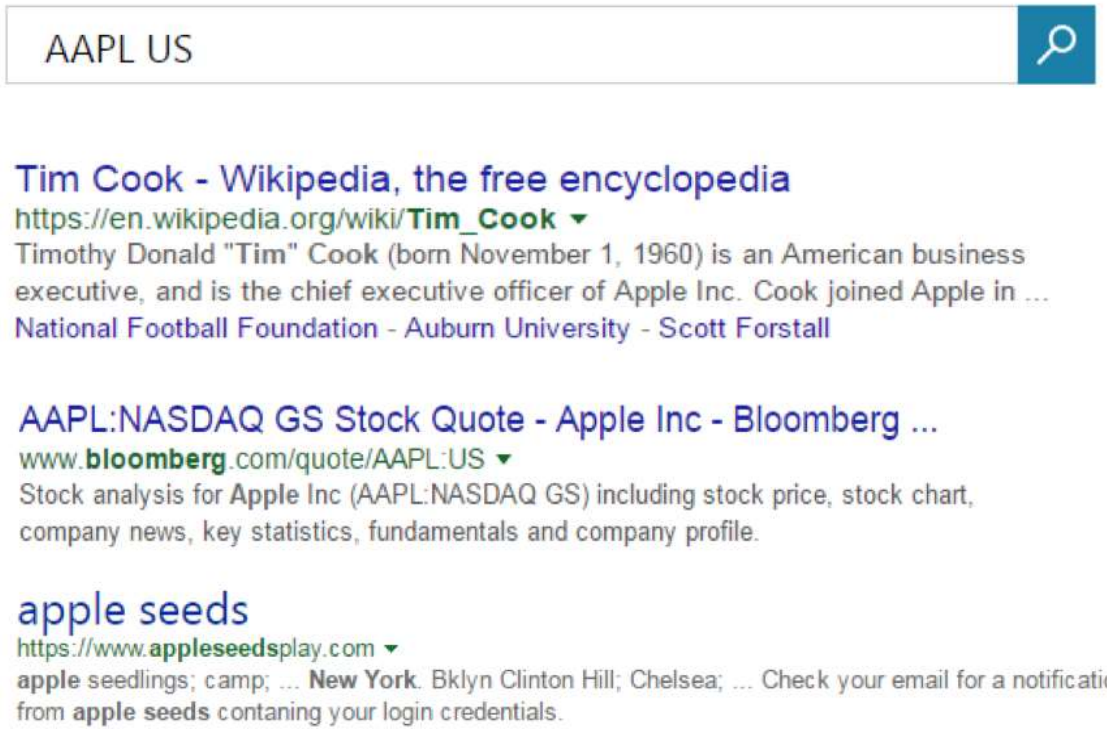
annotated by gritmind

II. Extract Features

3달 전 뉴스보다 어제 뉴스가 훨씬 더 중요하다 (Freshness)
사람들의 클릭 수 (Popularity)

ACTIVATE

Signals that give an indication of a result's importance



Query matches the title	Freshness	Is it from bloomberg.com?	Popularity
0	0.7	0	3583
1	0.9	1	625
0	0.1	0	129

annotated by gritmind

II. Extract Features

type: 해당 feature를 어떻게 계산하는지 정의되어 있는 자바 파일.

이런 식으로 solr에서는 ranking에 사용되는 feature를 정의하는 것 같다...

ACTIVATE

- Define features to extract in `myFeatures.json`
- Deploy features definition file to Solr

```
curl -XPUT
'http://localhost:8983/solr/myCollection
/schema/feature-store' --data-binary
"@/path/myFeatures.json" -H 'Content-
type:application/json'
```

```
[
  {
    "name":  "matchTitle",
    "type": "org.apache.solr.ltr.feature. SolrFeature",
    "params": {
      "q": "{!field f=title}${text}"
  }, {
    "name":  "freshness",
    "type": "org.apache.solr.ltr.feature. SolrFeature",
    "params": {
      "q": "{!func}recip(ms(NOW,timestamp),3.16e-11,1,1)"
  },
  { "name":  "isFromBloomberg", ... },
  { "name":  "popularity", ... }
]
```


annotated by gritmind

II. Extract Features

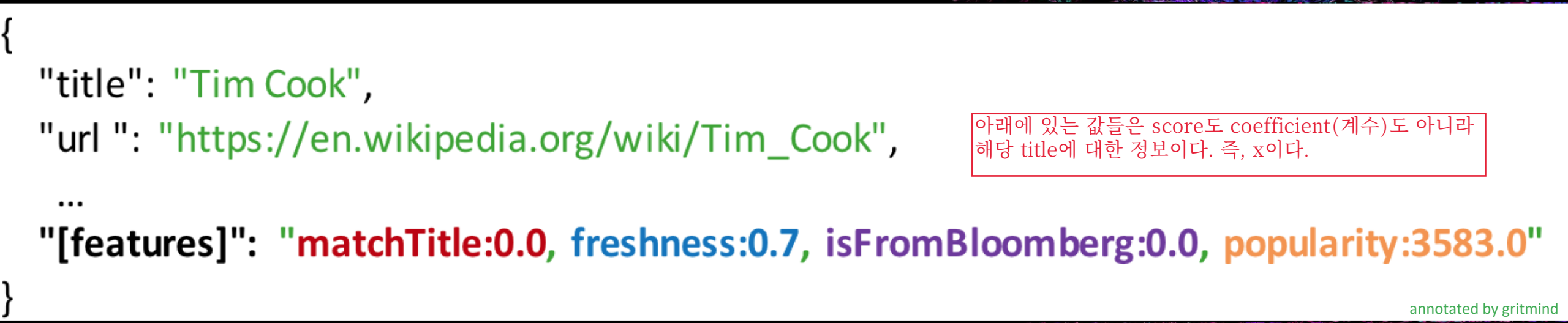
ACTIVATE

- Add features transformer to Solr config

```
<!– Document transformer adding feature vectors with each retrieved document ->
 <transformer name="features"
class="org.apache.solr…LTRFeatureLoggerTransformerFactory" />
```

- Request features for document by adding [features] to the fl parameter

```
http://localhost:8983/solr/myCollection/query?q=test&fl=title,url,[features]
```



```
{
  "title": "Tim Cook",
  "url ": "https://en.wikipedia.org/wiki/Tim_Cook",
    …
  "[features]": "matchTitle:0.0, freshness:0.7, isFromBloomberg:0.0, popularity:3583.0"
}
```

아래에 있는 값들은 score도 coefficient(계수)도 아니라 해당 title에 대한 정보이다. 즉, x이다.

annotated by gritmind

III. Train Model

ACTIVATE

- Combine query-document judgments & features into training data file

- Train ranking model offline
  - RankSVM[1] [liblinear]
  - LambdaMART[2] [ranklib]

Example: Linear model

```
score = 1.2 x matchTitle + 10 x popularity
        + 5.4 x isFromBloomberg -  2 x freshness
```

freshness는 왜 마이너스인가?
최신일수록 값이 작은 것일까?

[1]T. Joachims, *Optimizing Search Engines Using Clickthrough Data*, Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), ACM, 2002.
[2]C.J.C. Burges, "*From RankNet to LambdaRank to LambdaMART: An Overview*", Microsoft Research Technical Report MSR-TR-2010-82, 2010.

annotated by gritmind

IV. Deploy Model

ACTIVATE

- Generate trained output model in `myModelName.json`
- Deploy model definition file to Solr

```
curl -XPUT
'http://localhost:8983/solr/techproducts/schema/mode
l-store' --data-binary "@/path/myModelName.json" -H
'Content-type:application/json'
```

학습이 완료된 모델이
class 위치에 저장되어 있는 것 같다.


```
{
  "class": "org.apache.solr.ltr.model.MultipleAdditiveTreesModel",
  "name": "myModelName",
  "features": [ { "name": "freshness"}, { "name": "matchTitle"}, ... ],
  "params": {
    "trees": [ {
        "weight": 1,
        "tree": {
          "feature": "matchedTitle",
          "threshold": 0.5,
          "left": { "value": -100 },
          "right": {
            "feature": "freshness",
            "threshold": 0.5,
            "left": { "value": 50 },
            "right": { "value": 75 }
          }
        }
    } ]
  }
}
```


annotated by gritmind

V. Re-rank Results

ACTIVATE

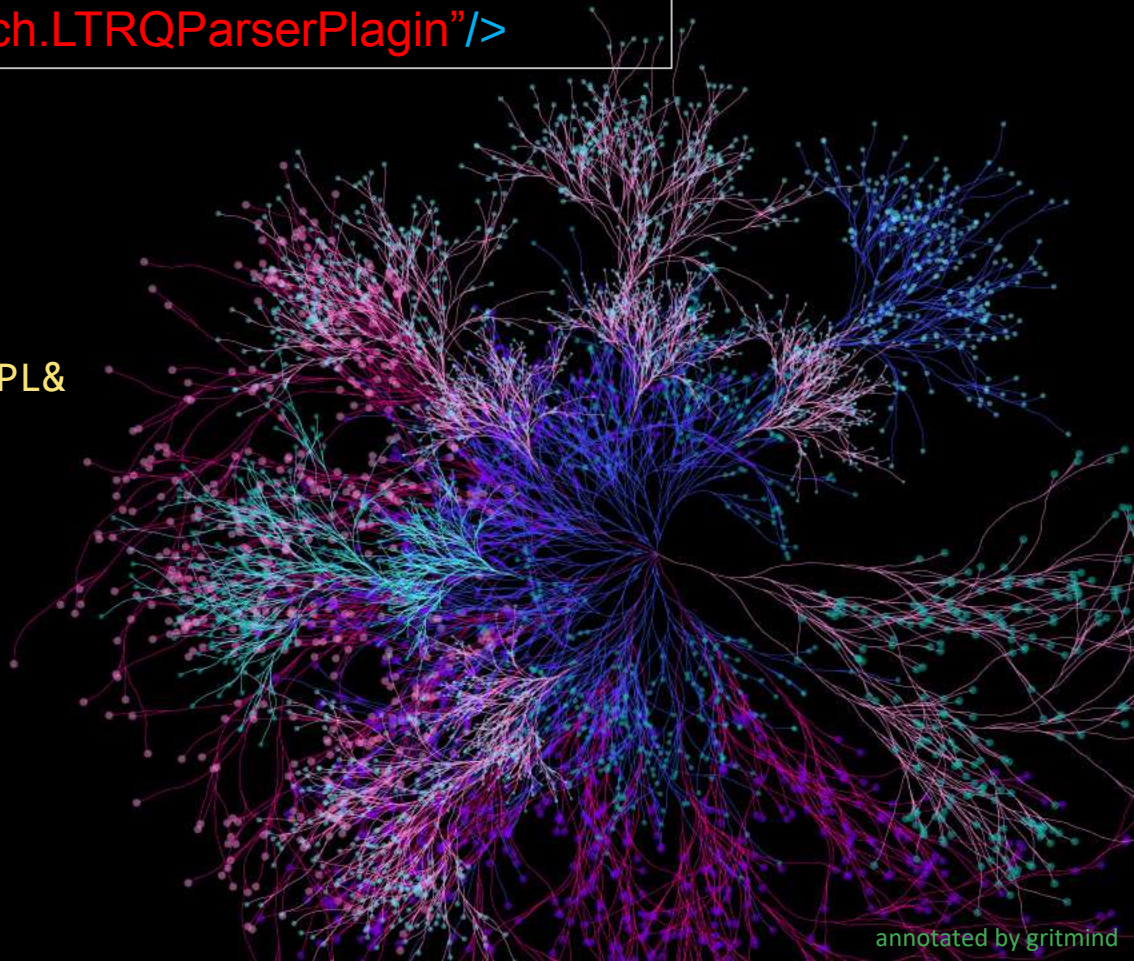
- Add LTR query parser to Solr config

```
<!– Query parser used to re-rank top docs with a provided model -->
<queryParser name="ltr" class="org.apache.solr.ltr.search.LTRQParserPlagin"/>
```

- Search and re-rank results

```
http://localhost:8983/solr/myCollection/query?q=AAPL&
rq={!ltr model="myModelName" reRankDocs=100}
```

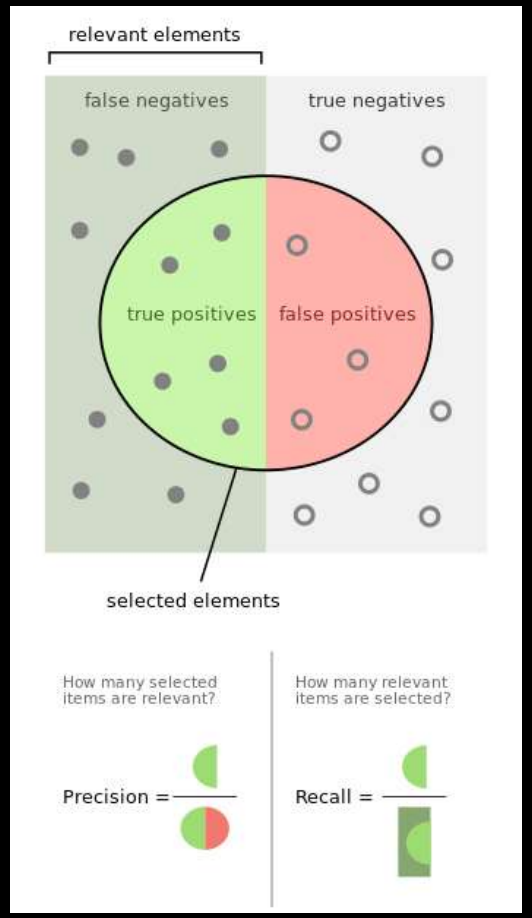
이런 식으로..
기존에 있던 시스템에서
reranking model (="myModelName")을 덧붙여서 추가하는 것 같다.
파이프라인에서 한 개의 모듈만 추가된 것과 같다.



annotated by gritmind

VI. Evaluate quality of search

ACTIVATE



Precision

how many relevant results I returned divided by total number of **results returned**

Recall

how many relevant results I returned divided by total number of **relevant results for the query**

NDCG (discounted cumulative gain)

Ranking에서 NDCG metric을 많이 사용한다.

precision이 똑같이 0.5 이더라도 리스트에서 맞춘 케이스들이 더 위에 있을수록 NDCG는 더 높은 값을 가진다.

https://commons.wikimedia.org/wiki/File:Precisionrecall.svg

annotated by gritmind

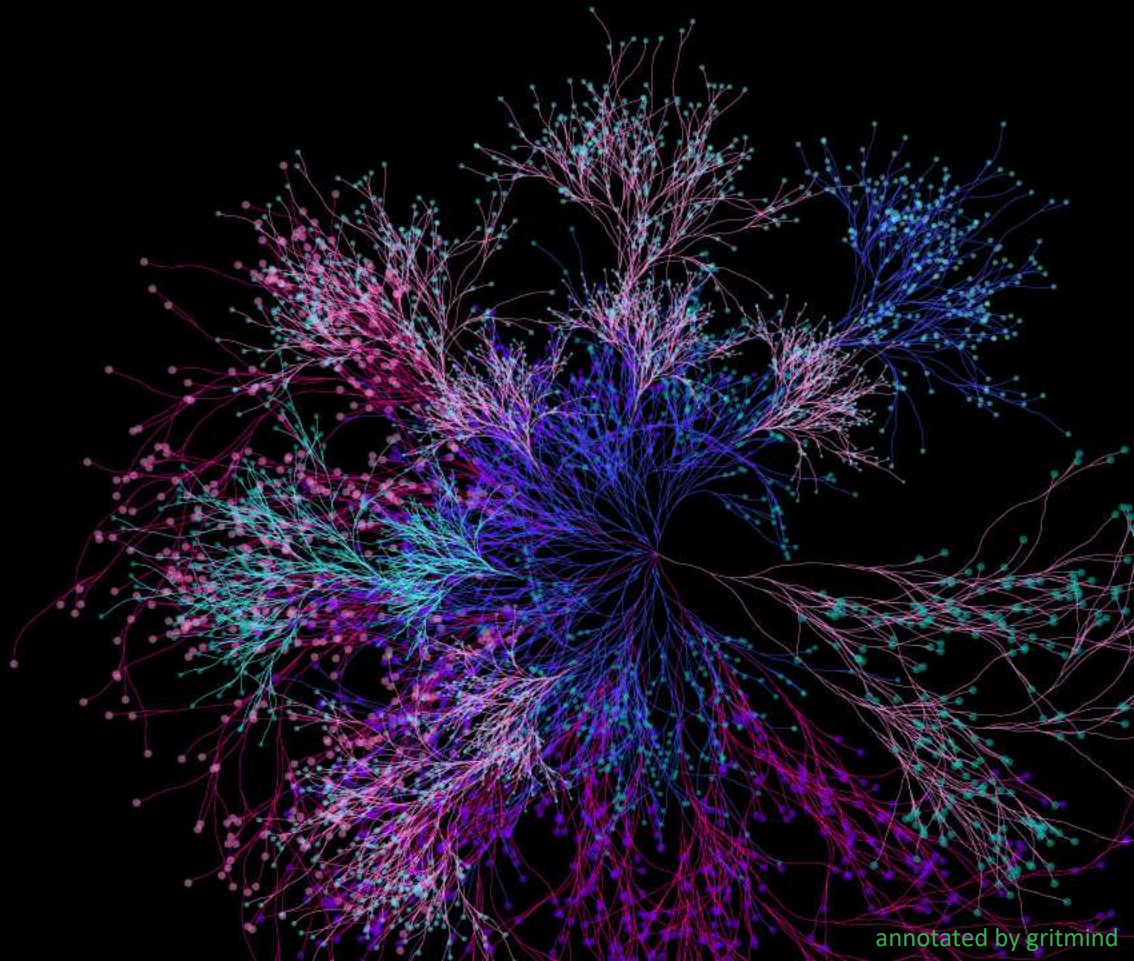
How to Ship LTR in Production in 3 Steps

ACTIVATE

Make it Work

Make it Fast

Deploy to Production

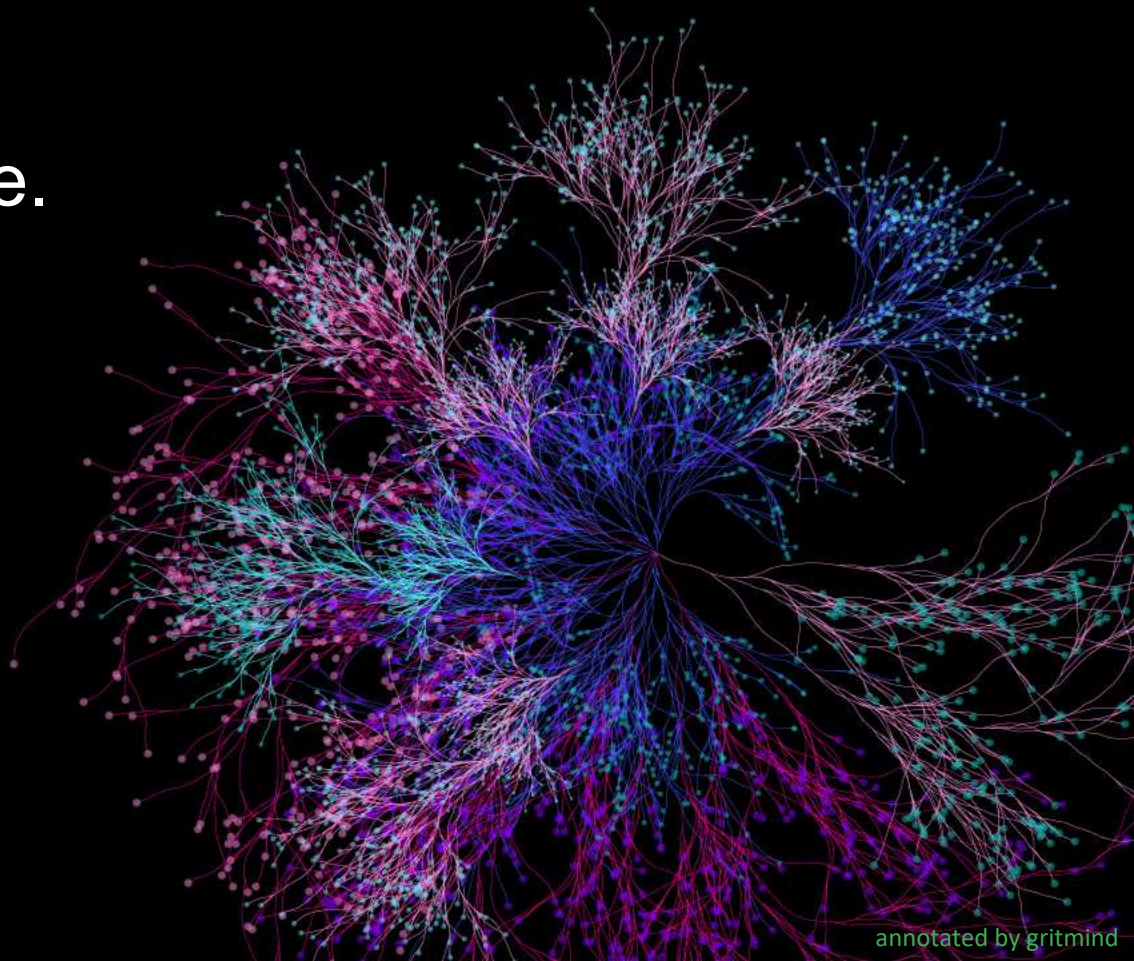


annotated by gritmind

Are We Good to Go?

ACTIVATE

- Mid 2017: the infrastructure bit seemed to be done.
- We had a ‘placeholder’ feature store. But, we needed useful features *computed in production.*
- We deployed a new shiny feature store…

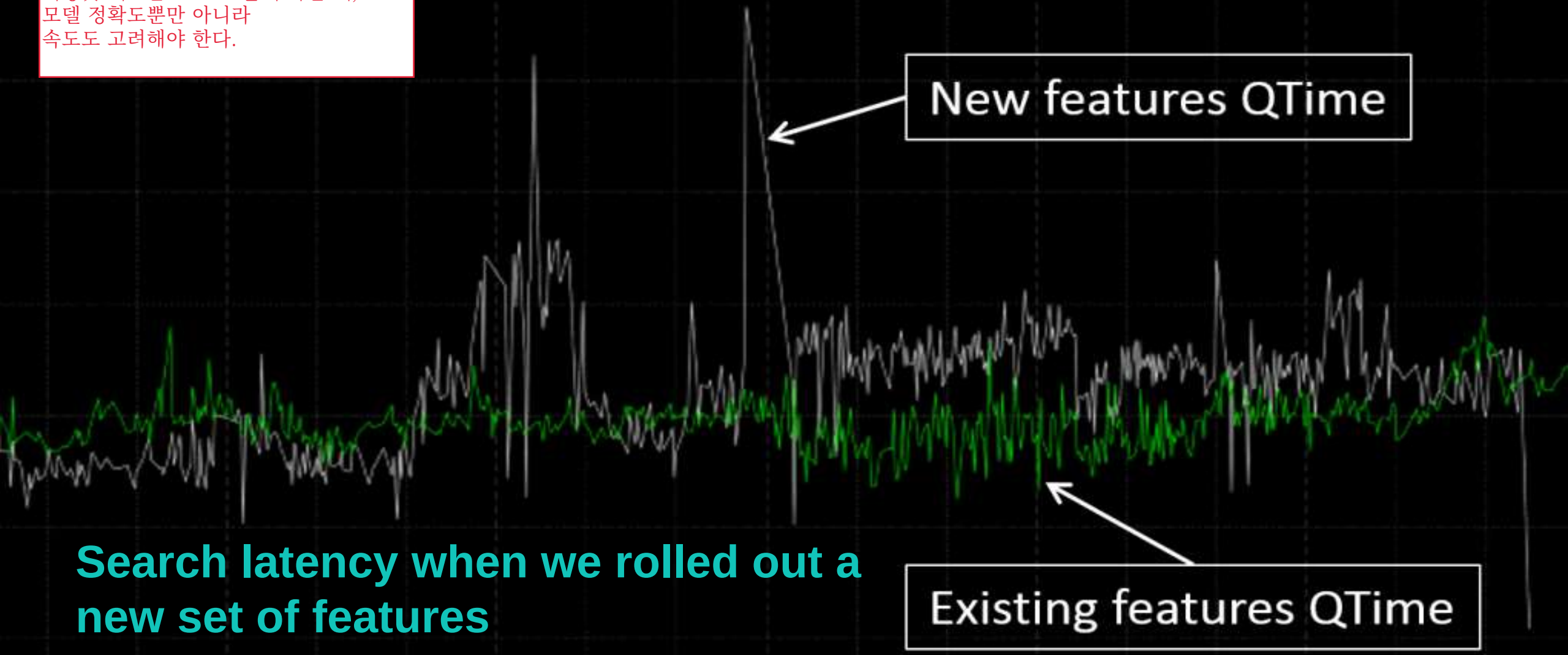


annotated by gritmind

There’s no model without features…

ACTIVATE

이렇듯 새로운 feature를 추가할 때,
모델 정확도뿐만 아니라
속도도 고려해야 한다.



Search latency when we rolled out a new set of features

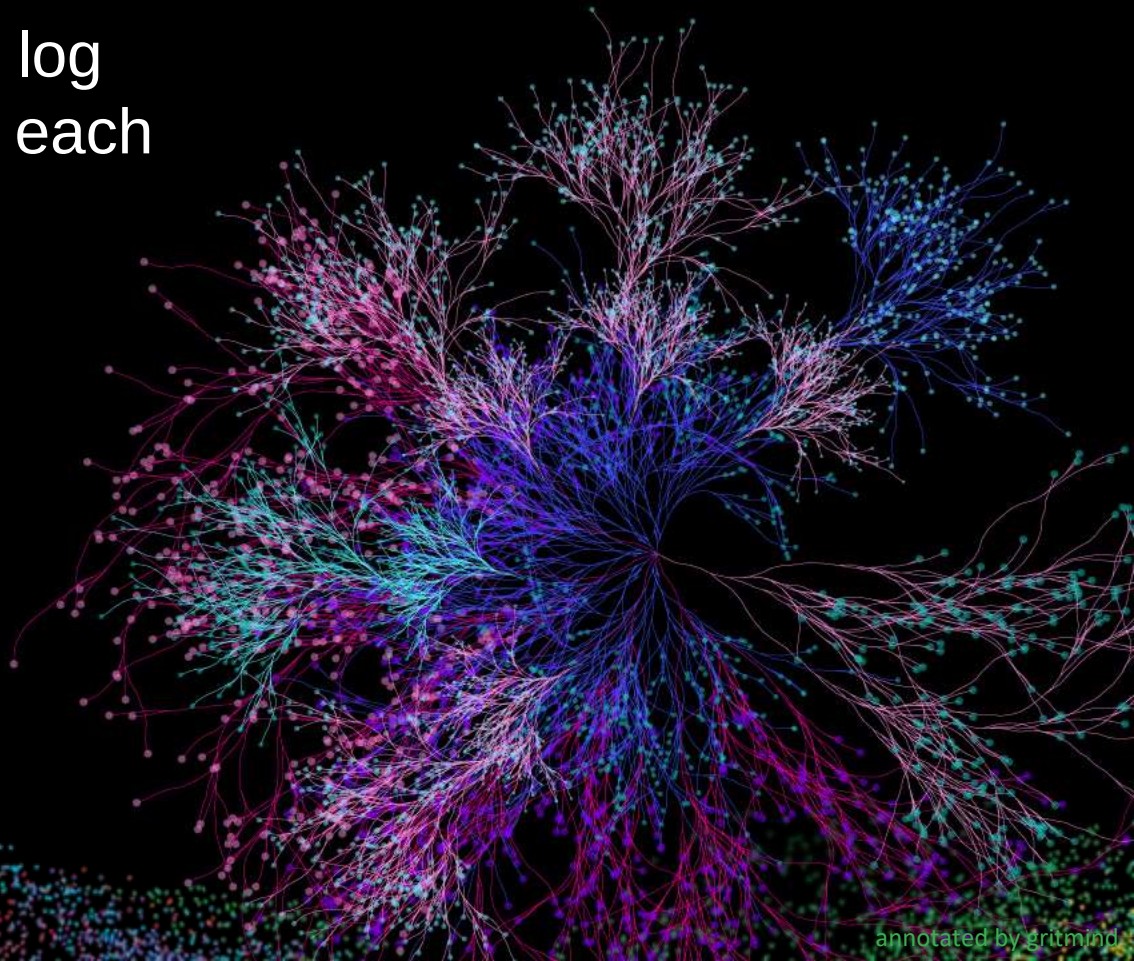
annotated by gritmind


Why was it slow?
Which features were to blame?
annotated by gritmind

Metrics on feature latency

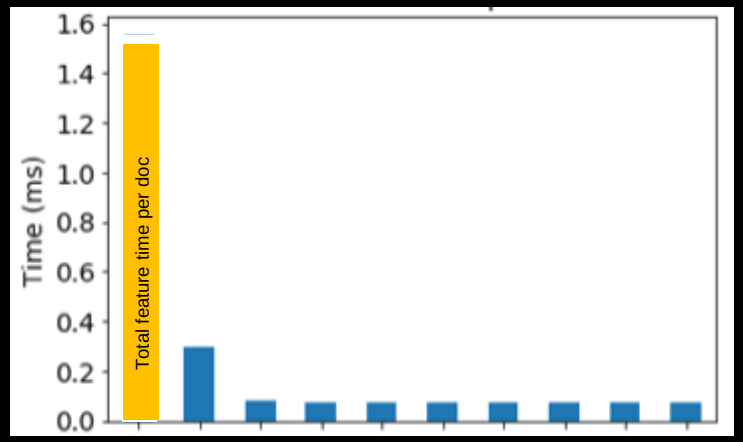
ACTIVATE

- We instrumented support in Apache Solr to log the time it took to compute each feature on each document.
- We added analytics on top of that.



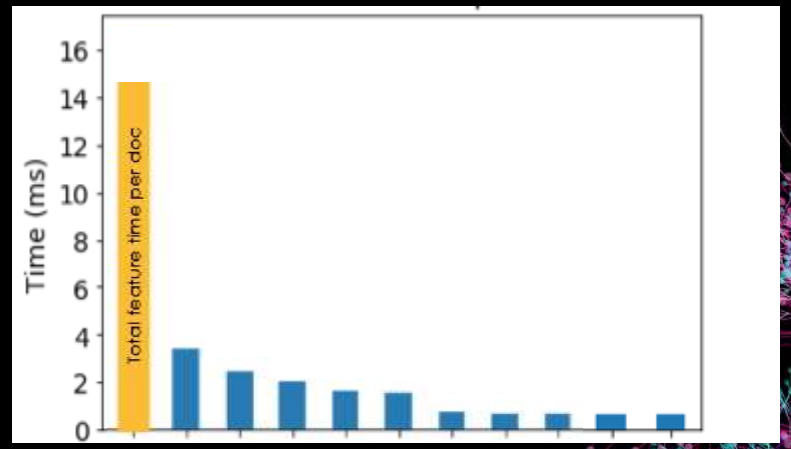
Why is it slow?

Feature Latencies using the **old set of features**



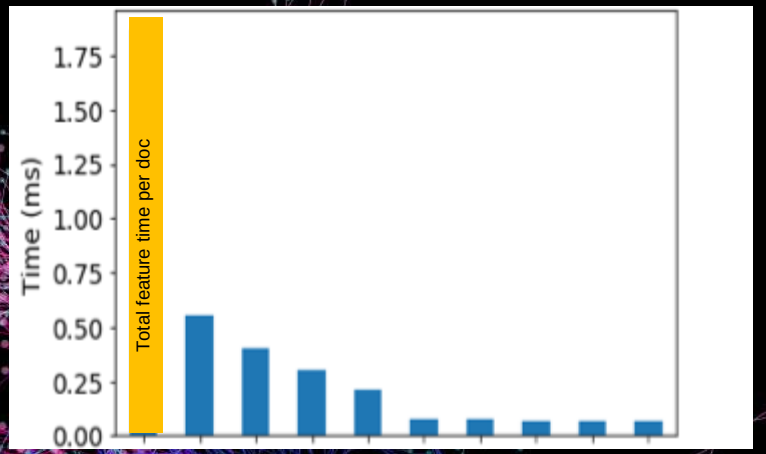
Total time per search: 15ms

Feature Latencies using the new set of features **SlothFeature**



Total time per search: 145ms

Feature Latencies using the new set of features, including the **FastSlothFeature**



Total time per search: 19ms

- 1.9ms is ok for 10 documents, but not for 100 docs…

annotated by gritmind

How do we go faster?

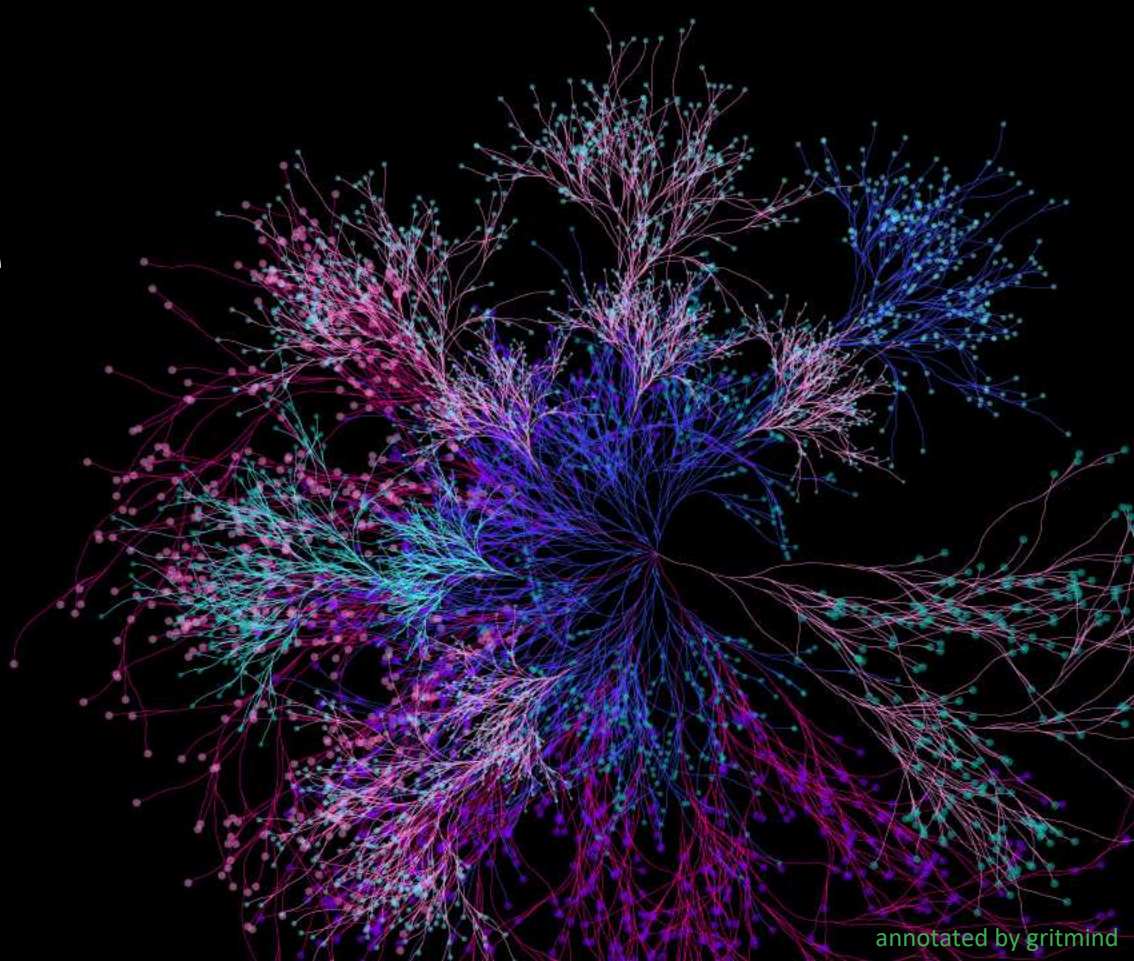
ACTIVATE

- Some of the features are unrelated to the query: For example: is the source reliable?

- We can precompute them and store them in the index.

속도 향상을 위해
미리 계산할 수 있으면, 미리 계산하는 것이 좋다.

핵심: 미리 계산할 수 있으면 미리 해놓자.



Index Static Features

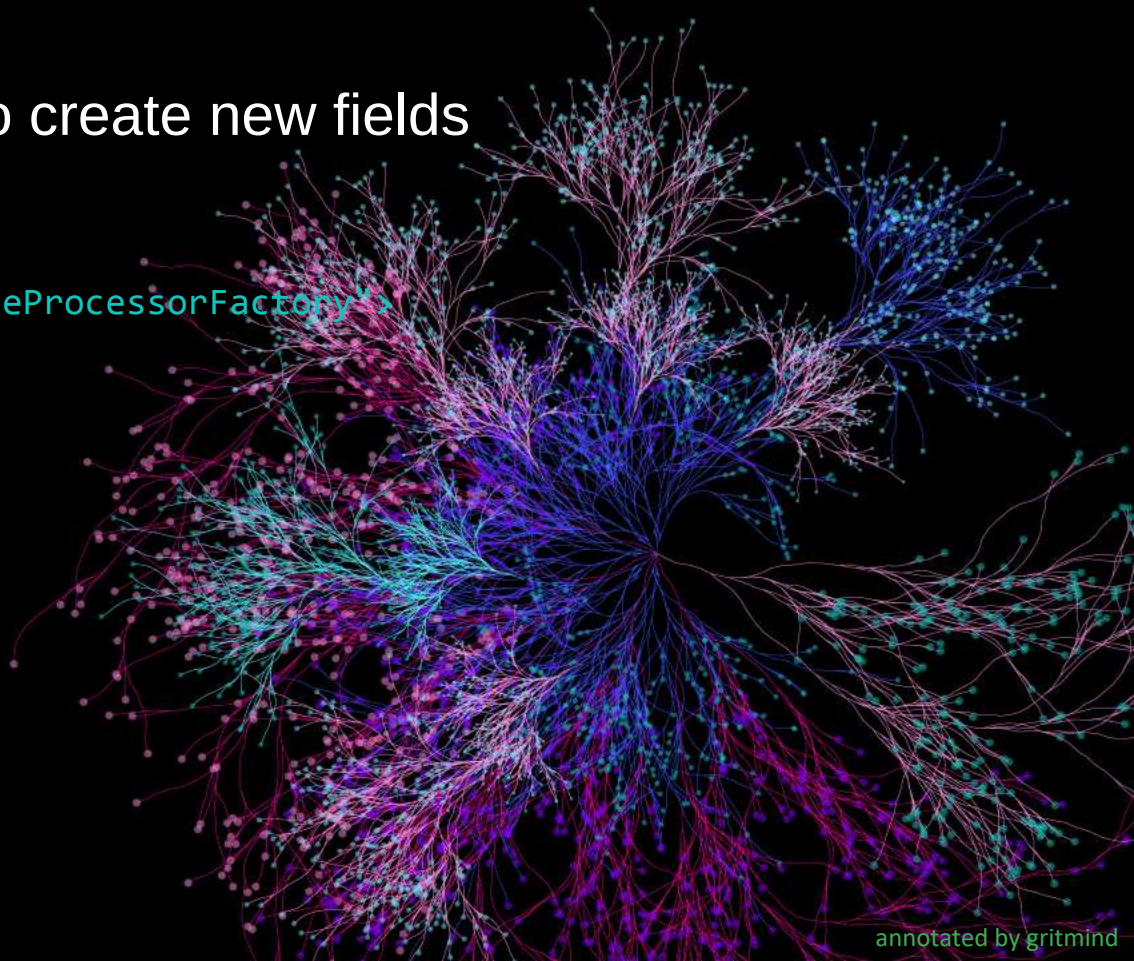
ACTIVATE

- Add *feature_is_wire_BLAH* to the Solr schema

```
<field name="feature_is_wire_BLAH" type="tdouble" indexed="false" stored="true"
   docValues=”false" required="false”/>
```

- Use UpdateRequestProcessors in Solr config to create new fields

```
<updateRequestProcessorChain
    <processor
        class="com.internal.solr.update.processor.IsFoundUpdateProcessorFactory">
        <str name="source">wire</str>
        <str name="dest">feature_is_wire_BLAH</str>
        <str name="map">
            {
            ”BLAH" : 1.0
            }
        </str>
        <double name="default">0.0</double>
    </processor>
</updateRequestProcessorChain>
```



annotated by gritmind

Index Static Features

ACTIVATE

- Features at runtime are produced by reading the value from the index using the FieldValueFeature.

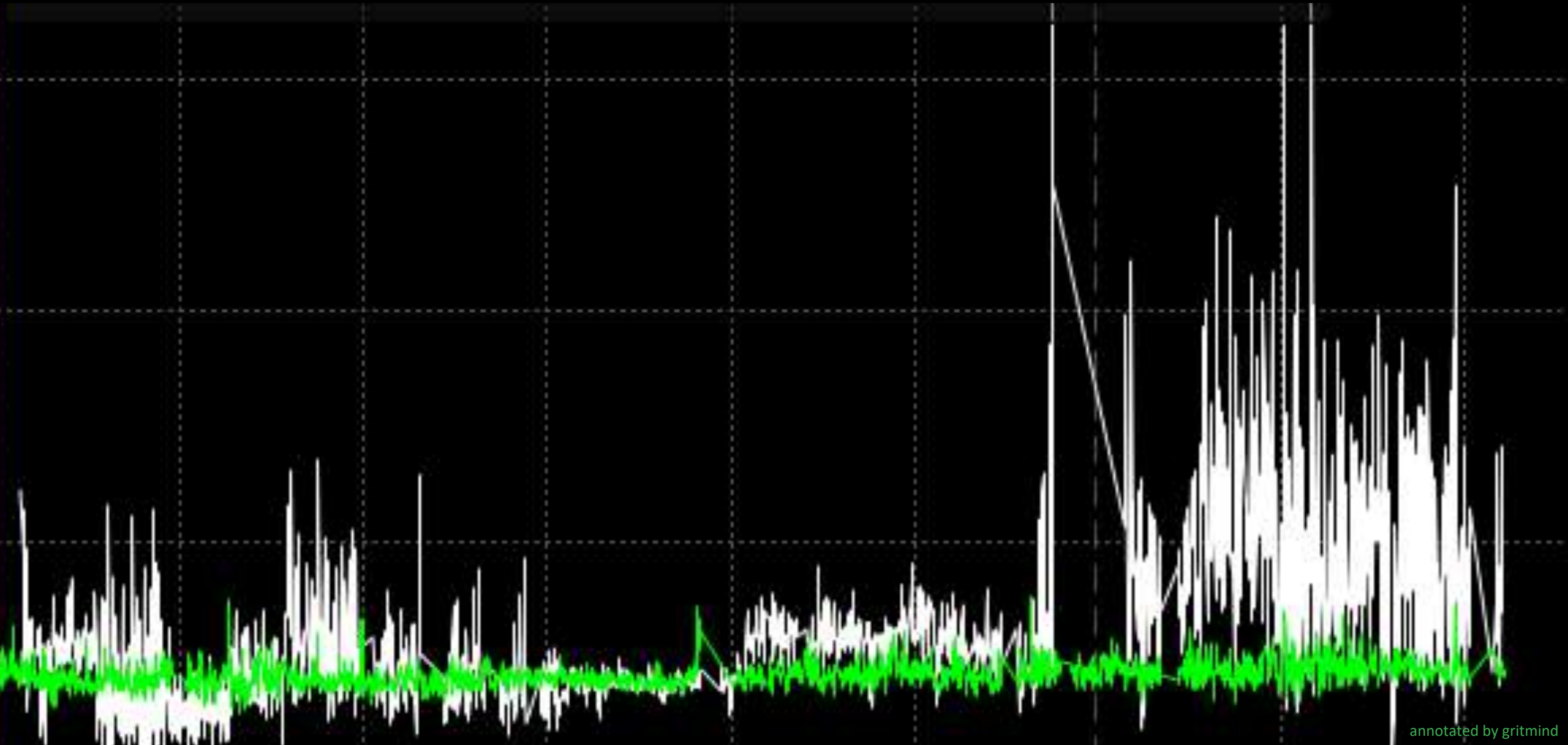
```
features":[
   {
       "type":"ltr.feature.impl.FieldValueFeature",
       "params": {
           "field": "feature_is_wire_BLAH"
       },
       "name": “is_wire_BLAH",
       "default": 0.0
   },
```

Would this ease all our performance troubles?

annotated by gritmind

Better?

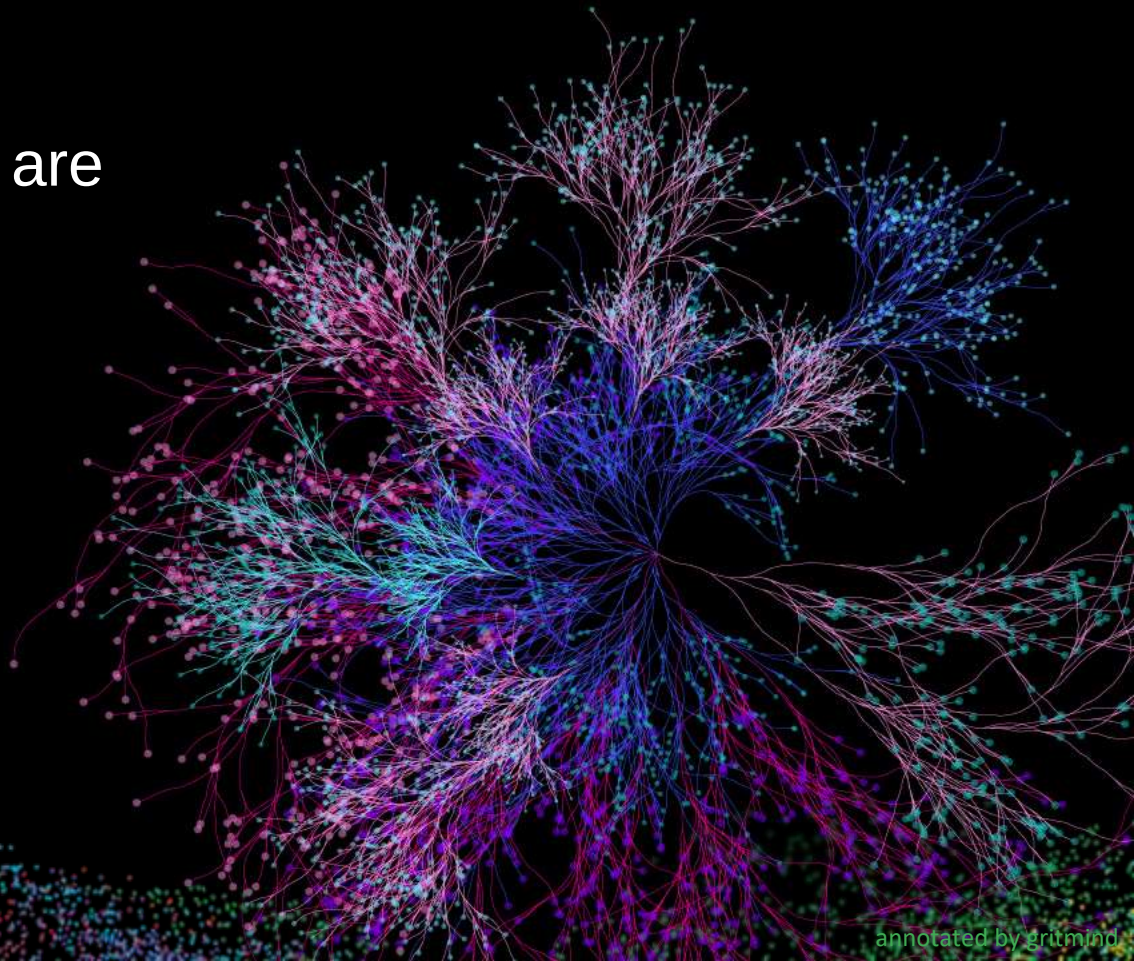
ACTIVATE



Reading from the index can still be slow…

ACTIVATE

- Retrieving field values from their stored values is slow
- So we changed the implementation of *FieldValueFeature* to use DocValues if they are available
- DocValues record field values in a column-oriented way, mapping doc ids to values

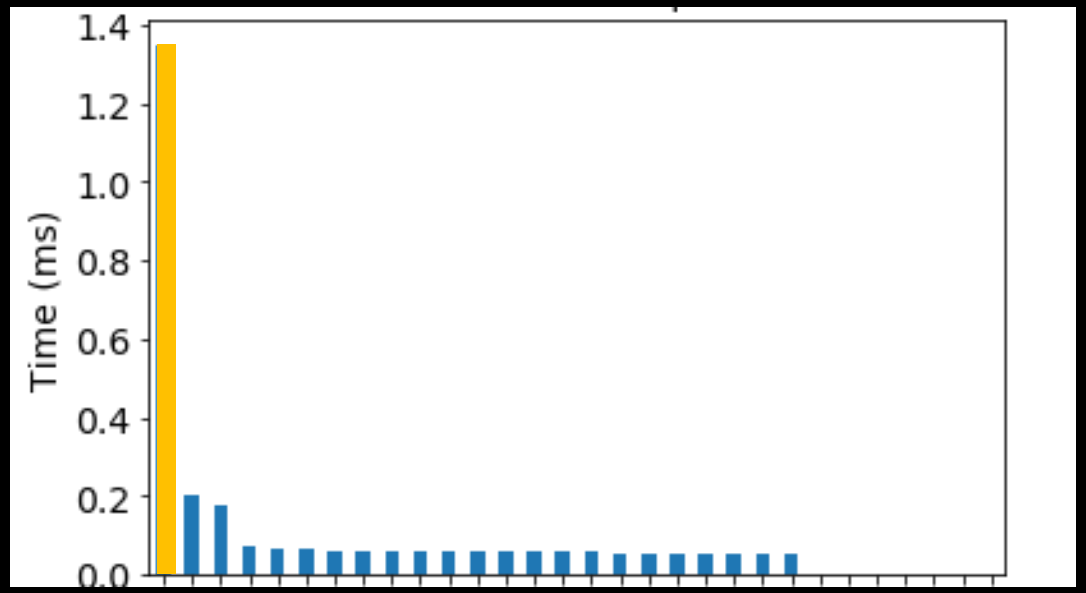


annotated by gritmind

From Stored Fields to Doc Values : 5x faster!

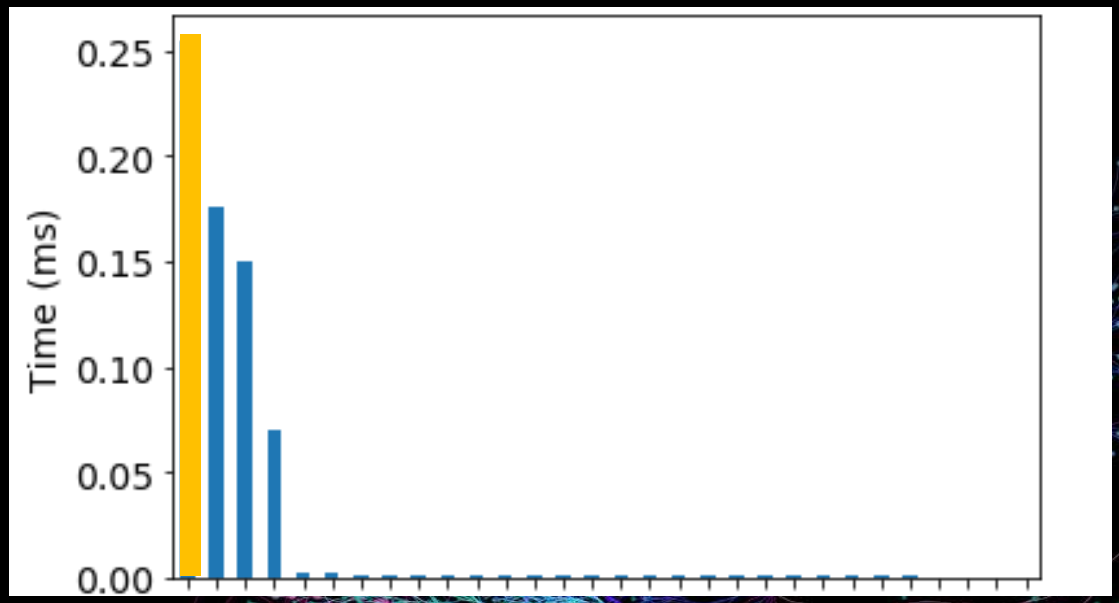
ACTIVATE

Feature latencies when retrieving 100 docs using **StoredFields**



Total time per search: **135ms**

Feature latencies when retrieving 100 docs using **DocValues**



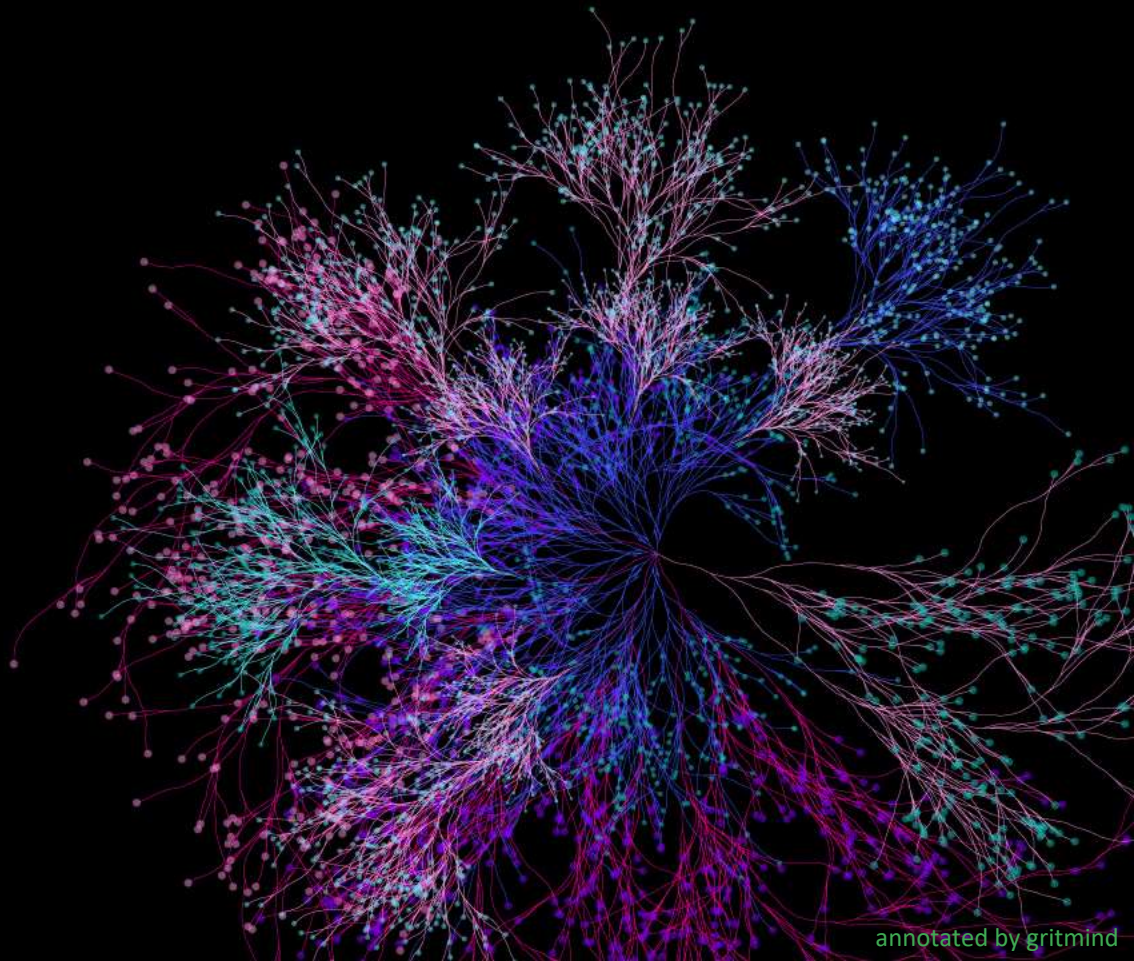
Total time per search: **25ms**

annotated by gritmind

Are we there yet?

ACTIVATE

- ✓ Feature logging/computation
- ✗ Document re-ranking



annotated by gritmind

Evaluating performance: NO-OP Model

ACTIVATE

- A linear model, performing all the computations but not modifying the original order

Original Solr Score	Query matches the title	Freshness	Is the document from Bloomberg.com?	Popularity
2.3	0	0.7	0	3583
2.1	1	0.9	1	625
1.3	0	0.1	0	129

x

No-op
1.0
0.0
0.0
0.0
0.0

=

Final Score
2.3
2.1
1.3

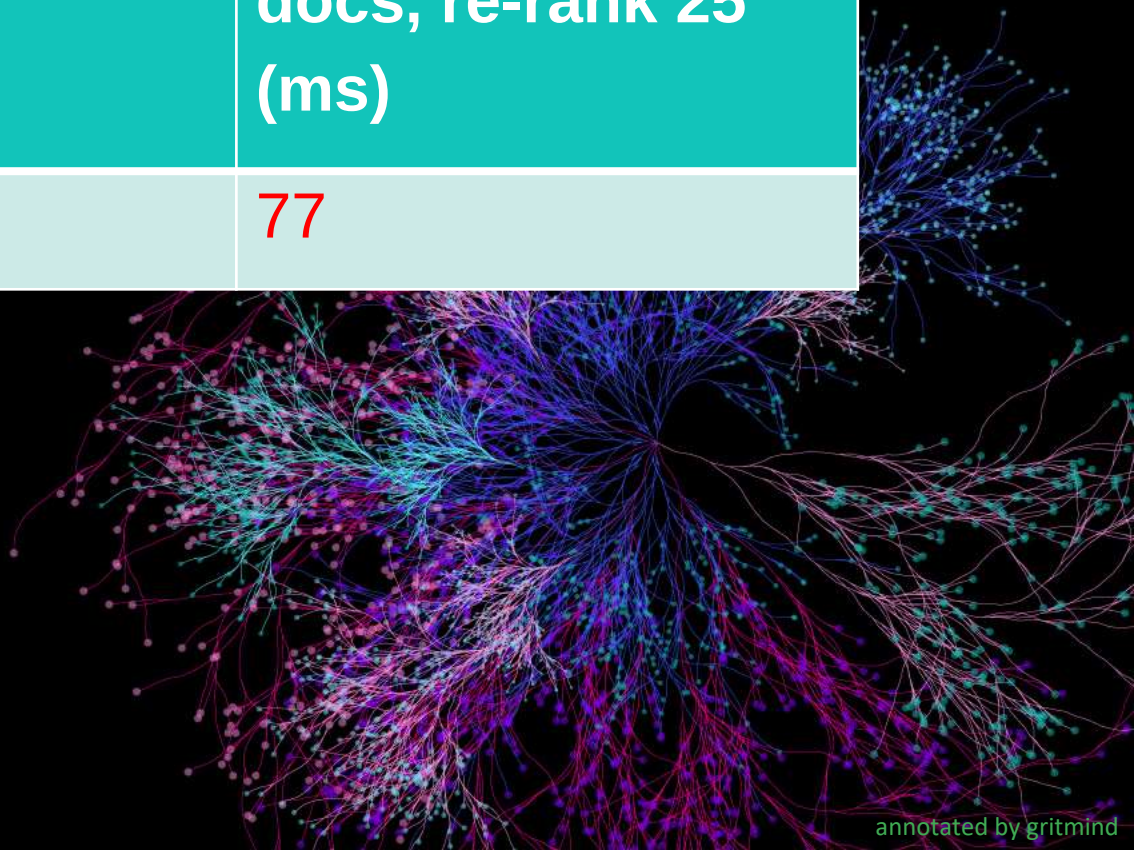
annotated by gritmind

Need for speed

ACTIVATE

	No LTR, retrieve 3 docs (ms)	LTR retrieve 3 docs, re-rank 25 (ms)
Median search time	39	77

Why is it so slow???



annotated by gritmind

News needs grouping

ACTIVATE

- Similar news stories are grouped (clustered) together and only the top-ranked story in each group is shown

annotated by gritmind

Grouping + Re-ranking: Not a match made in heaven

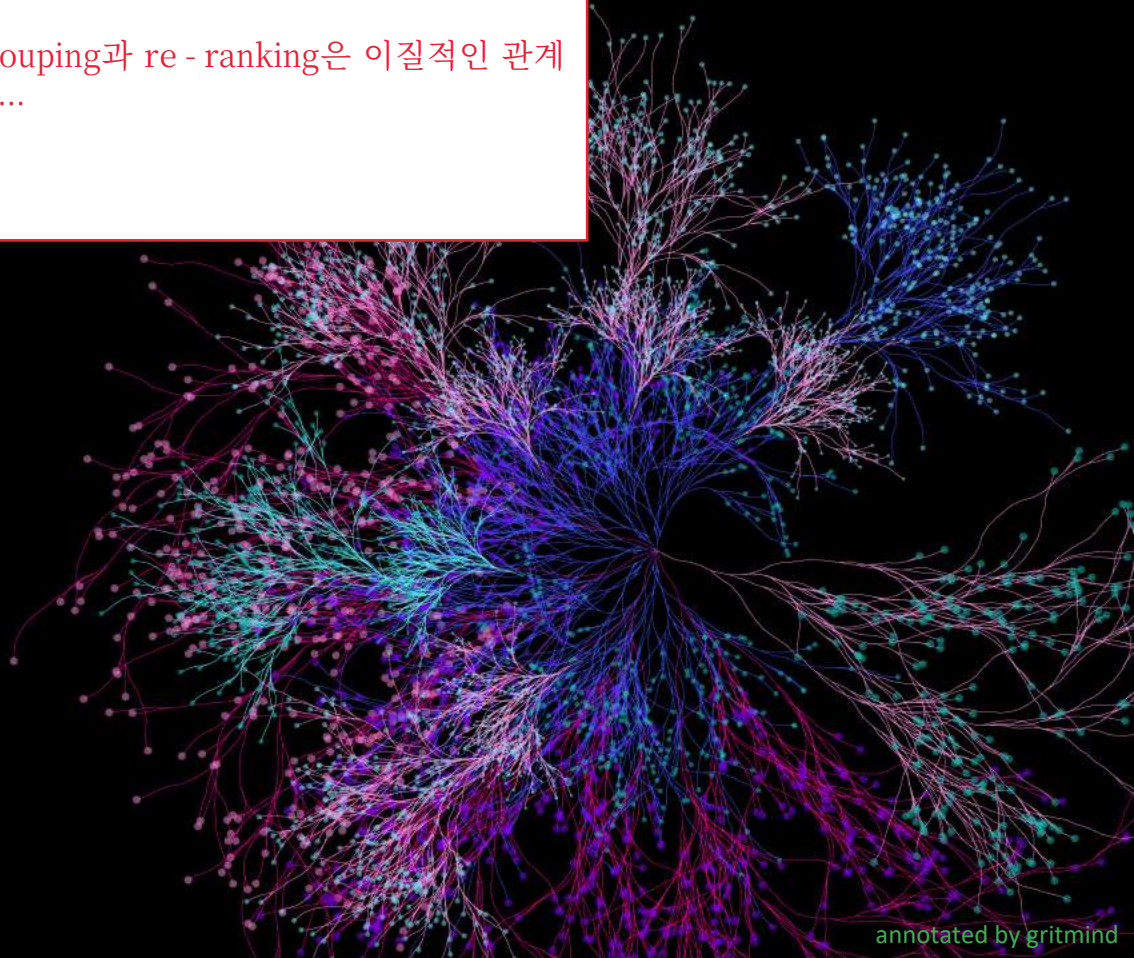
ACTIVATE

- Regular grouping involves 3 communication rounds between coordinator and shards

- With re-ranking, we have to re-rank the groups and the documents in each group

grouping을 하면 좋은데

grouping과 re - ranking은 이질적인 관계다...



[SOLR-8776 Support RankQuery in grouping]

annotated by gritmind

ACTIVATE

annotated by gritmind

What is the Las Vegas Patch?

ACTIVATE

Solr / SOLR-11831

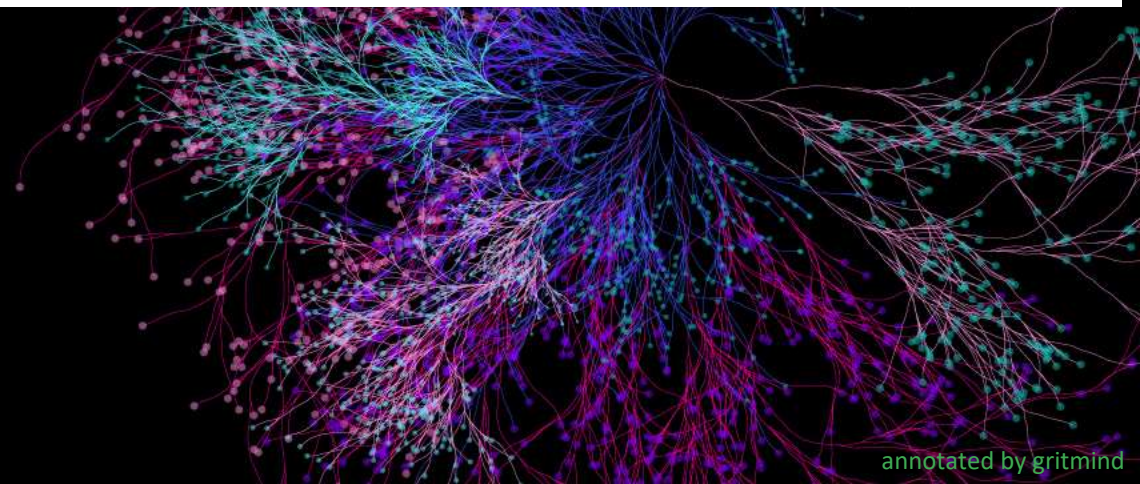
Skip second grouping step if group.limit is 1 (aka Las Vegas patch)

Details

Type:	Improvement	Status:	OPEN
Priority:	Minor	Resolution:	Unresolved
Affects Version/s:	None	Fix Version/s:	None
Component/s:	None	Security Level:	Public (Default Security Level. Issues are Public)
Labels:	None		

People

Assignee:	Unassigned
Reporter:	Malvina Josephidou
Votes:	1 Vote for this issue
Watchers:	7 Start watching this issue



annotated by gritmind

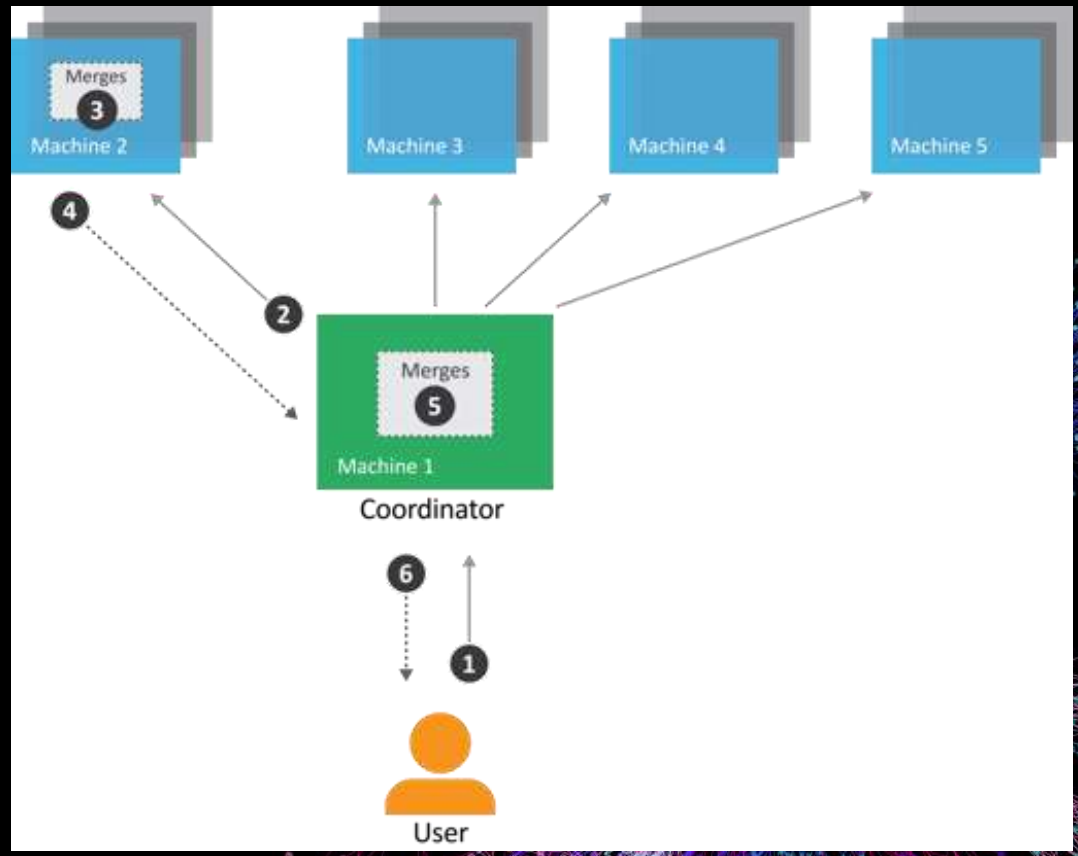
Grouping

대용량 데이터가 있는 시스템에서
grouping은 원래 어렵지만
ranking과 함께 할 때는 더더욱 어렵다.

ACTIVATE

Three requests from the coordinator:

1. Coordinator asks for top n groups for the query and computes top n groups.
2. Each shard compute top m documents for the top n groups
3. Coordinator retrieves top docs for each group and retrieve them from the shards



annotated by gritmind

Why? Example:

ACTIVATE

- Top 2 groups, top 2 documents, 2 shards

Machine 1

Doc	Group	Score
doc1	group1	20.0
doc2	group2	5.0
doc3	group1	100.0
doc4	group1	120.0
doc5	group2	10.0

Machine 2

Doc	Group	Score
doc6	group3	70.0
doc7	group2	65.0
doc8	group3	60.0
doc9	group2	60.0
doc10	group1	50.0

Group	Docs	Score
group1	doc4	120.0
	doc1	20.0
group2	doc5	10.0
	doc2	5.0

Group	Docs	Score
group3	doc6	70.0
	doc8	60.0
group2	doc7	65.0
	doc9	60.0

Top Groups:

Group1: doc4 doc1
Group3: doc6 doc8

Top docs should be:

Group1: doc4 **doc10**
Group3: doc6 doc8

annotated by gritmind

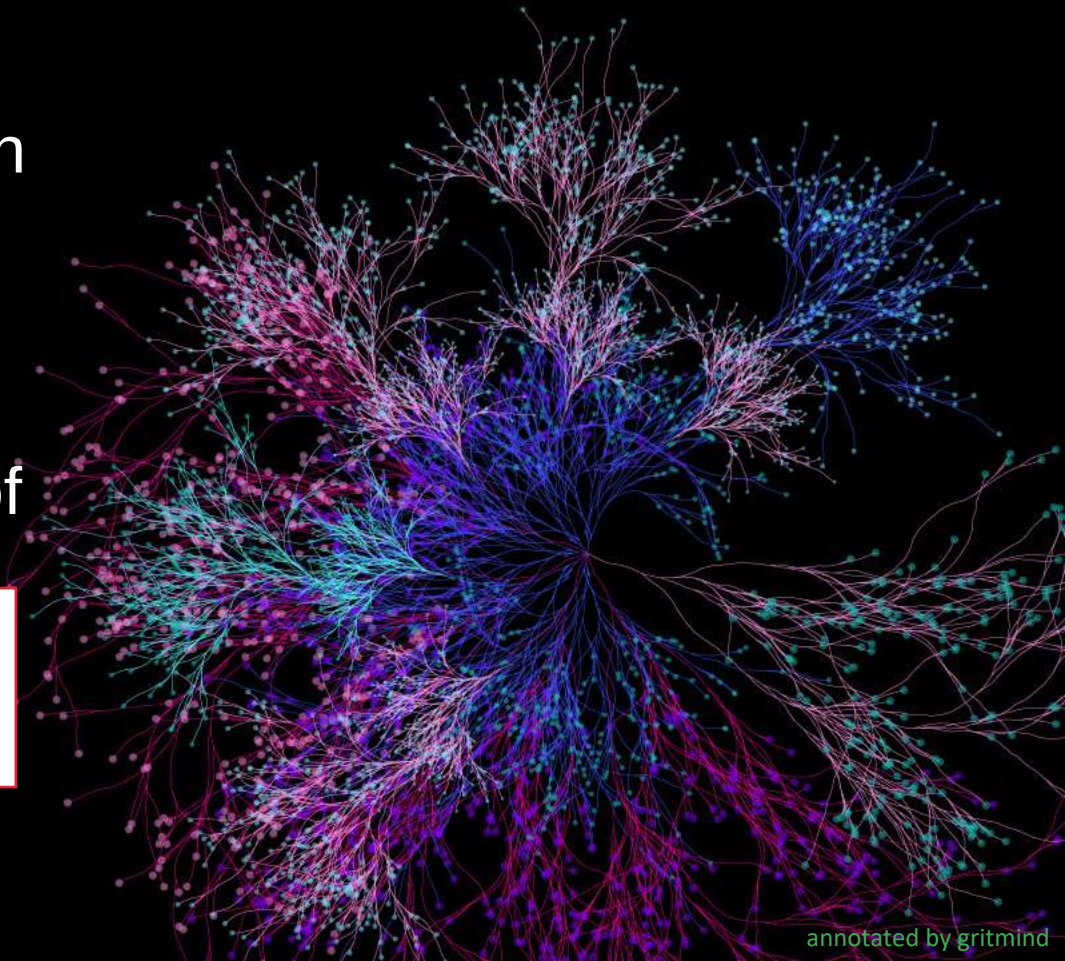
Las Vegas Idea

ACTIVATE

- If you want just one document per group, you do not have this problem
- We can return the top document from each group in the first step and skip the second step entirely
- For LTR: Re-rank only the top document of each group

grouping을 통해서
re - ranking model에 들어가는 입력 개수를 또 줄였다.

비슷한 것들 중에서 가장 좋은 애만 들어가! 라는 느낌.



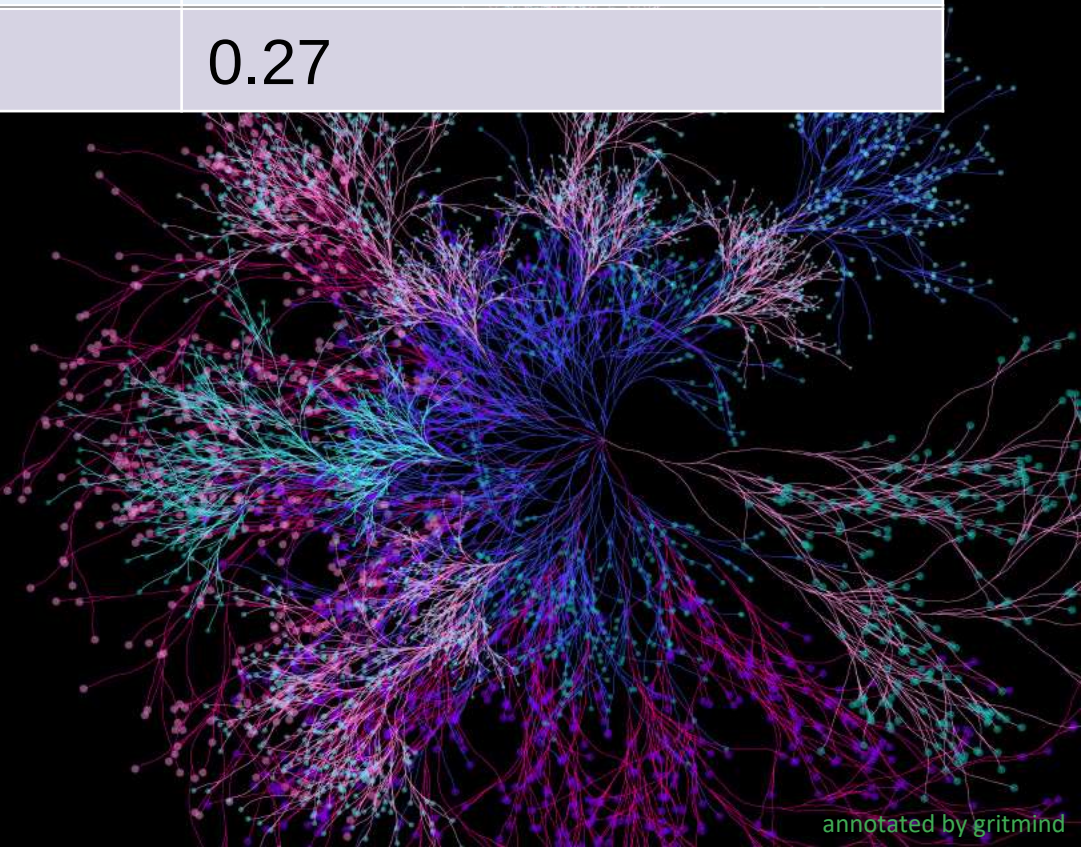
annotated by gritmind

Show me the numbers

ACTIVATE

Method	Median time	Perc95 time
Normal Grouping (No LTR)	0.20	0.35
Las Vegas (No LTR)	**0.12**	0.26
Las Vegas+**LTR no-op**	0.18	0.27

- We made plain old search faster: by about 40%!
- LTR-served searches are still faster than they were before we did the Las Vegas optimization



annotated by gritmind

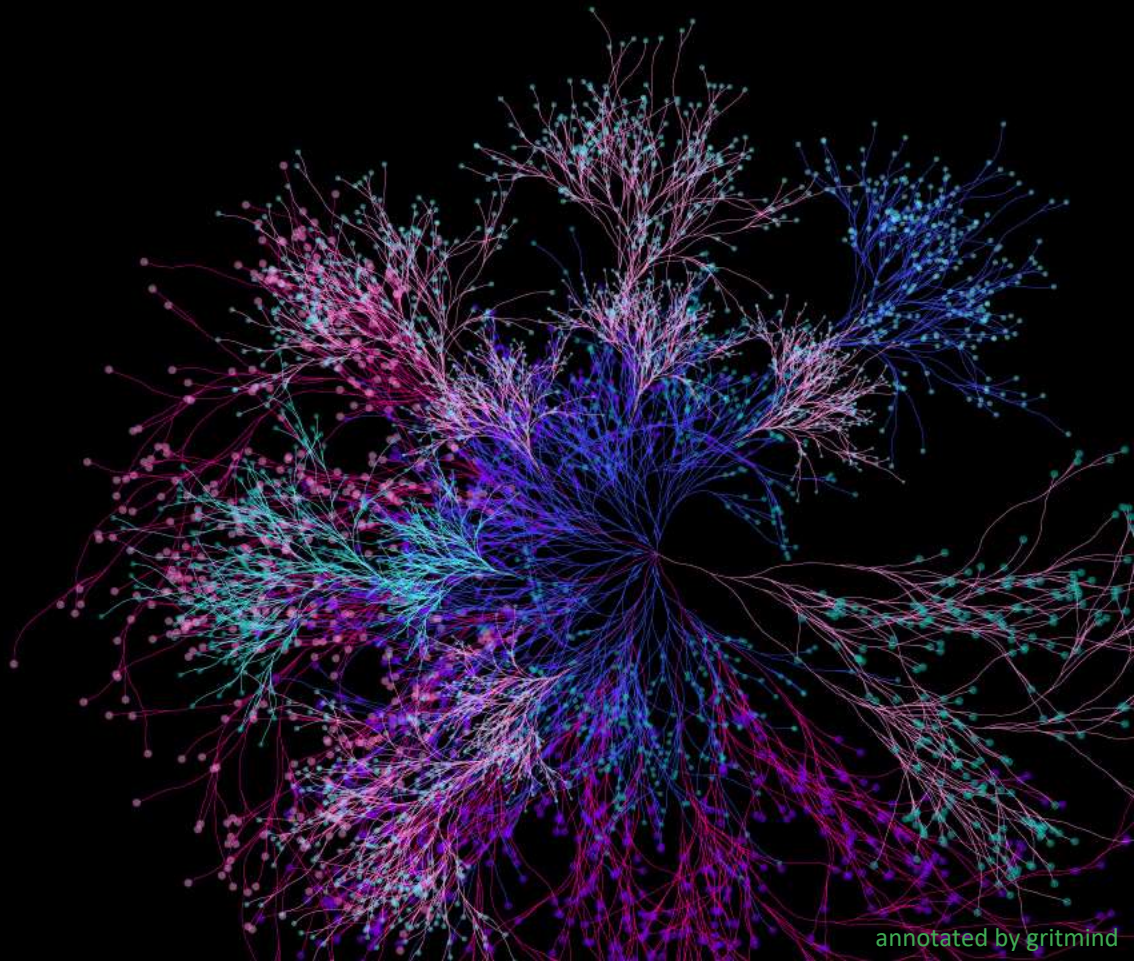
How to Ship LTR in Production in 3 Steps

ACTIVATE

Make it Work

Make it Fast

Deploy to Production



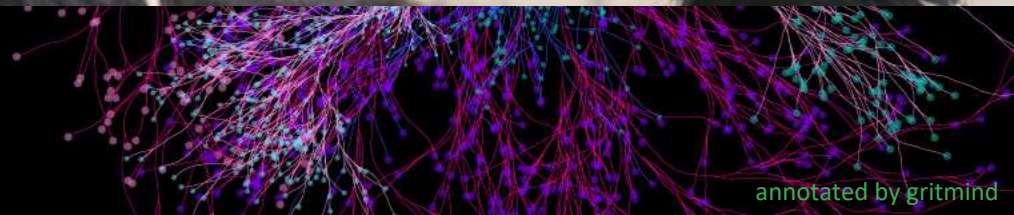
annotated by gritmind

Where is the model?

ACTIVATE

Let the LTR hackathon start…

- Write code to process training data in svmlite format.
- Wrappers around scikit-learn to train various linear models, do regularization, hyperparameter optimization and model debugging
- Evaluate the model: MAP, NDCG, MRR



A Small Model for LTR, a Giant Step for Bloomberg

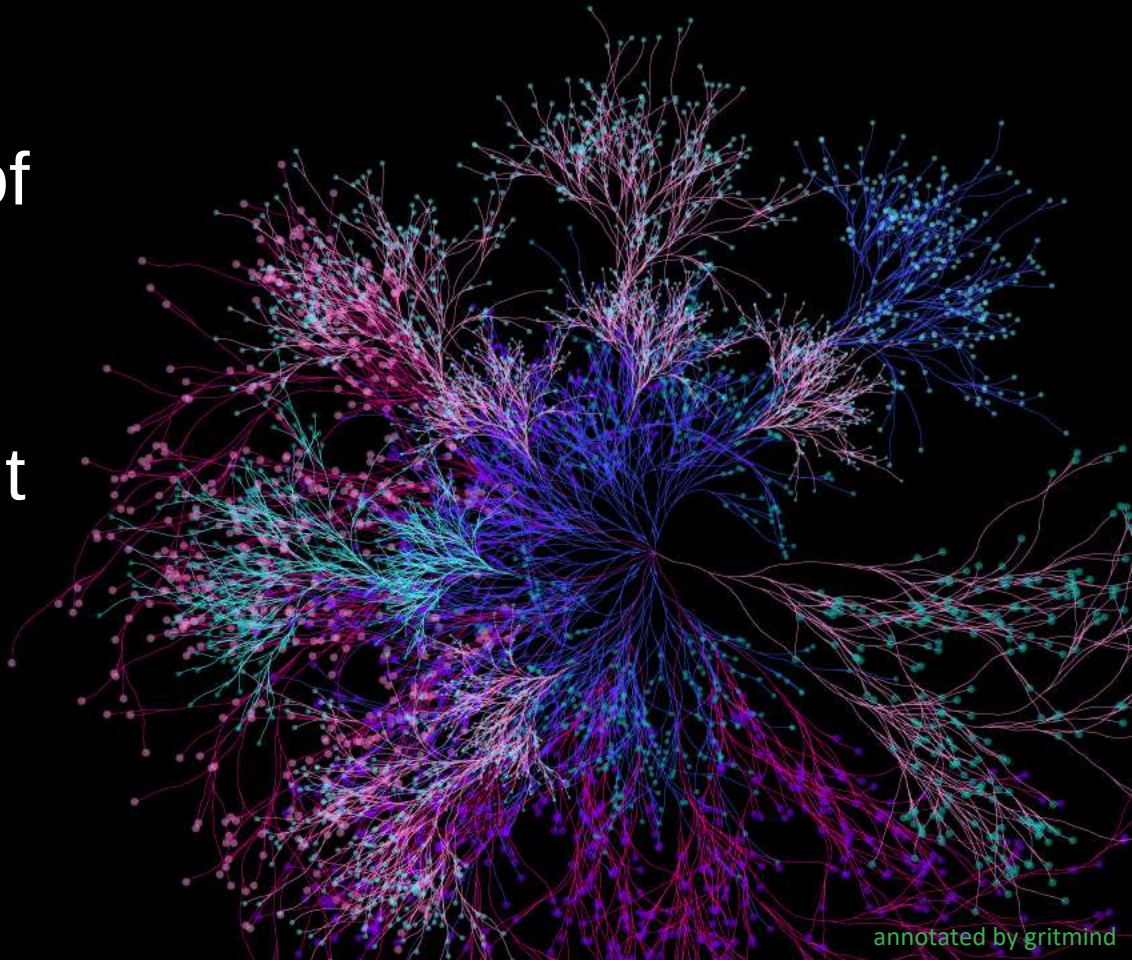
ACTIVATE

- Released initially to select internal users
- Then, to all internal users
- Then, to 10% of our clients
- And finally, to 100% of our clients

It’s a whole new (LTR) world!

ACTIVATE

- Trying out new features, new models and classes of models
- Experimenting with different types of training data
- Rolling out new rankers, for different types of queries



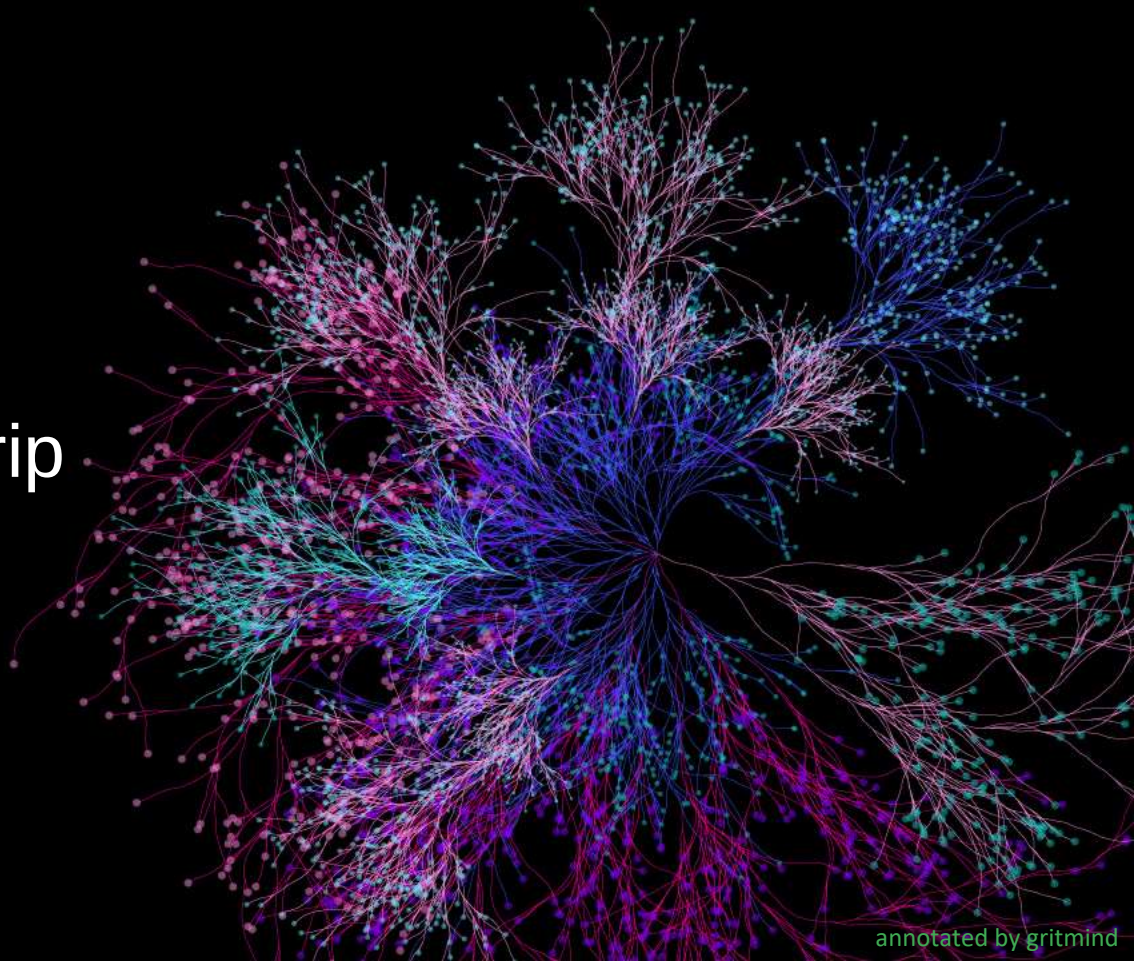
annotated by gritmind

Take home messages

ACTIVATE

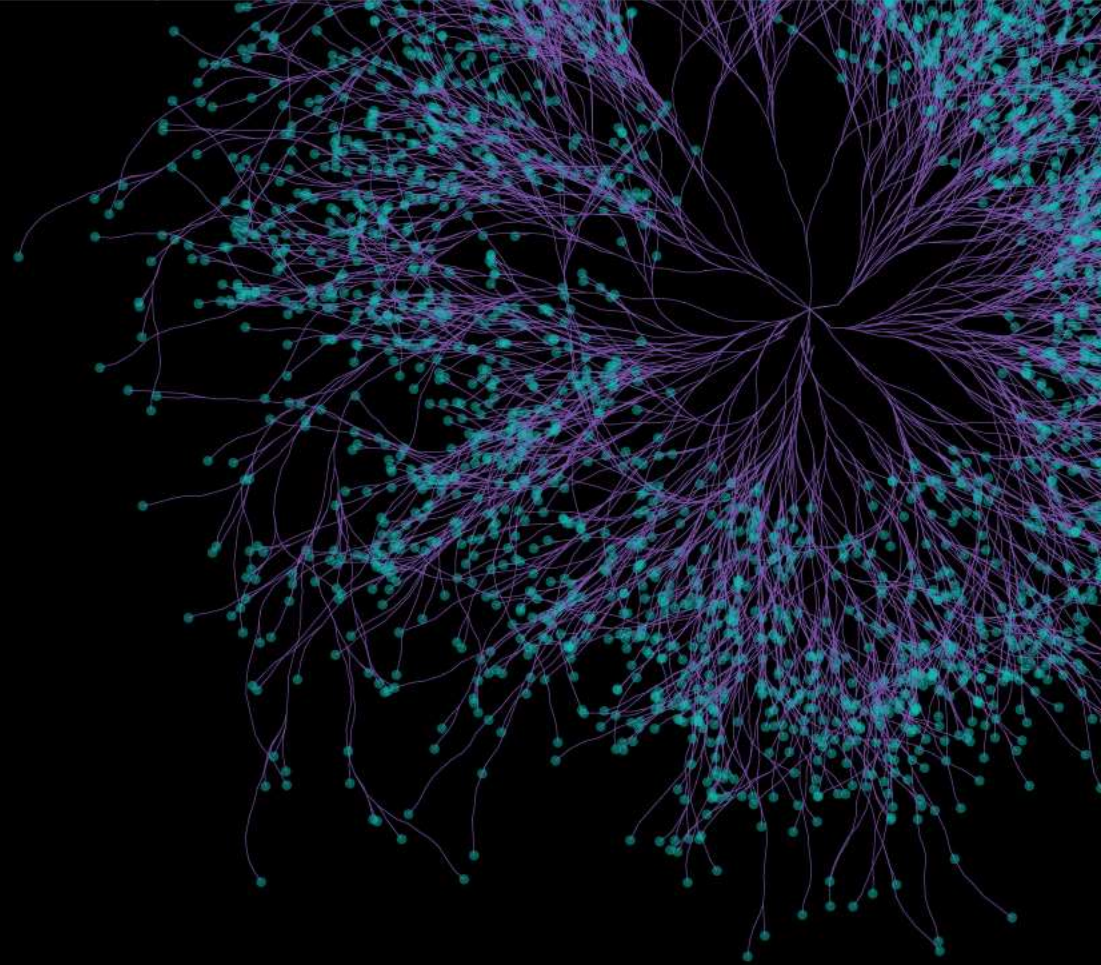
- Make sure you can measure success and failure – metrics, metrics, metrics!
- If a feature is static index it
- Don’t use stored values for static features, always rely on DocValues
- If you are not happy with the performance of search, consider a trip to Las Vegas

you may end up improving performance by 40% ☺



annotated by gritmind

ACTIVATE



Thank you! Ευχαριστούμε! Grazie!

And btw: we are hiring a senior search relevance engineer!

bit.ly/2Orb8bc

https://www.bloomberg.com/careers

Malvina Josephidou & Diego Ceccarelli
Bloomberg

@malvijosephidou |@diegoceccarelli

#Activate18 #ActivateSearch

Powered by Lucidworks

annotated by gritmind