Learning to Rank @ Reddit

reddit

It Us

Doug Turnbull

Charles Njoroge

http://softwaredoug.com
http://reddit.com/u/softwaredoug

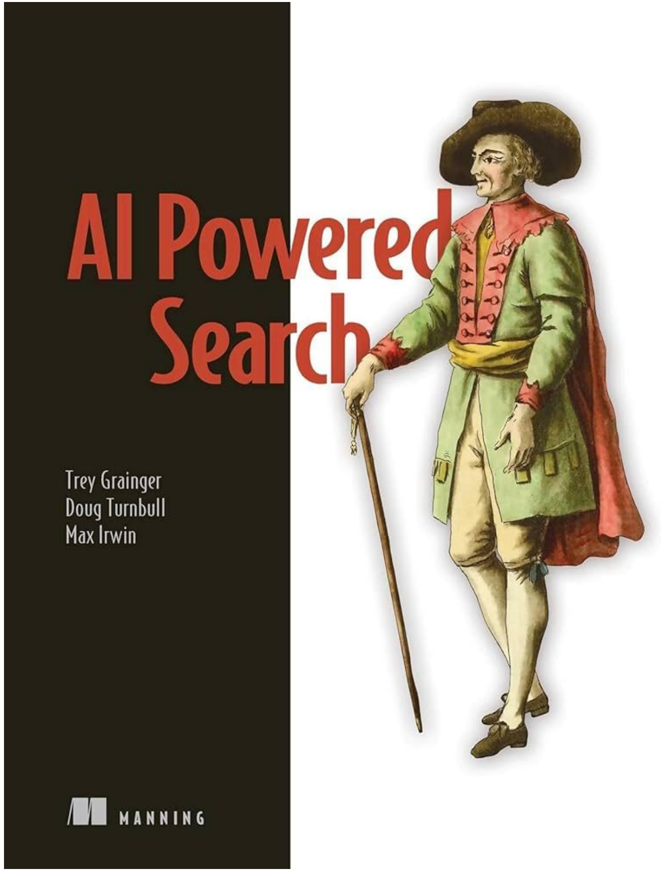
AI Powered Search

https://aipoweredsearch.com/

Discount Code: **bb24 for 45% off**

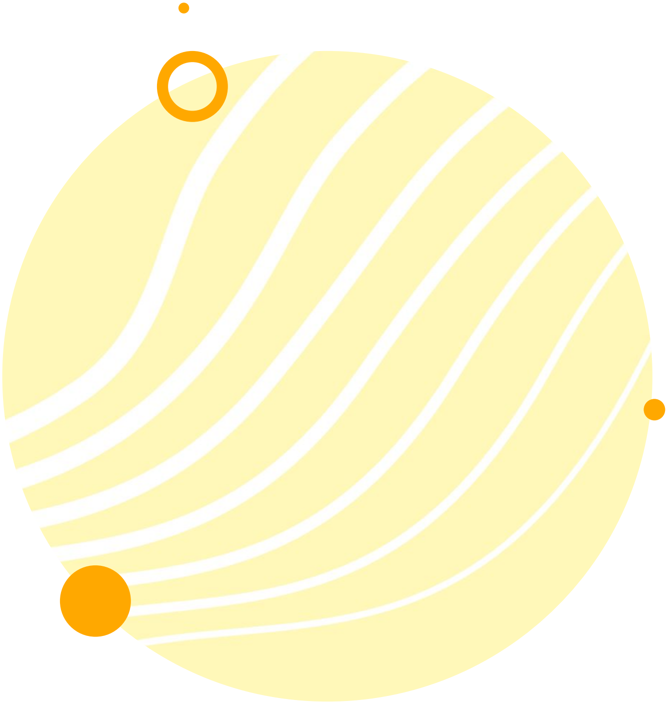
Being published soon!

Join the community!

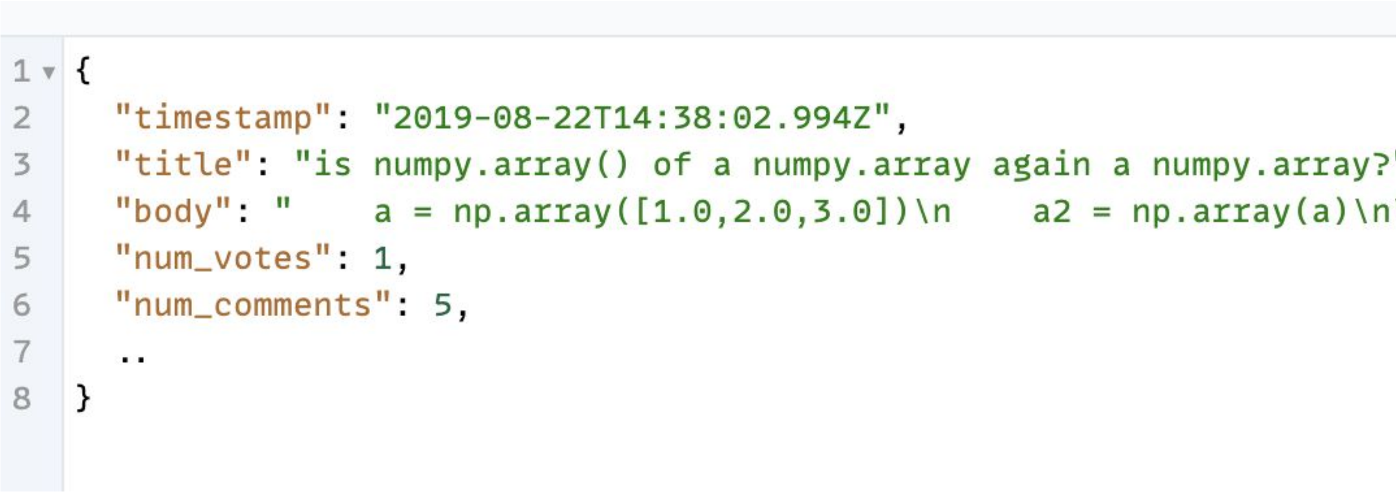


Today’s Topic

How do we add Learning to Rank to an existing, mostly working, high scale search system?



Reddit Search information?

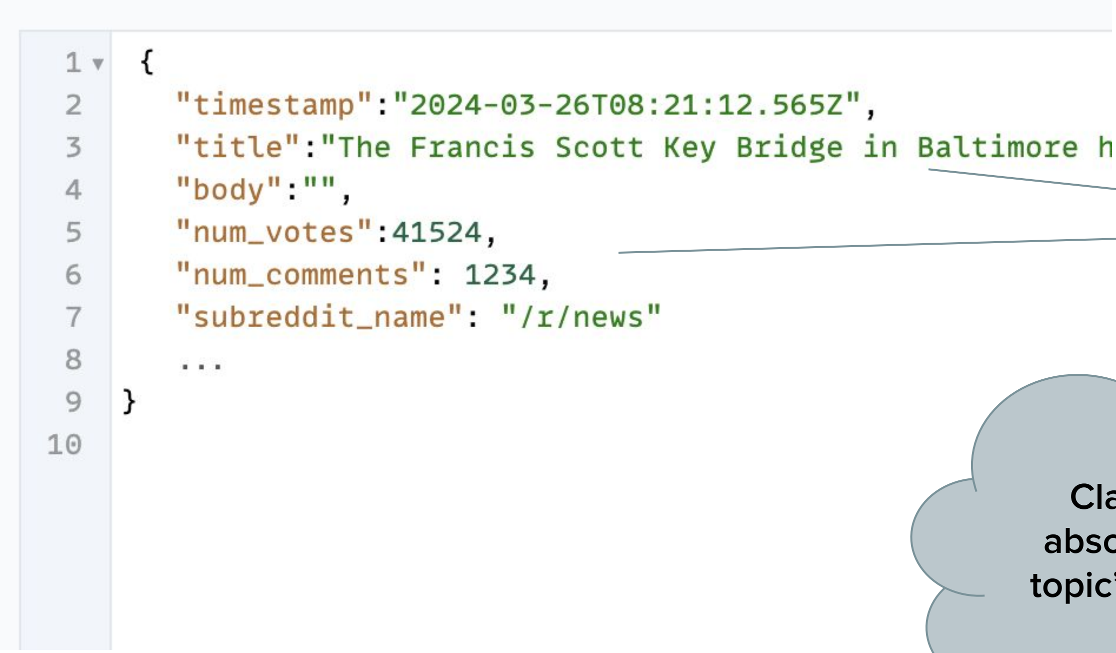


```
{
  "timestamp": "2019-08-22T14:38:02.994Z",
  "title": "is numpy.array() of a numpy.array again a numpy.array?"
  "body": "    a = np.array([1.0,2.0,3.0])\n    a2 = np.array(a)\n
  "num_votes": 1,
  "num_comments": 5,
  ..
}
```

First glance: classic, text-heavy informational search

… but with a social twist

Breaking news searches, ie “key bridge collapse”

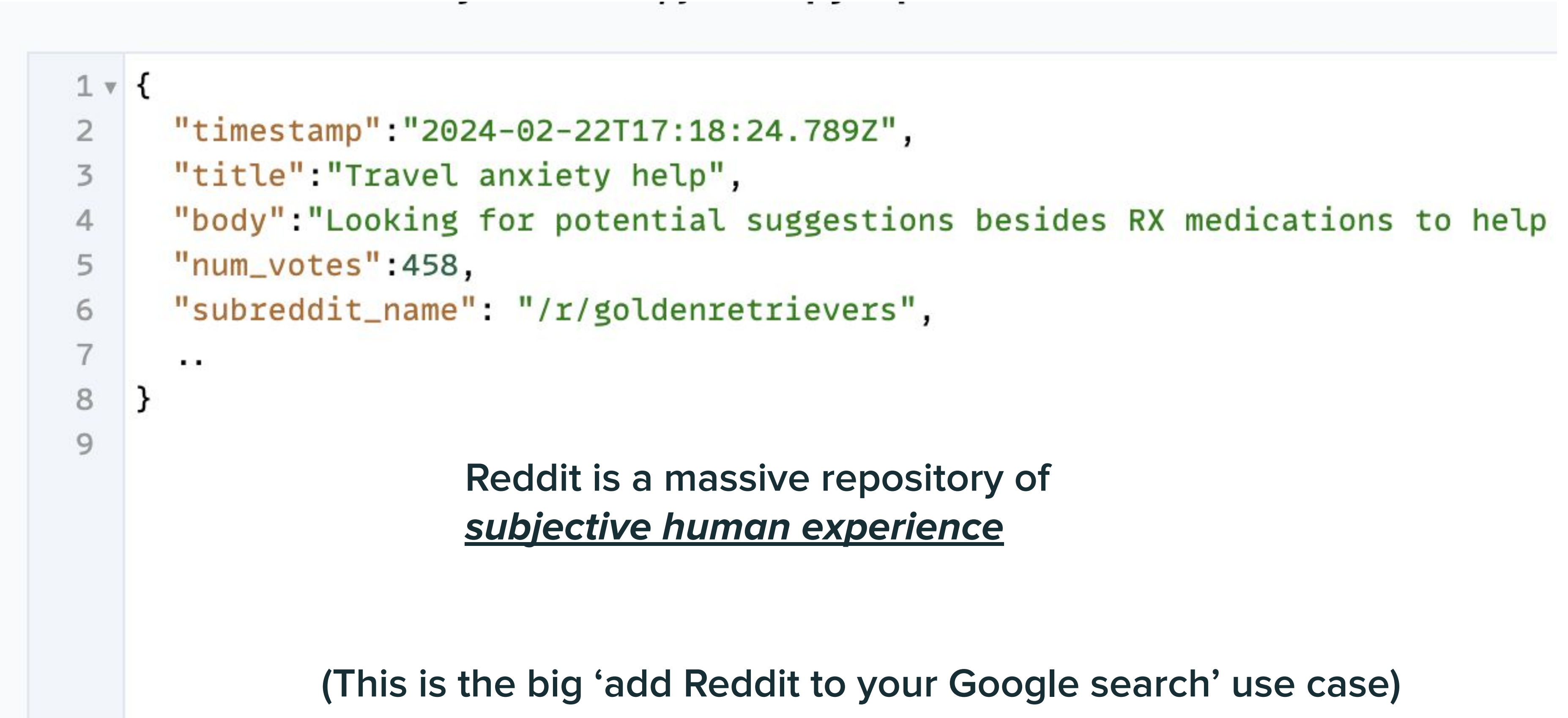


```
 {
   "timestamp":"2024-03-26T08:21:12.565Z",
   "title":"The Francis Scott Key Bridge in Baltimore h
   "body":"",
   "num_votes":41524,
   "num_comments": 1234,
   "subreddit_name": "/r/news"
     ...
}
```

Care about recency / popularity

Classic “what’s the absolute latest on this topic” information need

… and sometimes very personal

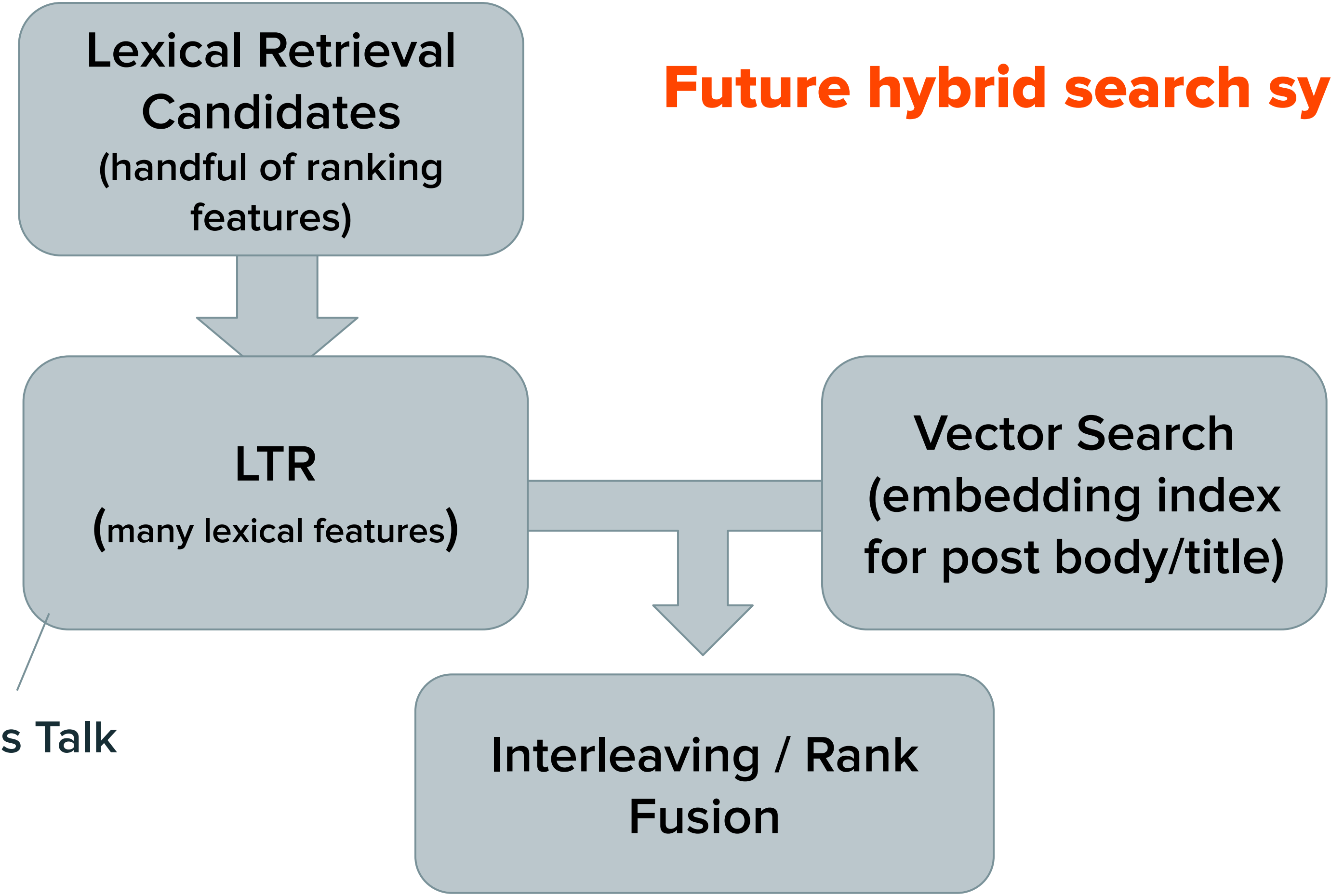


```
{
  "timestamp":"2024-02-22T17:18:24.789Z",
  "title":"Travel anxiety help",
  "body":"Looking for potential suggestions besides RX medications to help
  "num_votes":458,
  "subreddit_name": "/r/goldenretrievers",
  ..
}
```

Reddit is a massive repository of
subjective human experience

(This is the big ‘add Reddit to your Google search’ use case)

Future hybrid search system



Lexical Retrieval Candidates (handful of ranking features)

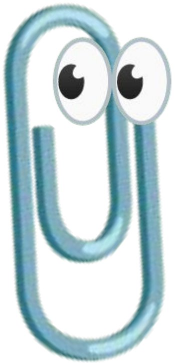
LTR (many lexical features)

Vector Search (embedding index for post body/title)

Interleaving / Rank Fusion

Today's Talk

LTR over ‘lexical’ - Why do we care?



Mr. ML Model

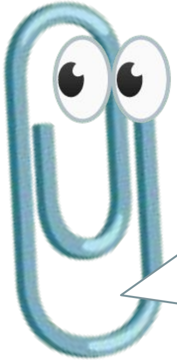
Hi! I’m

Mr. ML Model!

It looks like you’re trying to optimize your search relevance!

Training Data

Query	Post ID	Rel?
Key bridge	1234	1
Key bridge	5678	0
Golden retrieval travel anxiety	12412	1



Mr. ML Model

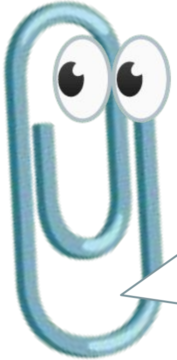
First, give me some examples of relevant / irrelevant search results

(are these any good!?)

Features

- Did the title match the keywords?
- What was the BM25 score of the body?
- How recent was it?
- Did the subreddit match the query?
- …
- ?

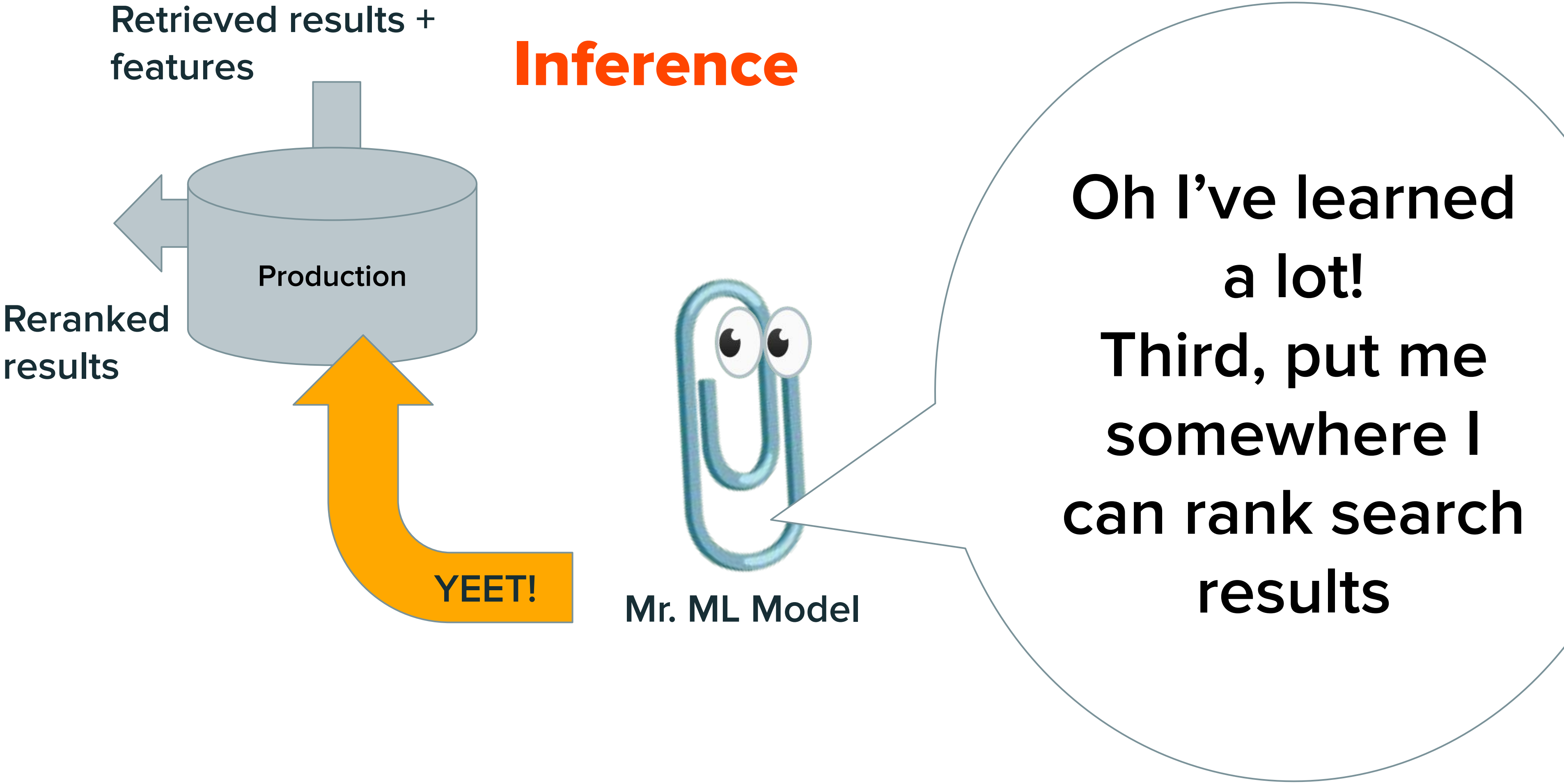
(do these predict relevance!?)



Mr. ML Model

Second, give me some information about query / posts so I can see the patterns

Inference



Retrieved results + features

Production

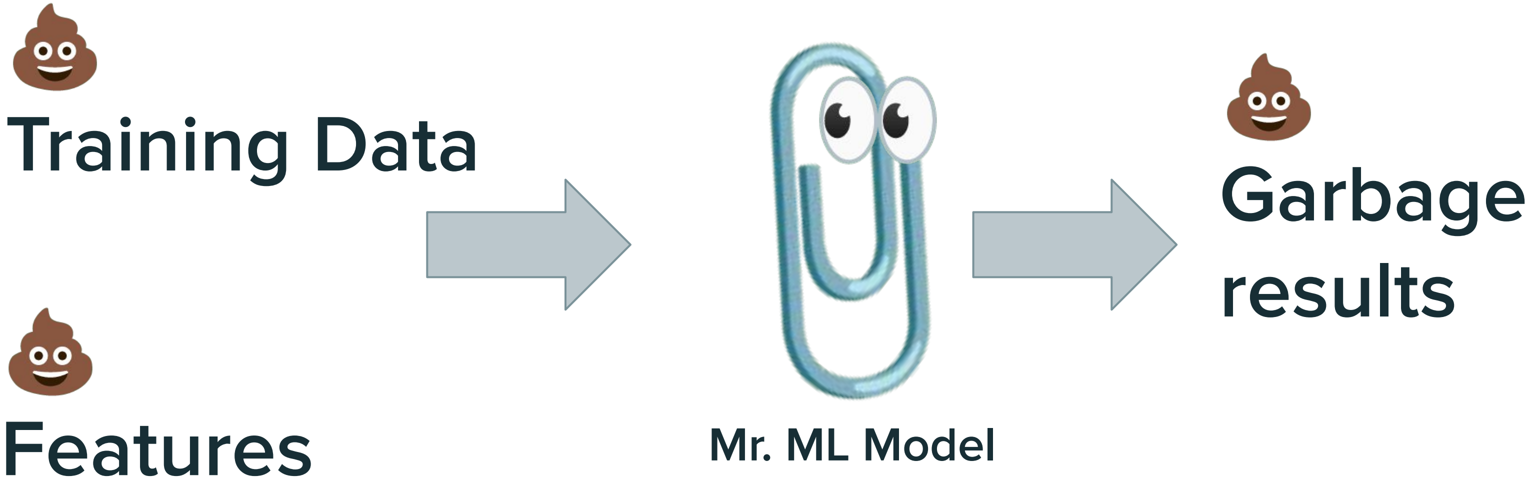
Reranked results

YEET!

Mr. ML Model

Oh I’ve learned a lot!
Third, put me somewhere I can rank search results

Answering Mr. ML Models questions as a forcing function



💩 Training Data

💩 Features

Mr. ML Model

💩 Garbage results

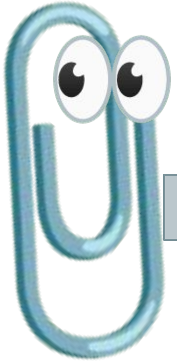
Answering Mr. ML Models questions as a forcing function

🎂

Training Data

😍

Features



Mr. ML Model

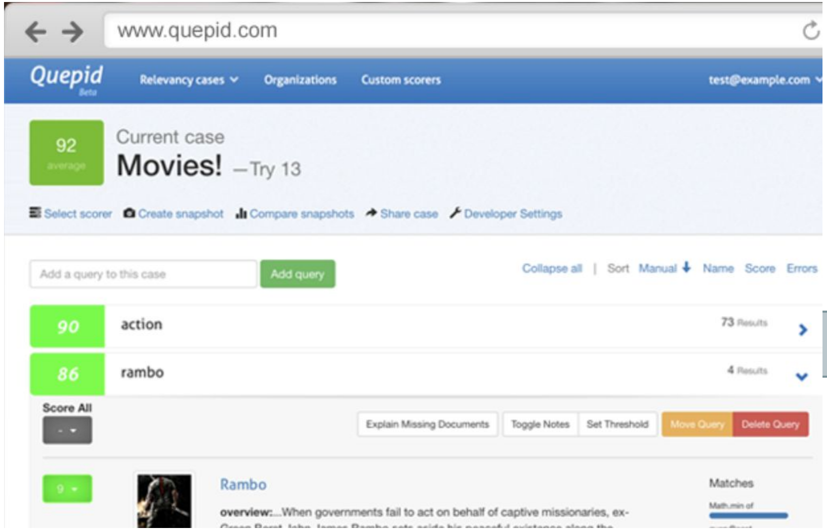
… Even without Mr. ML Model

🎂

Training Data

😍

Features



overview:...When governments fail to act on behalf of captive missionaries, ex-

🤓🔨

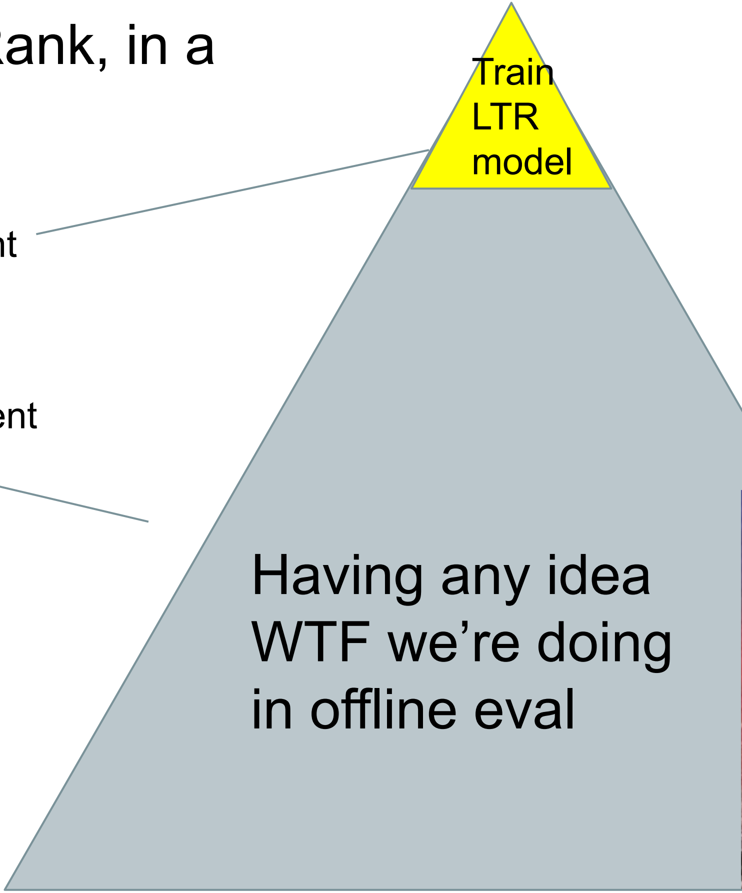
(hand tuned features to meet training data in tool like Quepid)

Training Data + Feature Selection

Learning to Rank, in a nutshell

5% of time spent

95% of time spent



Train LTR model

Having any idea WTF we’re doing in offline eval

Doug, having no idea what he’s doing, until we run more real experiments in search bench

Training Data - started with human eval

Hand labeled results (~1000 queries, 20 per query, head and tail queries)

q=zoolander

👍 Zoolander 2 Trailer

👎 Meet my puppy name “Zoolander”

👍 I love the part where he does “Magnum”

… To derive “engagement judgments”

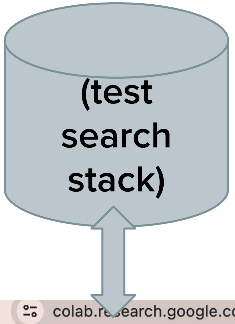
	Relative weights
Position	0.05
Click	0.05
Click + dwell	0.9

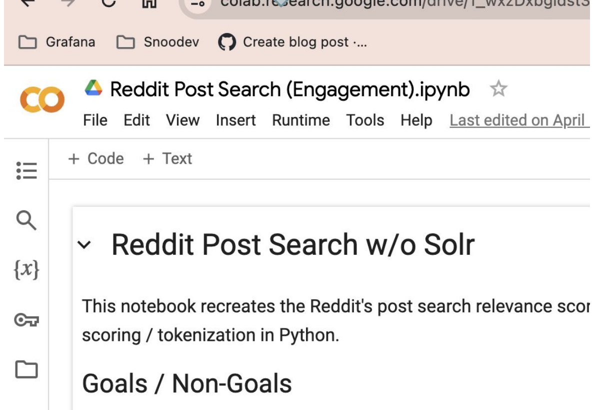
(30 / 60 day sums)

🤓 - Good sign!

→ Human labelers agree w/ click+dwell

Next steps - USE the judgments





Reddit Post Search w/o Solr

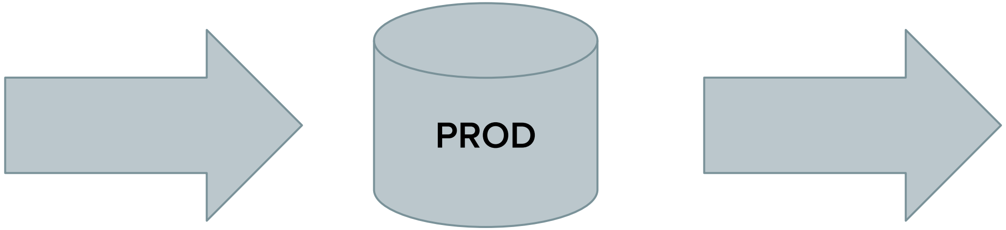
This notebook recreates the Reddit's post search relevance scor
scoring / tokenization in Python.

Goals / Non-Goals

(Offline Experiments)

🤓🔨

(manual relevance
tuning at retrieval)

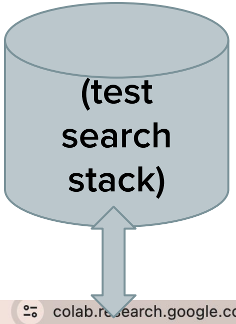


NDCG go up?
Ship to A/B!

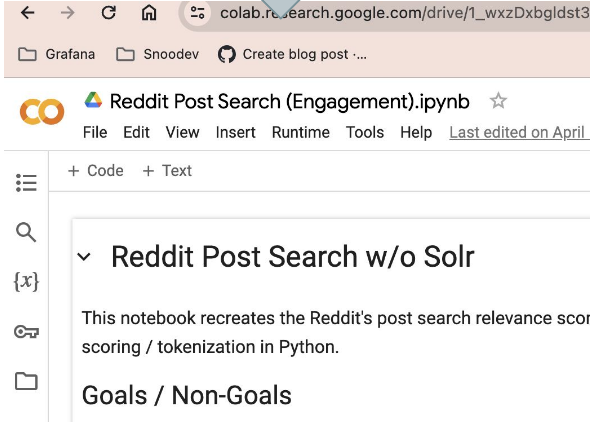
Agreement with
offline?

👍 Generally good

… And train w/ judgments



(test search stack)

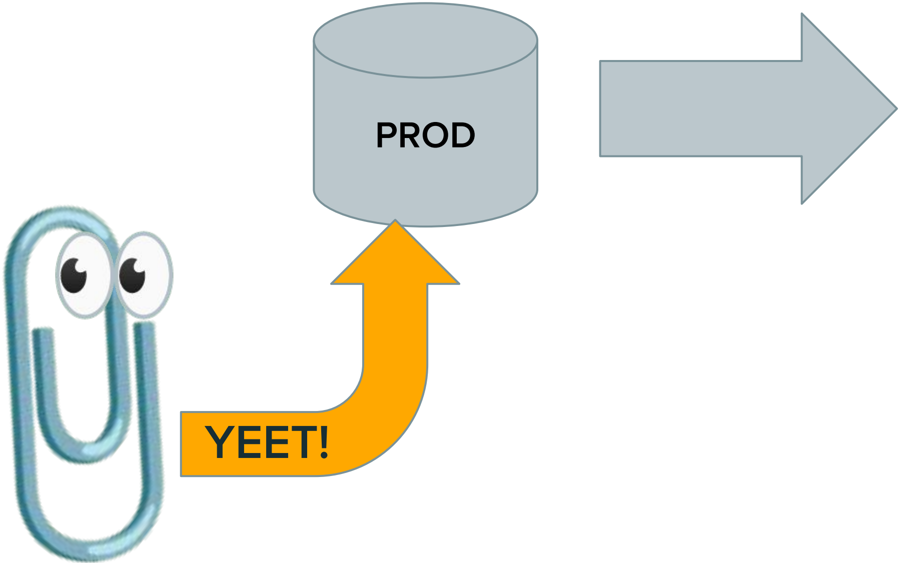


Reddit Post Search w/o Solr

This notebook recreates the Reddit's post search relevance scor scoring / tokenization in Python.

Goals / Non-Goals

(Offline Experiments)



PROD

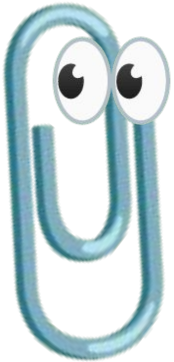
YEET!

NDCG go up?
Ship to A/B!

Agreement with offline?

… Training w/ judgments

Query	Post ID	Rel?	Title Match?
Key bridge	1234	1	1
Key bridge	5678	0	1
Golden retrieval travel anxiety	12412	1	1



Mr. ML Model

PROBLEM - engagement based judgments have SOME relationship to document!

(even irrelevant ones) - why?

… We sample other queries for negative labels

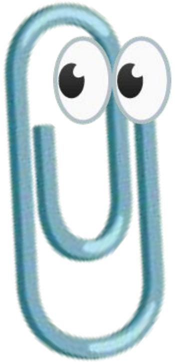
Query	Post ID	Rel?	Title Match?
Key bridge	1234	1	1
Key bridge	5678	0	1
Key bridge	12412	0	0
Golden retrieval travel anxiety	12412	1	1

Inject as irrelevant

(Inject some N random other query labels as negative for each query)

Mr. ML Model can see the patterns better

Post ID	Rel?	Title Match?
1234	1	1
5678	0	1
12412	0	0
12412	1	1

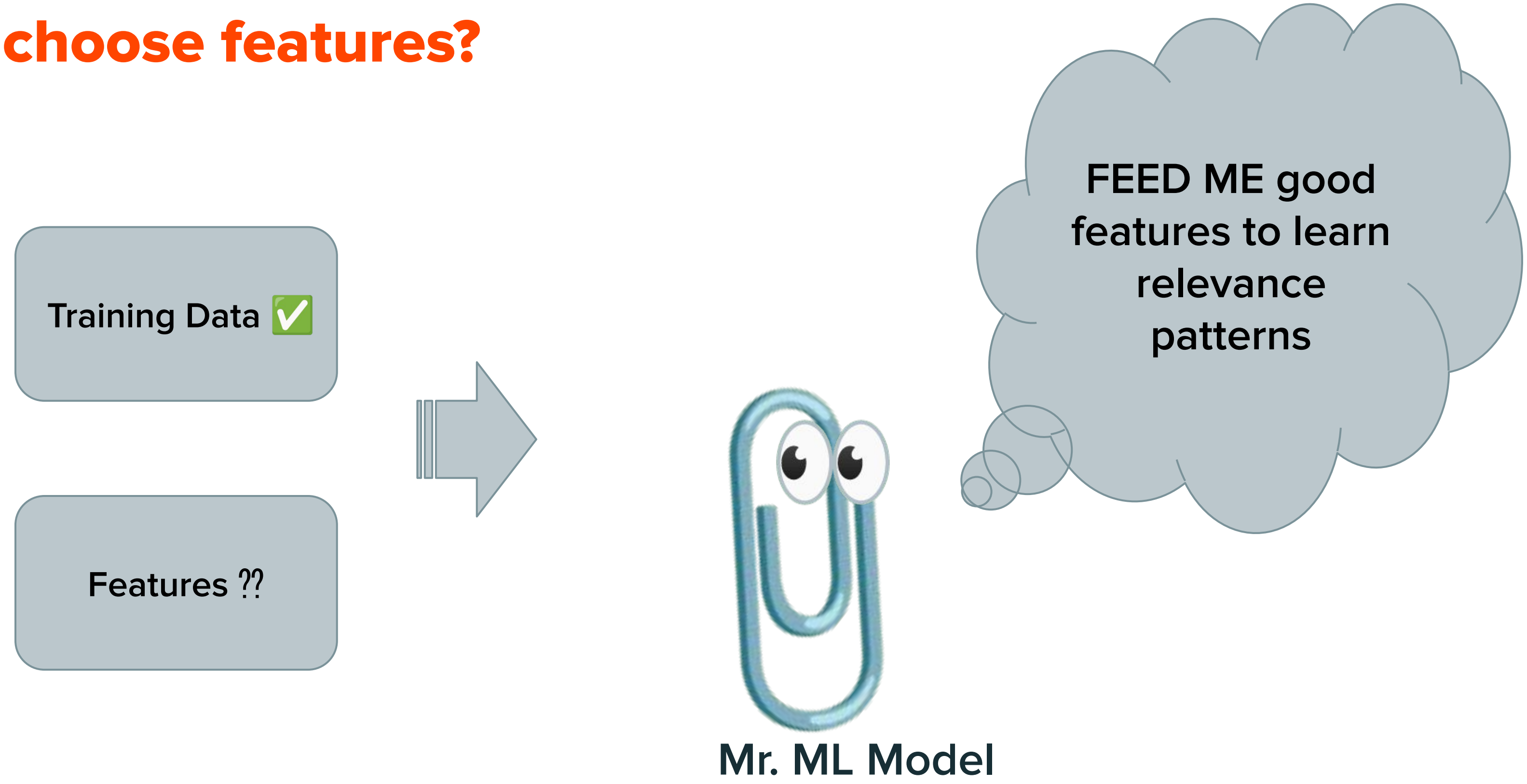


Mr. ML Model

I see now:

no title match == maybe irrelevant

How to choose features?



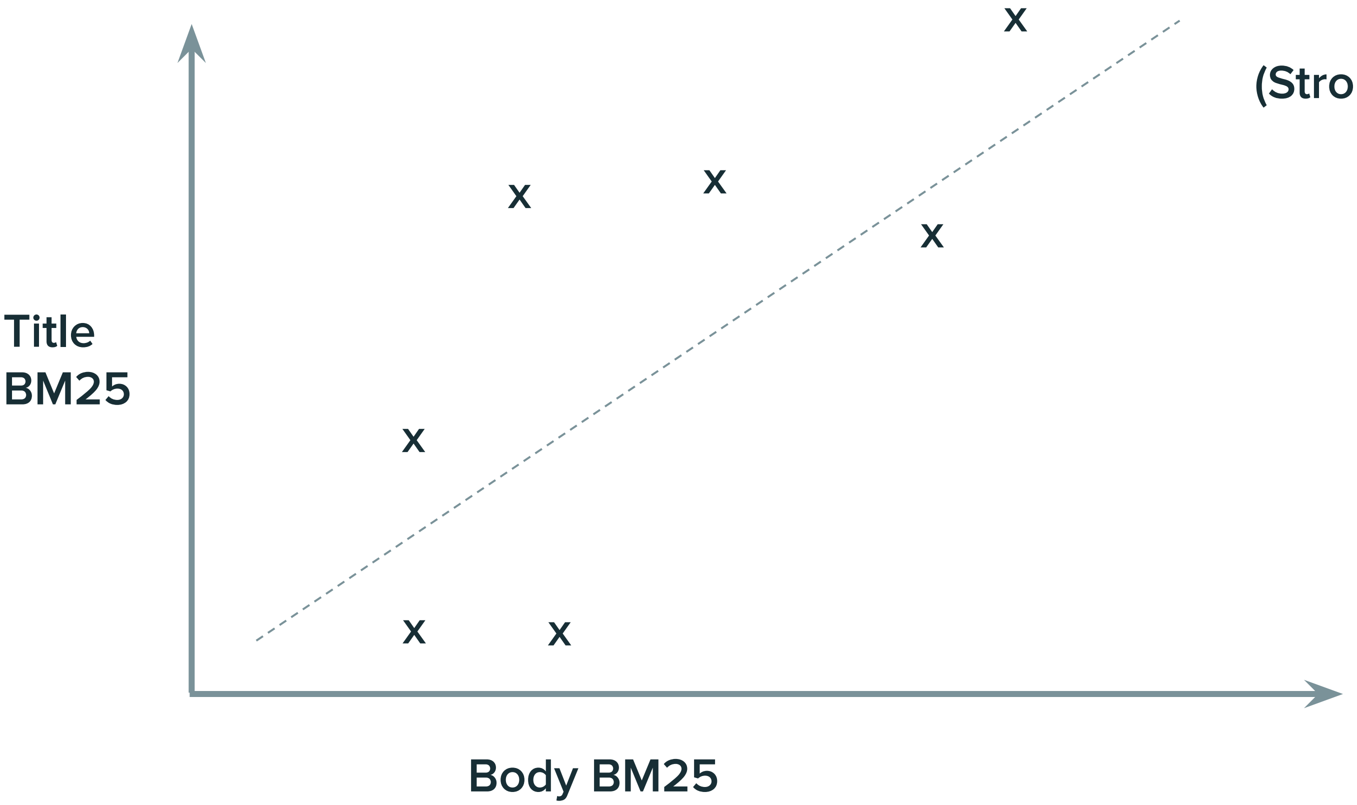
Training Data ✅

Features ⁇

FEED ME good features to learn relevance patterns

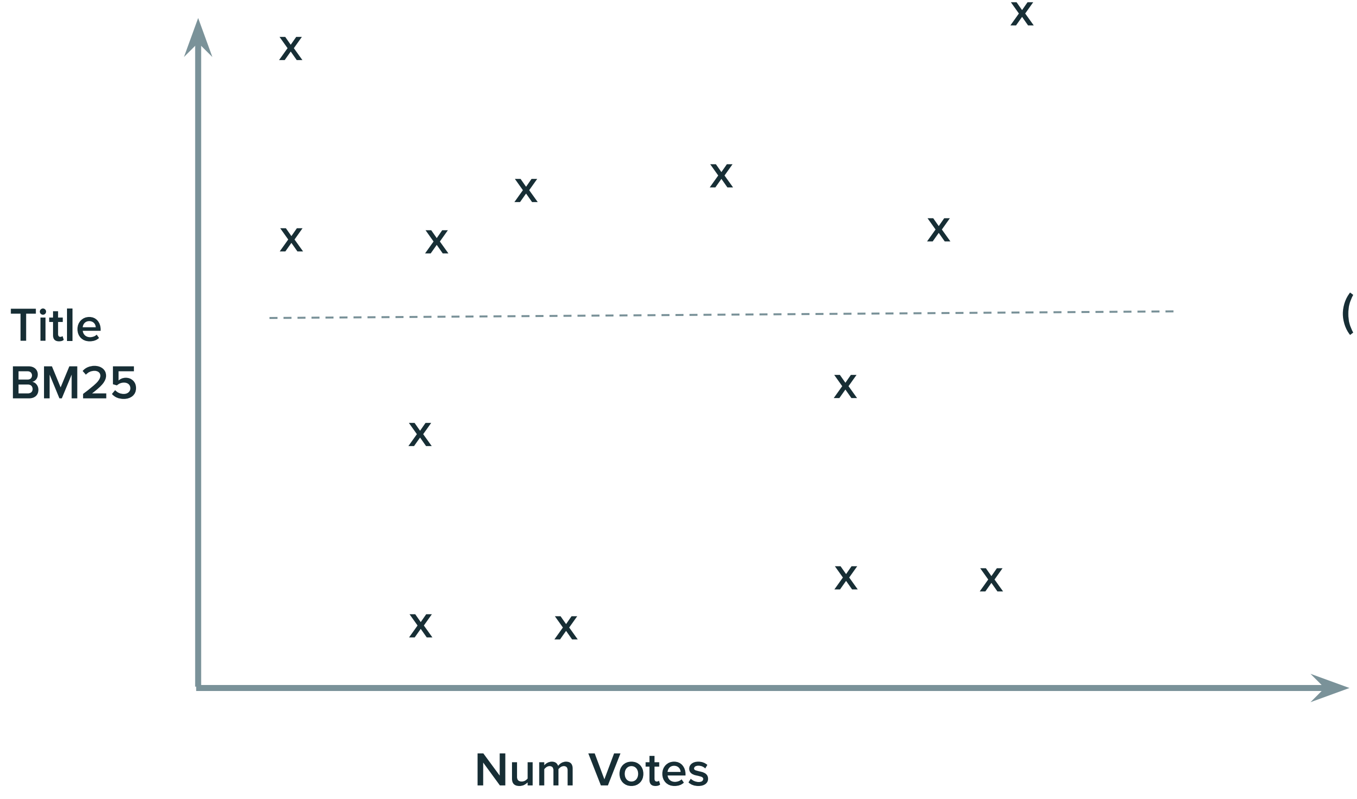
Mr. ML Model

Features often heavily correlated in LTR



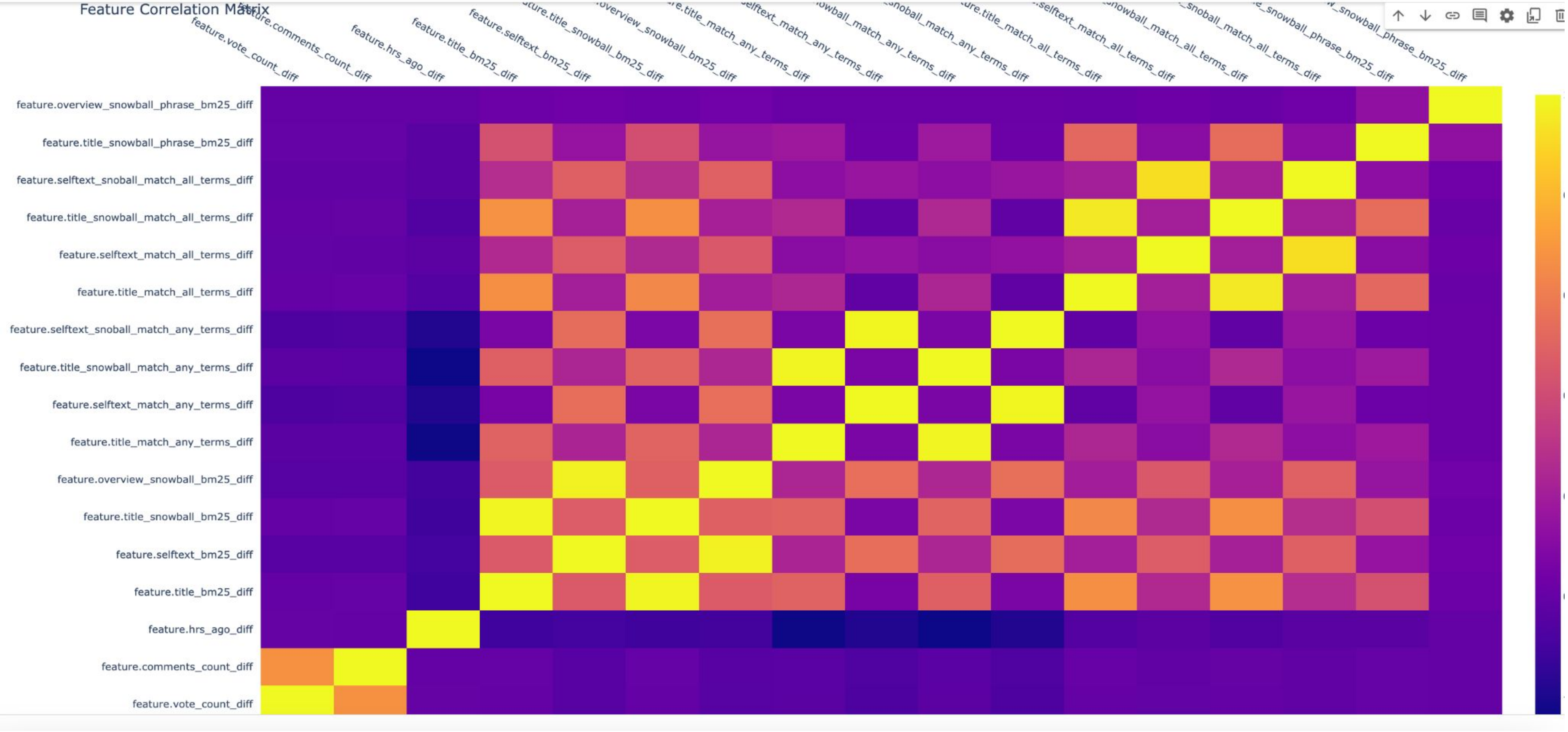
(Strong correlation) 😕

Good features add information

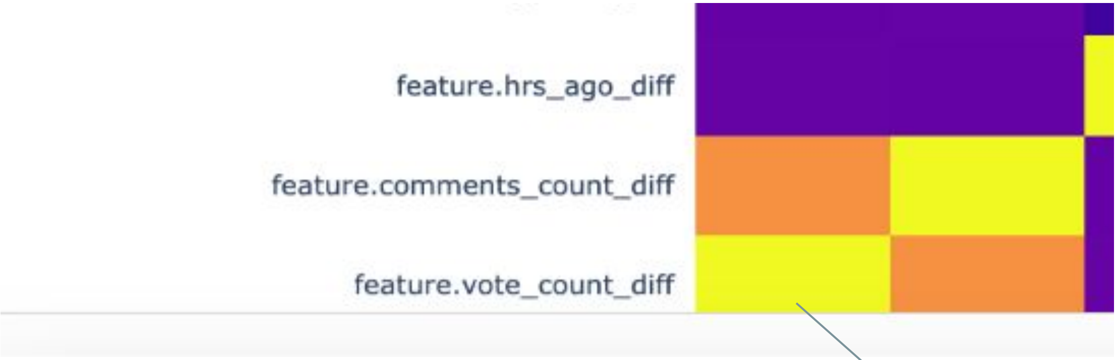


(No correlation) 👍

Analyze via correlation matrix



Analyze via correlation matrix

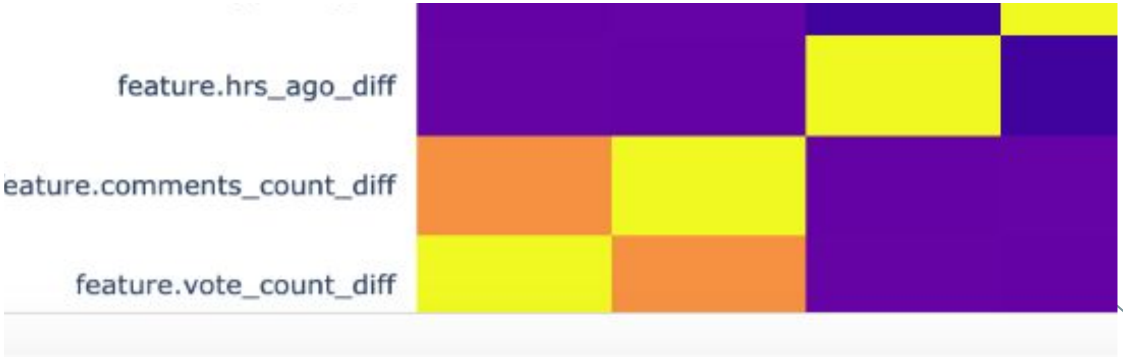


Votes / Num Comments Correlate, don’t add much new info relative to each other

Analyze via correlation matrix

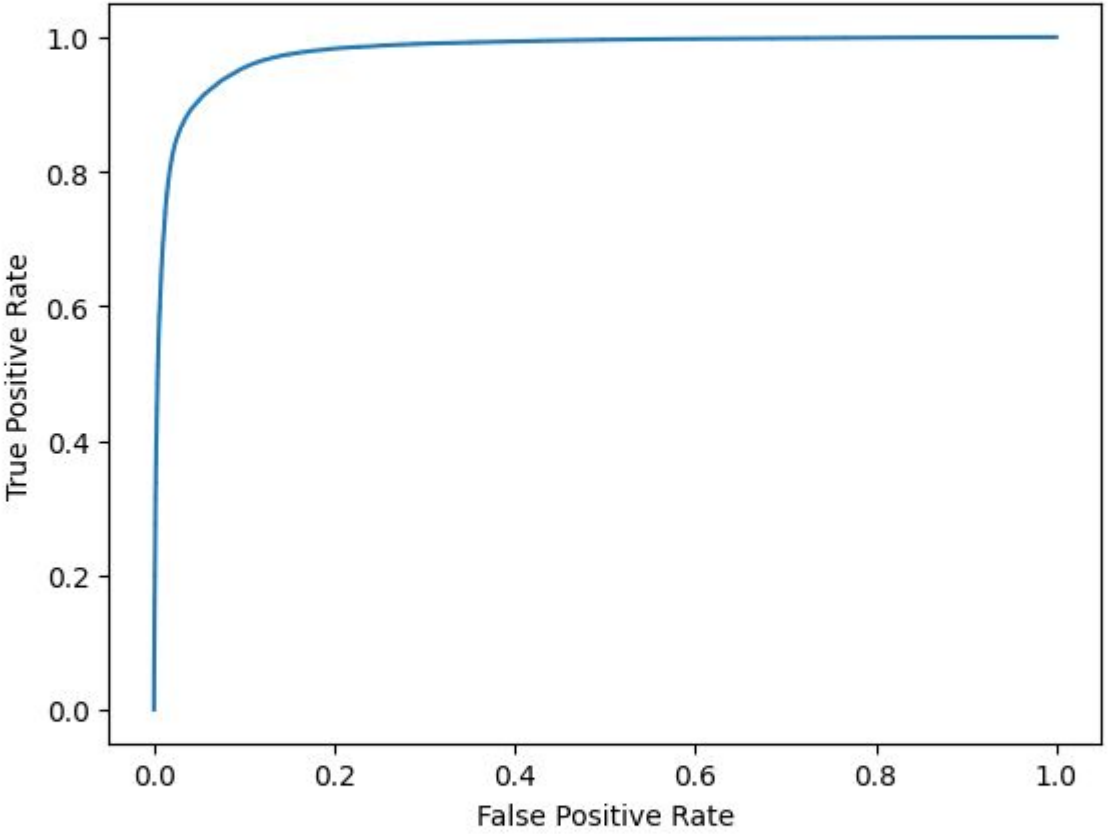
(hrs ago)

Title BM25



But add quite a bit on top of these features

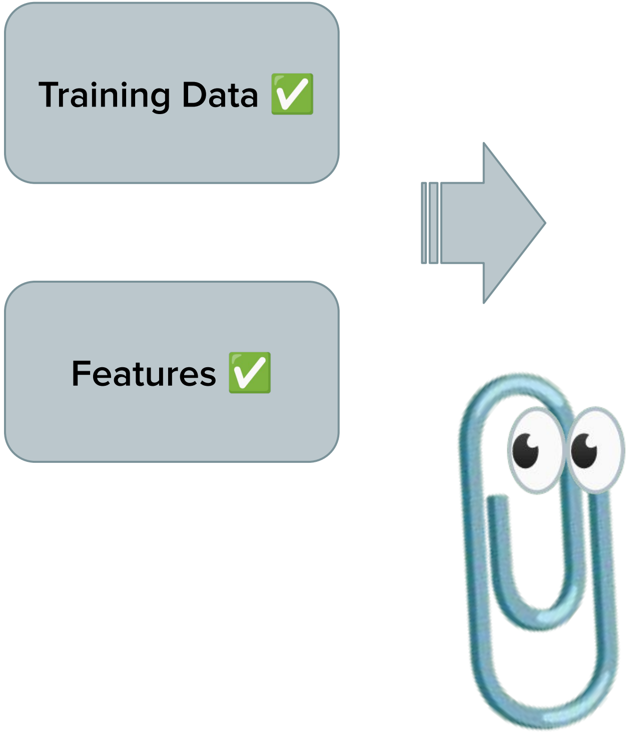
Goal: find INDEPENDENT features, that IMPROVE model



Feature adds value when:

1. Orthogonal to other features
2. Improves model
3. Is readily accessible and computationally feasible to compute

How to choose features?



Training Data ✅

Features ✅

Mr. ML Model

Model architecture:

Lots of Choices, main requirements:

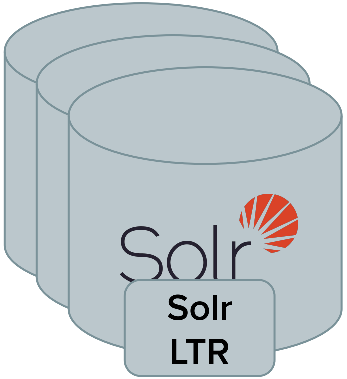
- Listwise / pairwise loss function
- Handle non-linear and correlated features

We chose

- LambdaMART loss
- Deep learning model

Yeeting Features + Models to Prod

Choosing Solr LTR Plugin



(Lexical)
Feature Calculation + Model Inference

Solr functionality for

- Feature calculation
- Top N Reranking

Pros / Cons Solr LTR vs Reddit extra

	Solr LTR	Reddit’s existing ML infra
Query-dependent features?	Yes	Not easily
Exists (at Reddit?)	No	Yes
Time horizon of content	~19 years	90 days
Features available	Minimal	Extensive
Network hops	None	Several
Types of models	Limited	Extensive
Model store size	~1MB*	Unbounded
Vertical scalability	Shared with Solr	Unshared

🧐 Which to choose?

Solr LTR Plugin

Solr LTR - Reference Guide

Feature Store + Logging

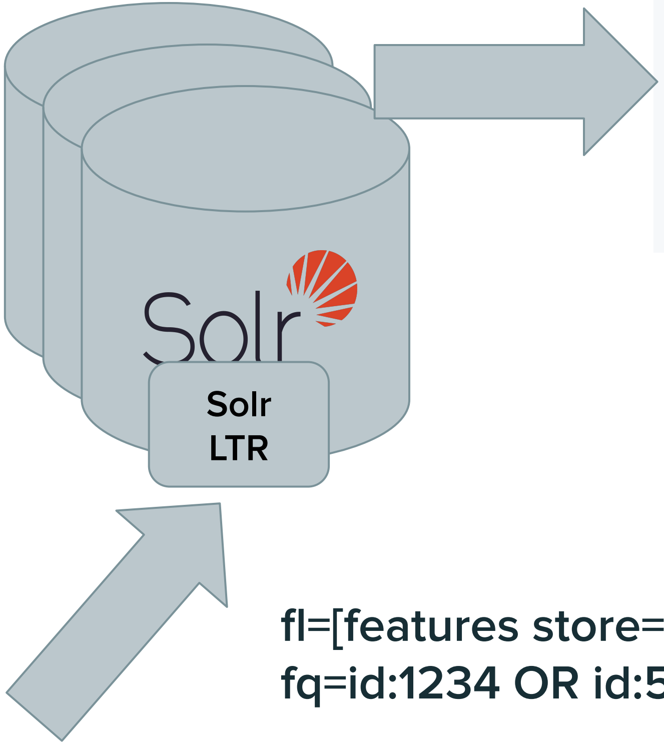
IE From Zero to Solr LTR:

Solr Query DSL

MY_EFI_FEATURE_STORE

```
[
  {
    "store" : "my efi feature_store",
    "name" : "tfidf_sim_a",
    "class" : "org.apache.solr.ltr.feature.SolrFeature",
    "params" : { "q" : "{!dismax qf=text_tfidf}${keywords}" }
  },
  {
    "store" : "my_efi_feature_store",
    "name" : "tfidf_sim_b",
    "class" : "org.apache.solr.ltr.feature.SolrFeature",
    "params" : { "q" : "{!dismax qf=text_tfidf}${keywords}" }
   },
```

Training Time



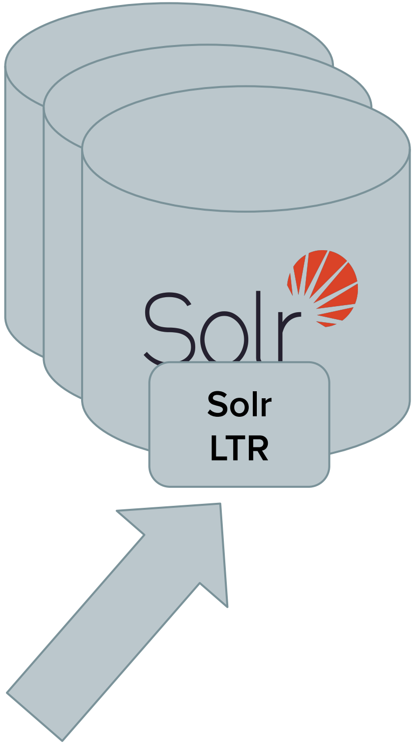
```
[
  {
      “id”: 1234,
      “[features]”: “tfidf_sim_a=1.56,...”
  },
  {
      “id”: 5678,
      “[features]”: “tfidf_sim_a=0.05,...”
  },
  …
```

(training examples for docs 1234… ,
… for query ‘football’)

fl=[features store=my_efi_feature_store efi.keywords=’football’]&
fq=id:1234 OR id:5678 OR id:1010

Keyword “football” posts: 1234, 5678, 1010

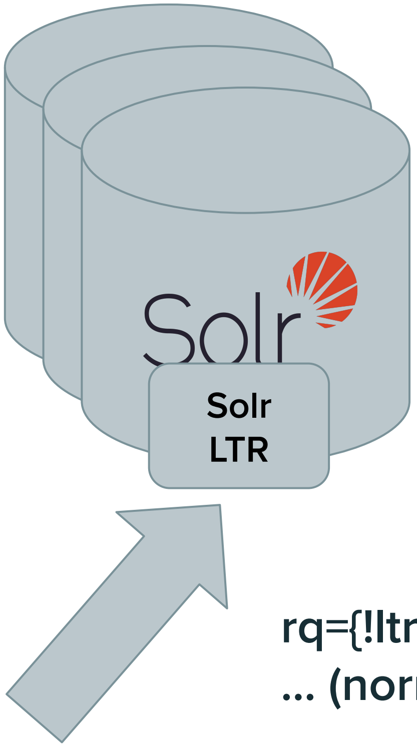
Store model for inference



`Model: foo`

`Store:` my_efi_feature_store

Inference Time



Top N to rerank:

```
[
  {
      “id”: 1234,
      “[features]”: “tfidf_sim_a=1.56,...”
  },
  {
      “id”: 5678,
      “[features]”: “tfidf_sim_a=0.05,...”
  },
  …
```

To model

(Features Computed internal to Solr)

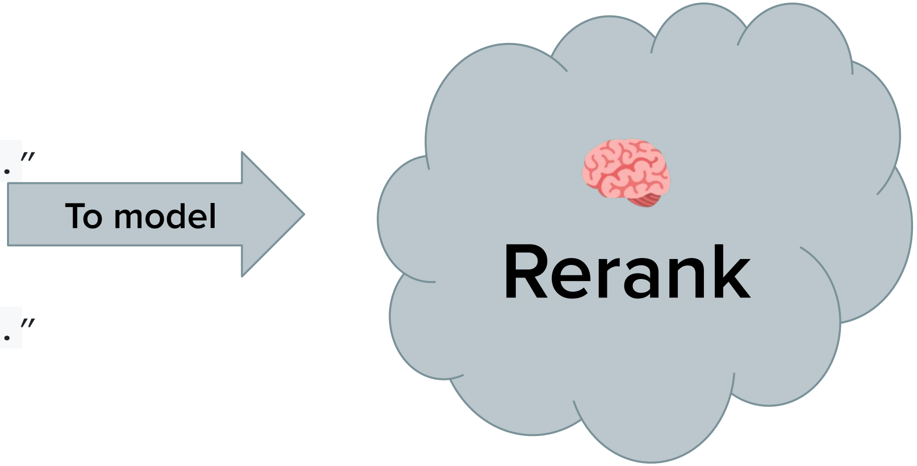
rq={!ltr model=foo-model efi.keywords=’football’]&
… (normal retrieval query)

Inference Time

Top N to rerank:

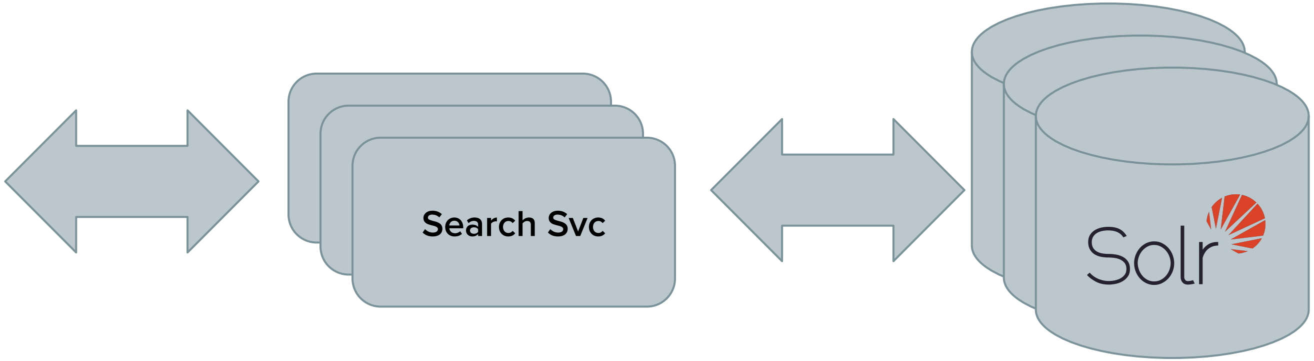
```
[
   {
      “id”: 1234,
      “[features]”: “tfidf_sim_a=1.56,...”
   },
   {
      “id”: 5678,
      “[features]”: “tfidf_sim_a=0.05,...”
   },
   …
```

(Features Computed internal to Solr)



rq={!ltr model=foo-model efi.keywords=’football’]&
… (normal retrieval query)

Our search infra



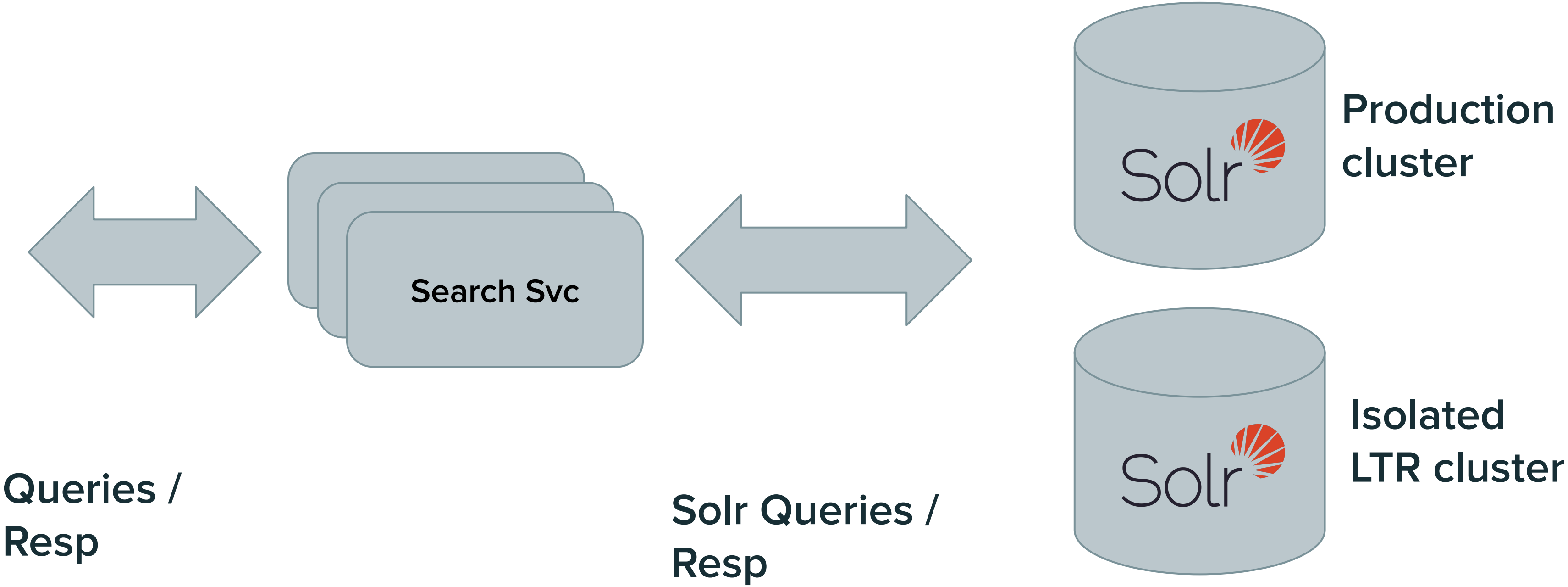
Search Svc

Queries /
Resp

Solr Queries /
Resp

Solr
Cloud

Our search infra: build in isolation or production cluster?



Search Svc

Production cluster

Isolated LTR cluster

Queries / Resp

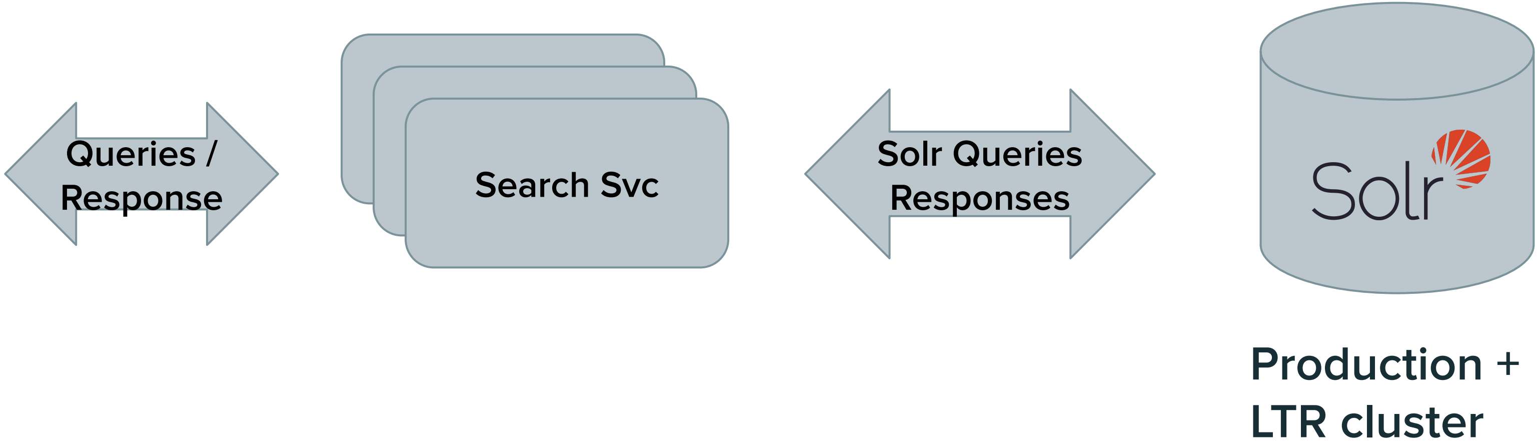
Solr Queries / Resp

Pros / Cons

	Isolated	Single cluster
Implementation speed	Need to add a new cluster	Already built!
Development isolation	Build/ iterate fast independently of other work	Slower b/c of need to integrate with other work
Safety	Faults don’t cascade	Faults affect prod traffic
Experiment confounders	Different latencies	Same latency in prod and experiment
Operational cost	One more cluster to maintain	Maintain two use cases in same cluster
$$$	One more cluster to buy (non-trivial cluster cost)	Vertically scale existing cluster slightly

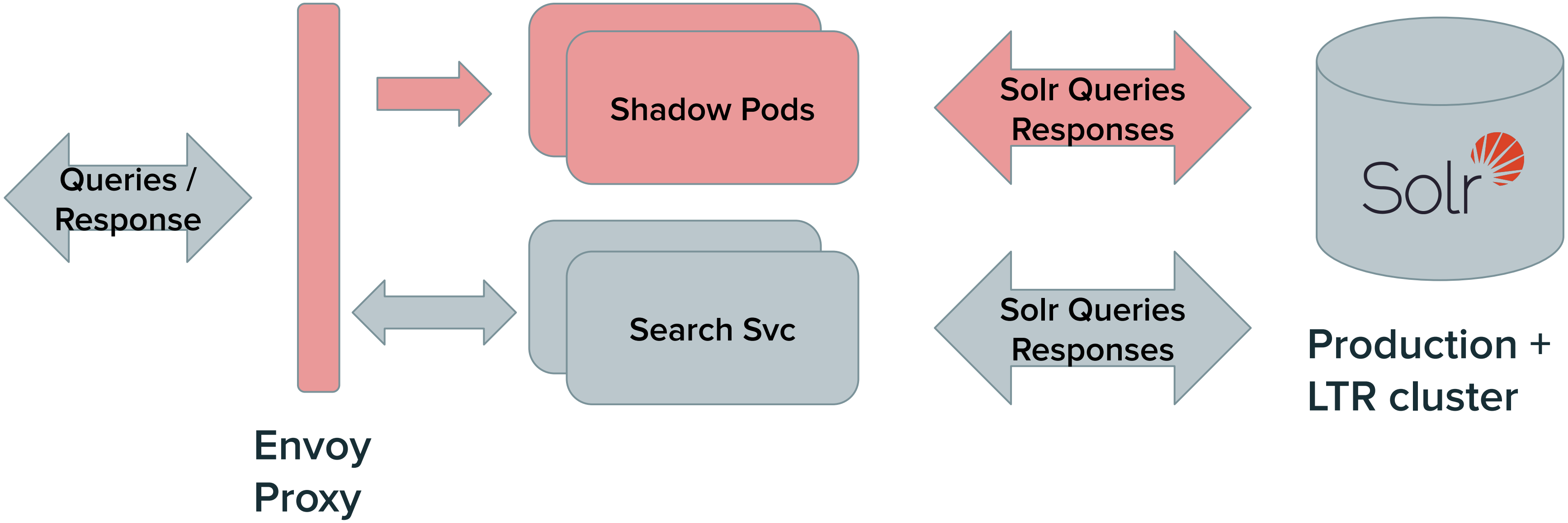
🧐 Which to choose?

Take 1: single cluster

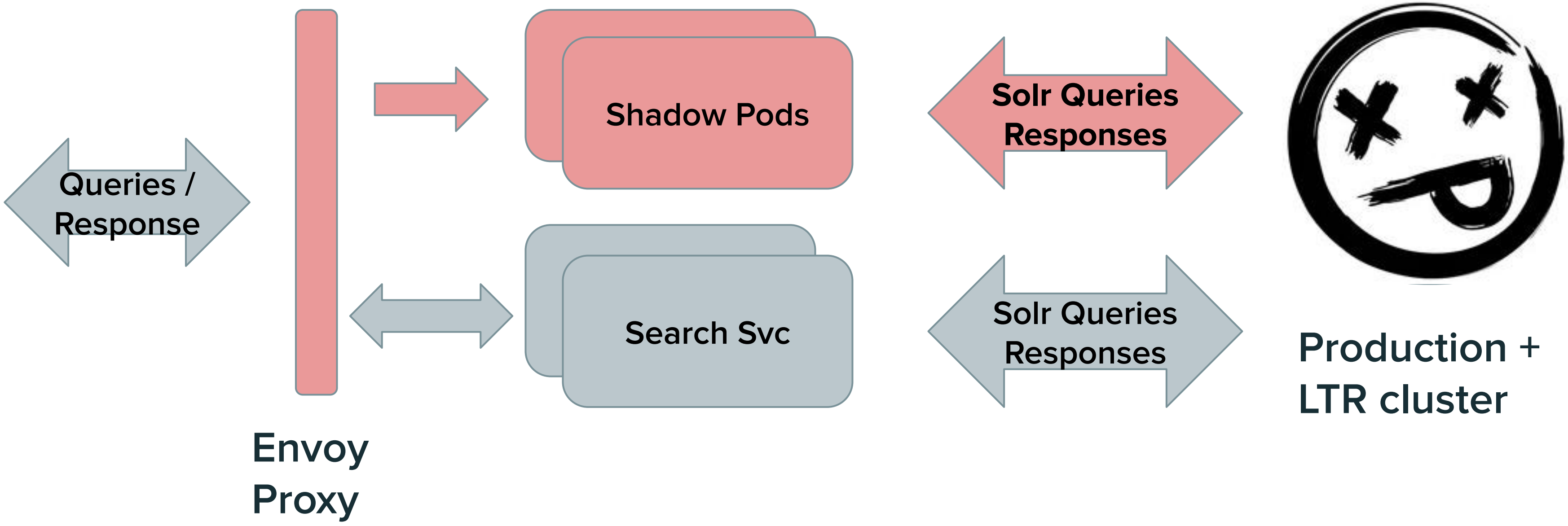


Production +
LTR cluster

Take 1: Envoy for Shadow Traffic (Single Cluster)



Problems with co-location



Queries / Response

Envoy Proxy

Shadow Pods

Search Svc

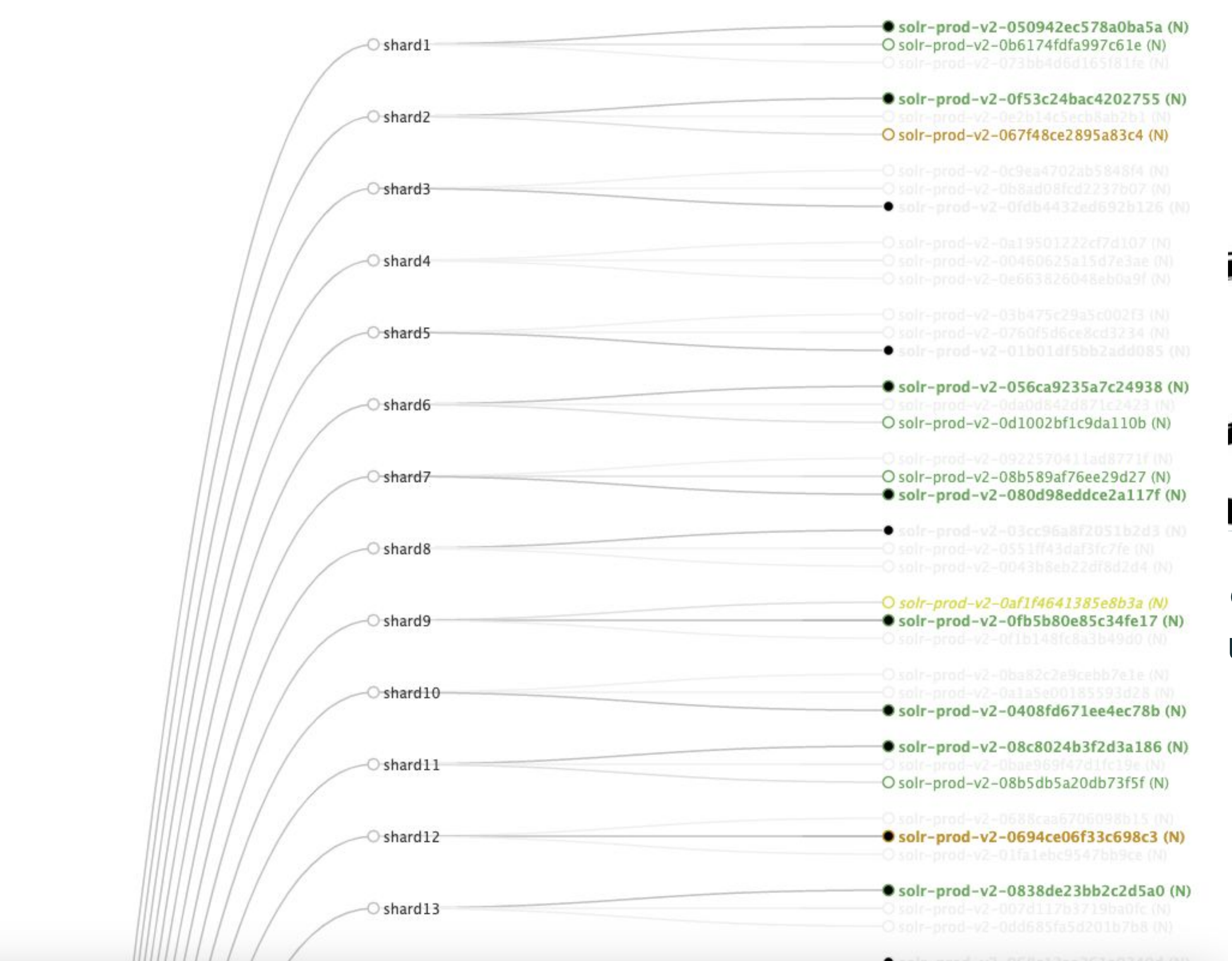
Solr Queries Responses

Solr Queries Responses

Production + LTR cluster

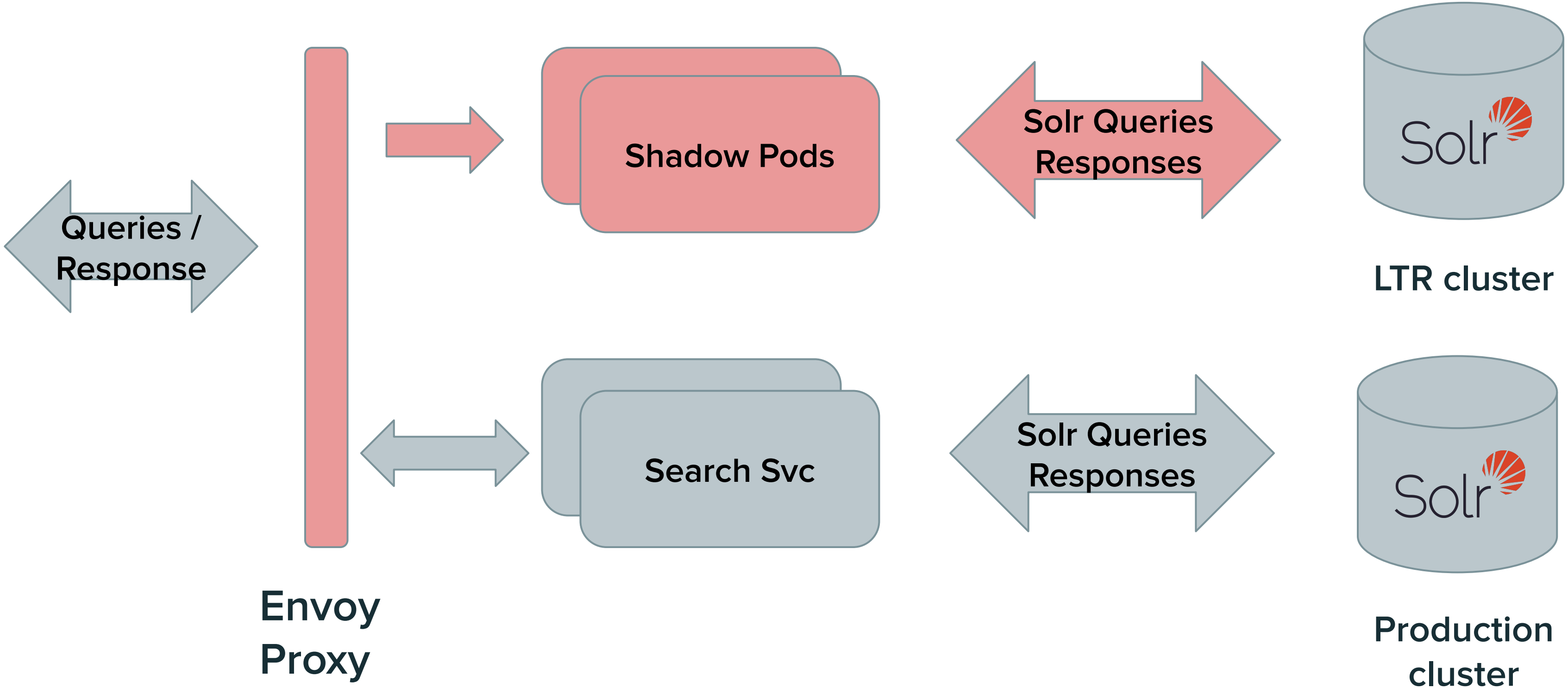
Proble

Queries
Respons

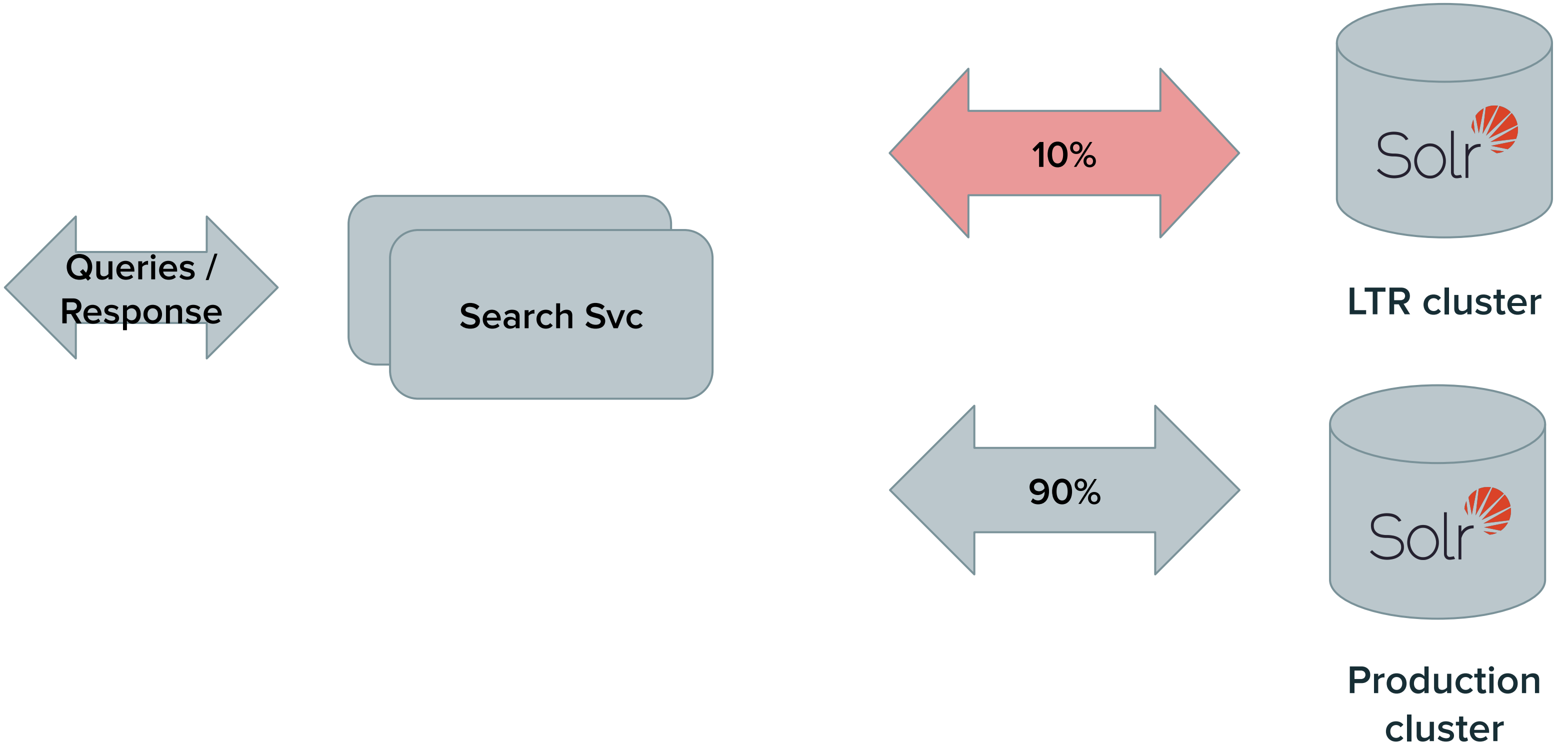

shard1
solr-prod-v2-050942ec578a0ba5a (N)
solr-prod-v2-0b6174fdfa997c61e (N)
solr-prod-v2-073bb4d6d165f81fe (N)
shard2
solr-prod-v2-0f53c24bac4202755 (N)
solr-prod-v2-0e2b14c5ecb8ab2b1 (N)
solr-prod-v2-067f48ce2895a83c4 (N)
shard3
solr-prod-v2-0c9ea4702ab5848f4 (N)
solr-prod-v2-0b8ad08fcd2237b07 (N)
solr-prod-v2-0fdb4432ed692b126 (N)
shard4
solr-prod-v2-0a19501222cf7d107 (N)
solr-prod-v2-00460625a15d7e3ae (N)
solr-prod-v2-0e663826048eb0a9f (N)
shard5
solr-prod-v2-03b475c29a5c002f3 (N)
solr-prod-v2-0760f5d6ce8cd3234 (N)
solr-prod-v2-01b01df5bb2add085 (N)
shard6
solr-prod-v2-056ca9235a7c24938 (N)
solr-prod-v2-0da0d842d871c2423 (N)
solr-prod-v2-0d1002bf1c9da110b (N)
shard7
solr-prod-v2-092257041 1ad8771f (N)
solr-prod-v2-08b589af76ee29d27 (N)
solr-prod-v2-080d98eddce2a117f (N)
shard8
solr-prod-v2-03cc96a8f2051b2d3 (N)
solr-prod-v2-0551ff43daf3fc7fe (N)
solr-prod-v2-0043b8eb22df8d2d4 (N)
shard9
solr-prod-v2-0af1f4641385e8b3a (N)
solr-prod-v2-0fb5b80e85c34fe17 (N)
solr-prod-v2-0f1b148fc8a3b49d0 (N)
shard10
solr-prod-v2-0ba82c2e9cebb7e1e (N)
solr-prod-v2-0a1a5e00185593d28 (N)
solr-prod-v2-0408fd671ee4ec78b (N)
shard11
solr-prod-v2-08c8024b3f2d3a186 (N)
solr-prod-v2-0bae969f47d1fc19e (N)
solr-prod-v2-08b5db5a20db73f5f (N)
shard12
solr-prod-v2-0688caa6706098b15 (N)
solr-prod-v2-0694ce06f33c698c3 (N)
solr-prod-v2-01fa1ebc9547bb9ce (N)
shard13
solr-prod-v2-0838de23bb2c2d5a0 (N)
solr-prod-v2-007d117b3719ba0fc (N)
solr-prod-v2-0dd685fa5d201b7b8 (N)

ction +
uster

Take 2: Isolated clusters



User-level Testing w/ traffic splitting



+ Solr + Learning to Rank

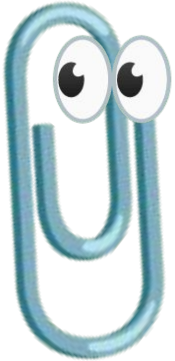
Java

Solr
Solr
LTR

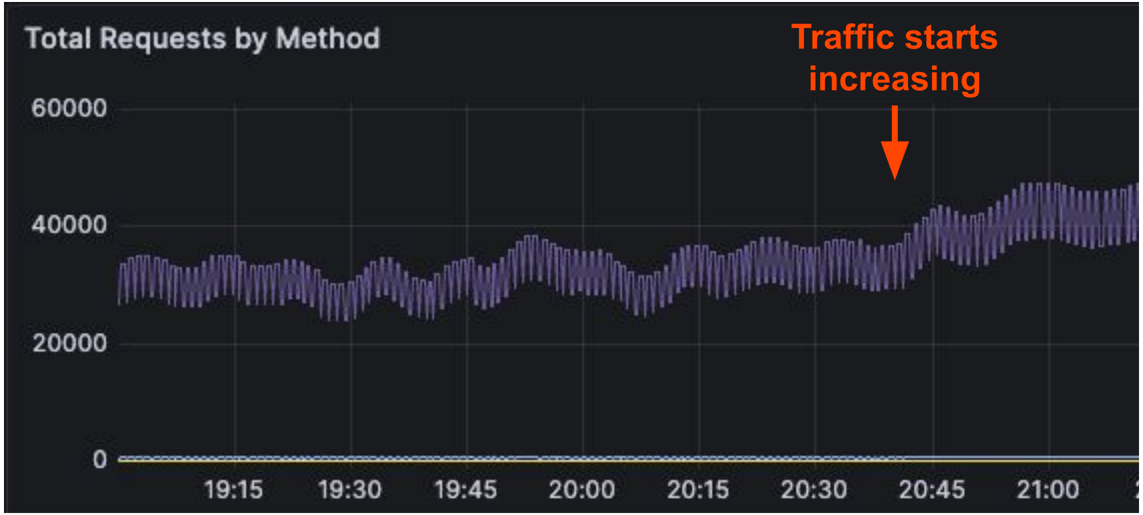
1. Retrieval (get top N docs per shard)
2. Re-rank (all N x shards docs)
   a. Features computed/queried

```
title:${keywords}
body:${keywords}
title_phrase:"{$keywords}"
```

   b. Mr. ML Model interprets features
3. Return re-ranked results

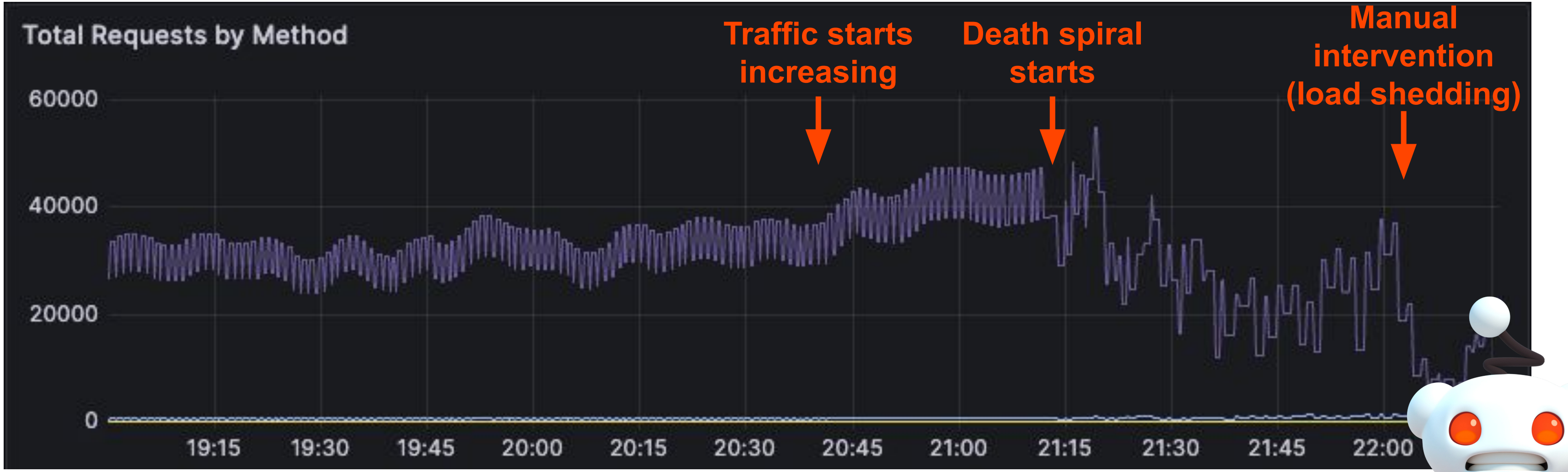


Scaling up ...



Traffic starts increasing

Scaling up … and running into failures

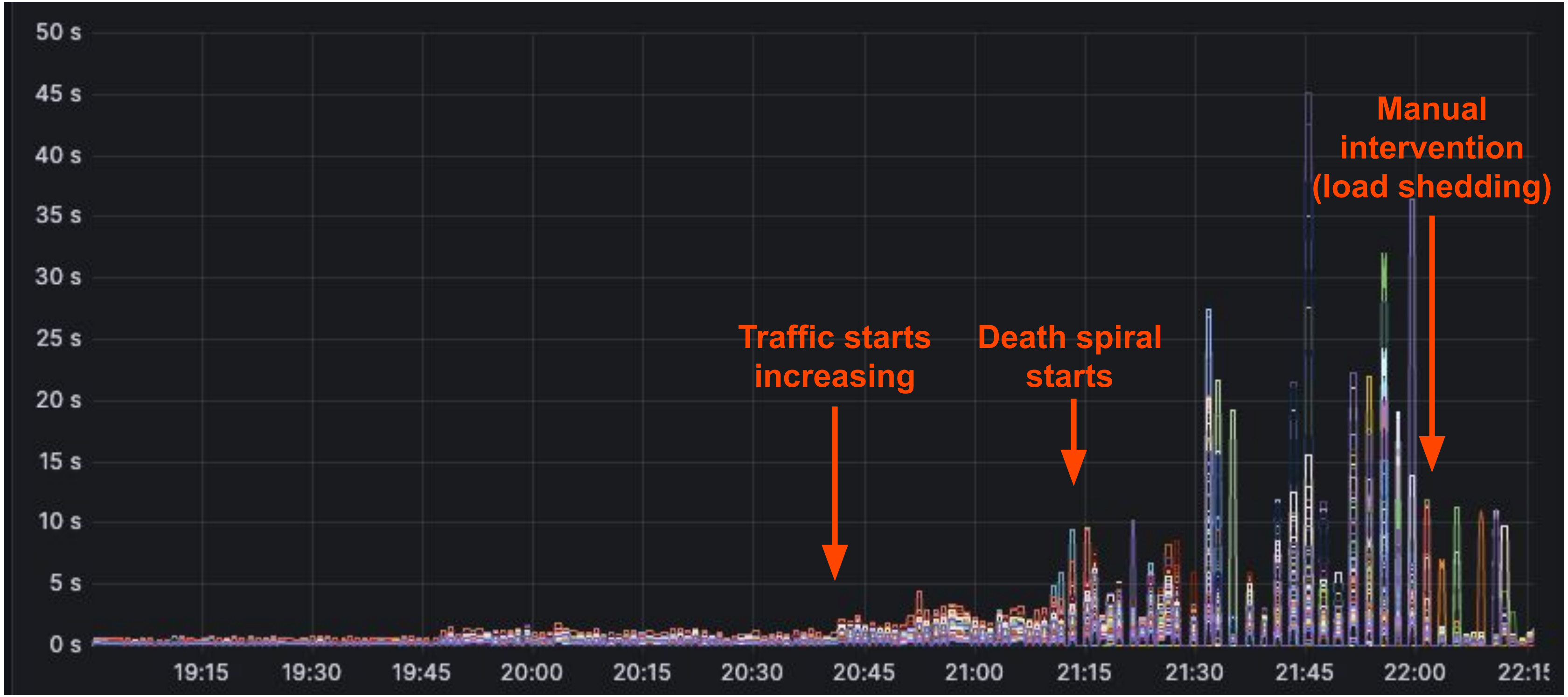


Traffic starts increasing

Death spiral starts

Manual intervention (load shedding)

Garbage Collection time spent

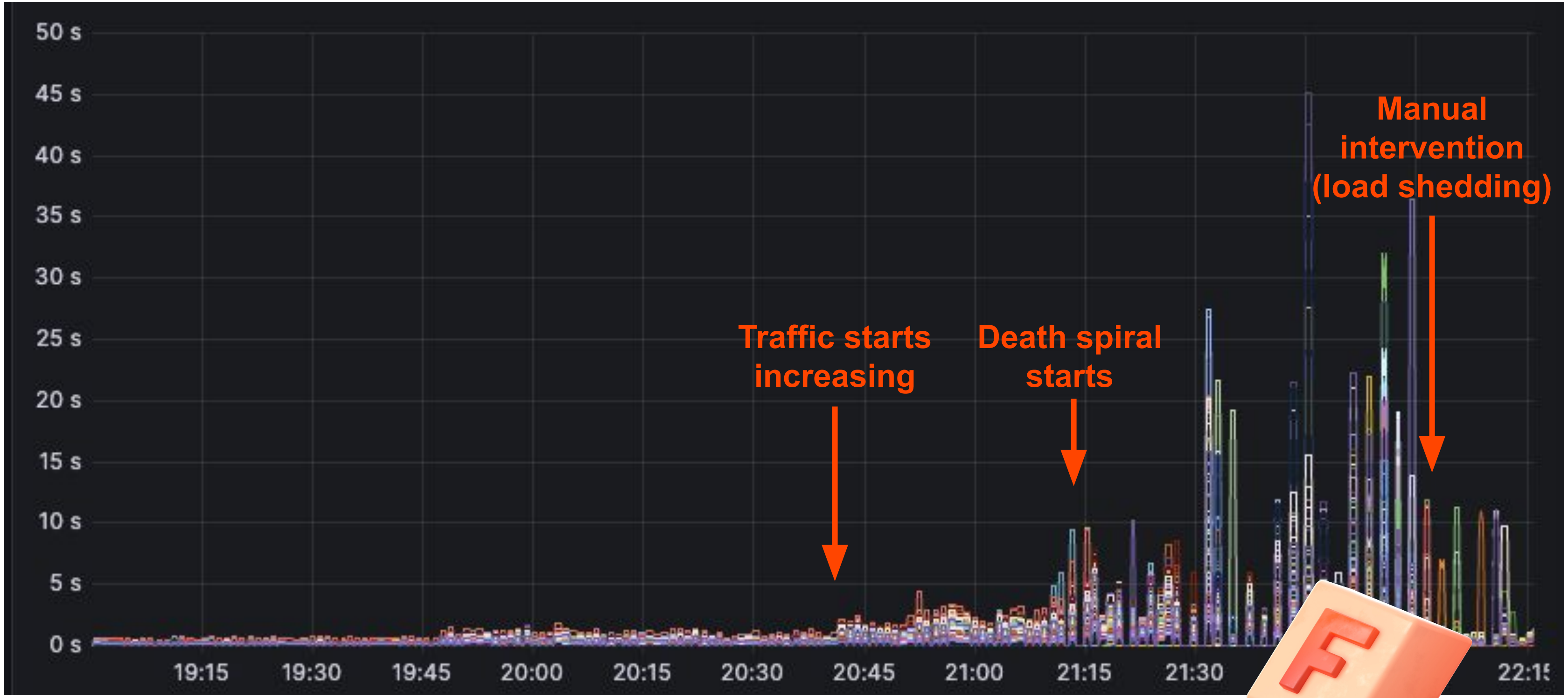


Traffic starts increasing

Death spiral starts

Manual intervention (load shedding)

Time spent in Garbage Collection

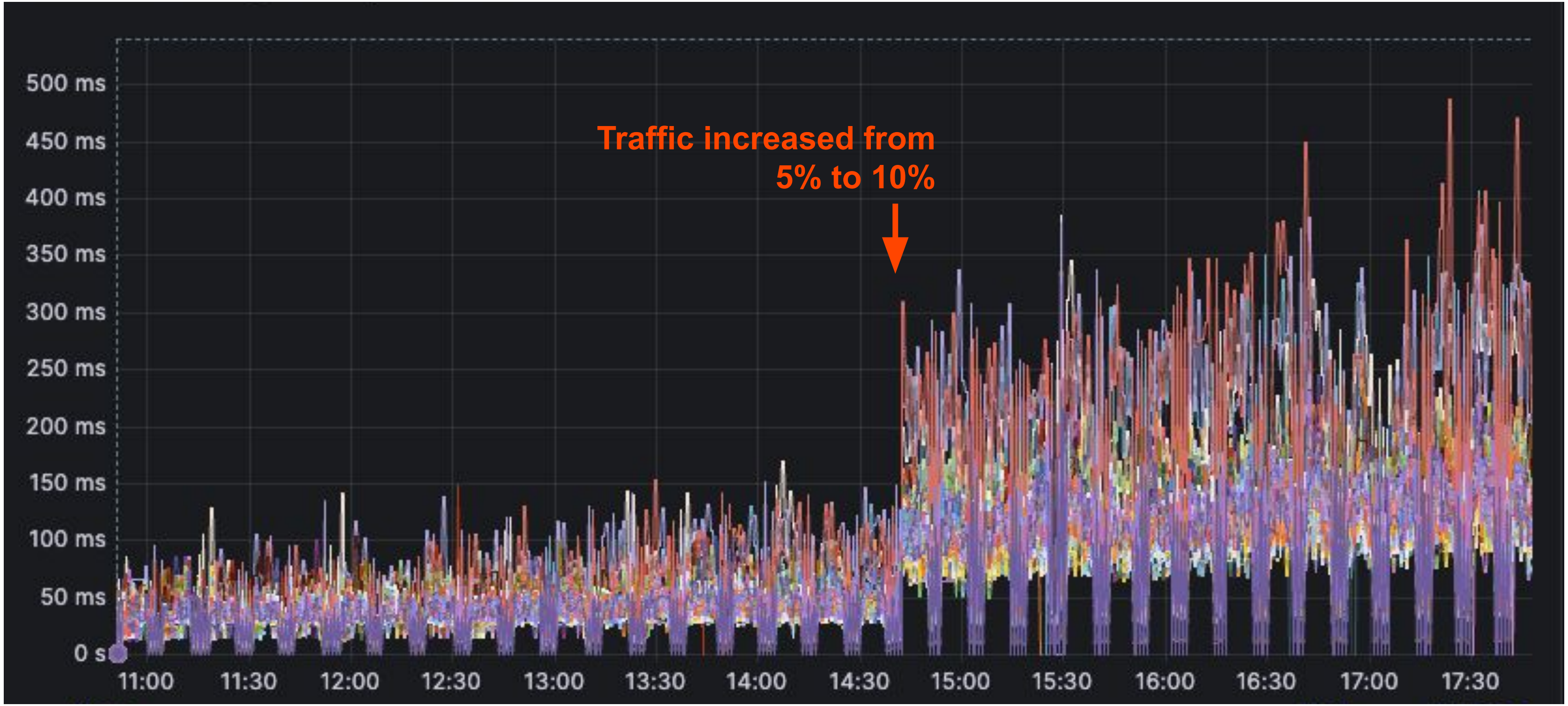


Traffic starts increasing

Death spiral starts

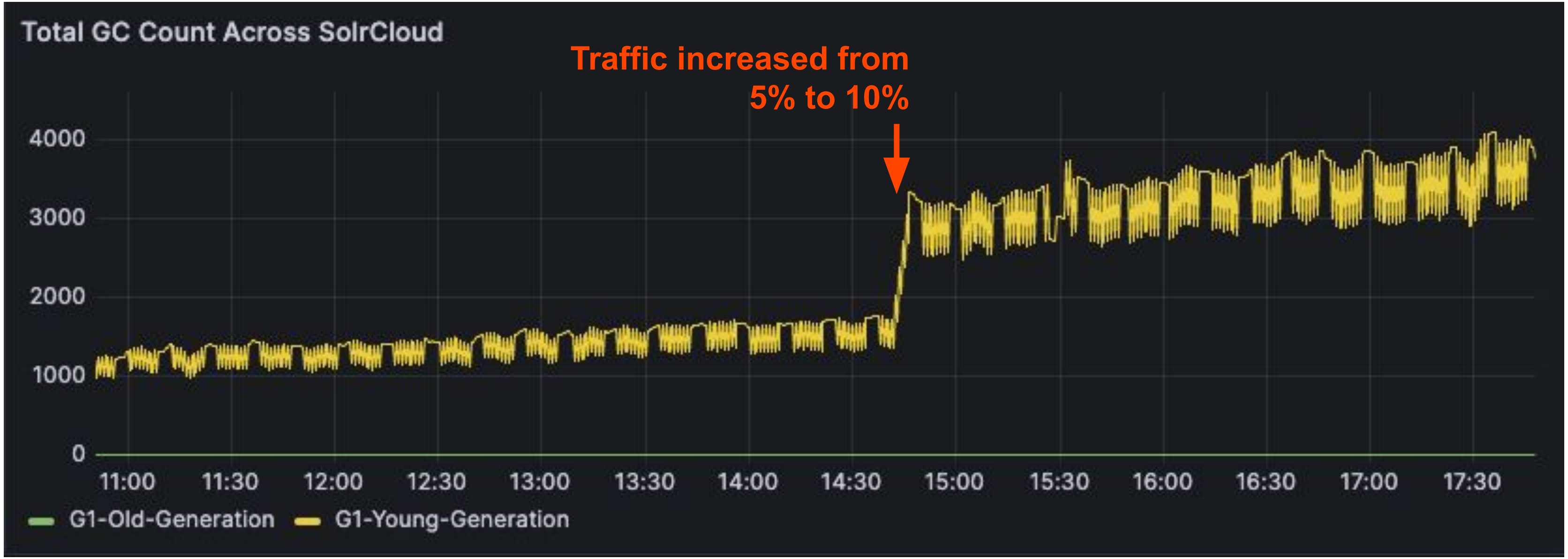
Manual intervention (load shedding)

Garbage Collection time spent (smaller jump)



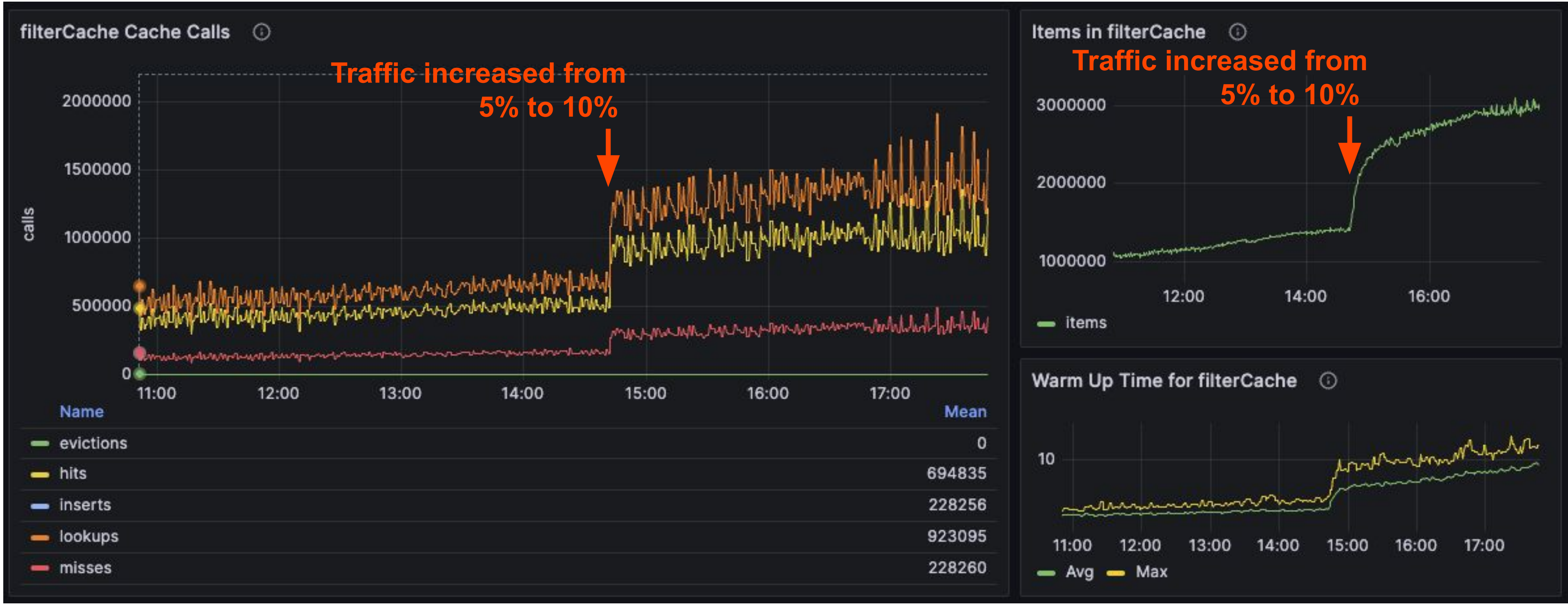
Traffic increased from 5% to 10%

Garbage Collection time spent (smaller jump)



Traffic increased from 5% to 10%

The caches look funny...



filterCache Cache Calls

Traffic increased from 5% to 10%

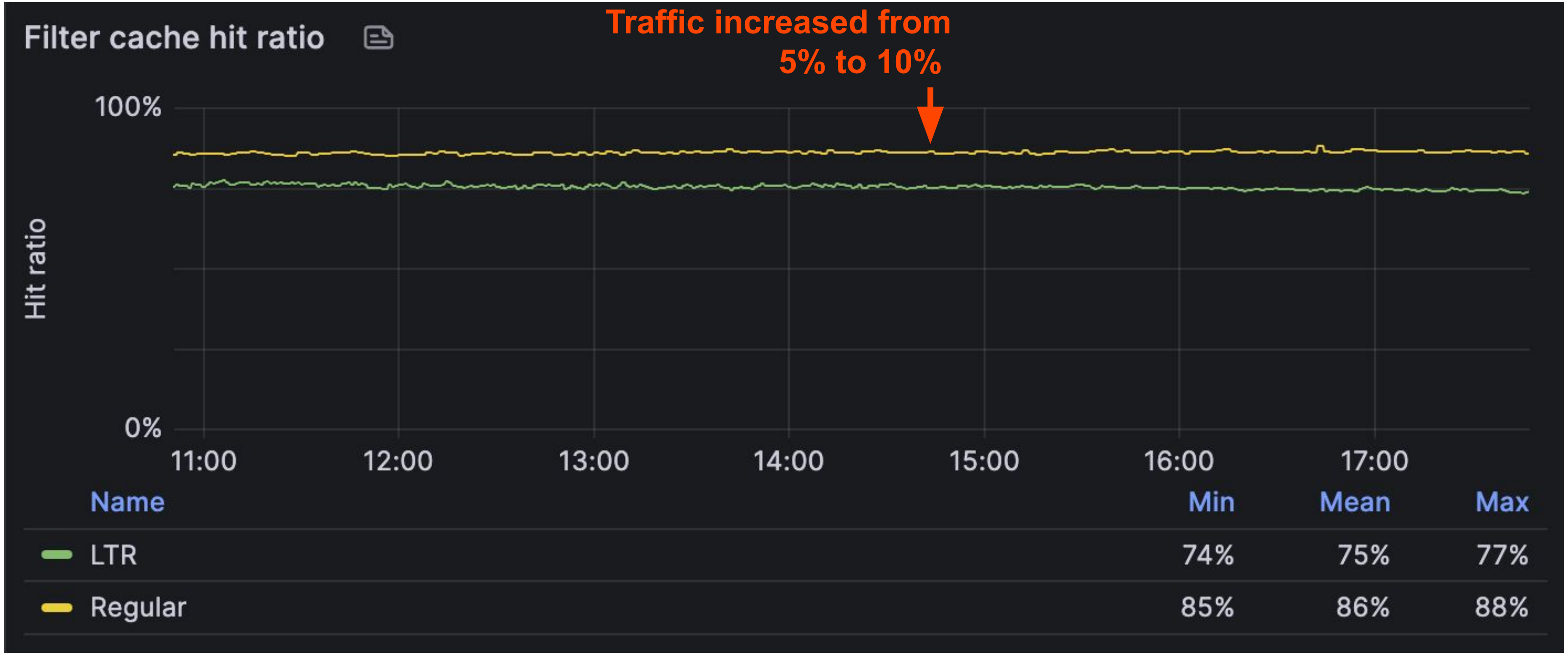
Name	Mean
evictions	0
hits	694835
inserts	228256
lookups	923095
misses	228260

Items in filterCache

Traffic increased from 5% to 10%

Warm Up Time for filterCache

The caches look funny...



Traffic increased from 5% to 10%

Name	Min	Mean	Max
🟩 LTR	74%	75%	77%
🟨 Regular	85%	86%	88%

What do our features look like? Do they cache?

```
{
  "name": "title_match_all_terms",
  "store": "LTR_TRAINING",
  "class": "org.apache.solr.ltr.feature.SolrFeature",
  "params":
  {
    "fq":
    [
      "{!edismax qf=title mm=100% v=\"${keywords}\"}"
    ]
  }
},
…
```

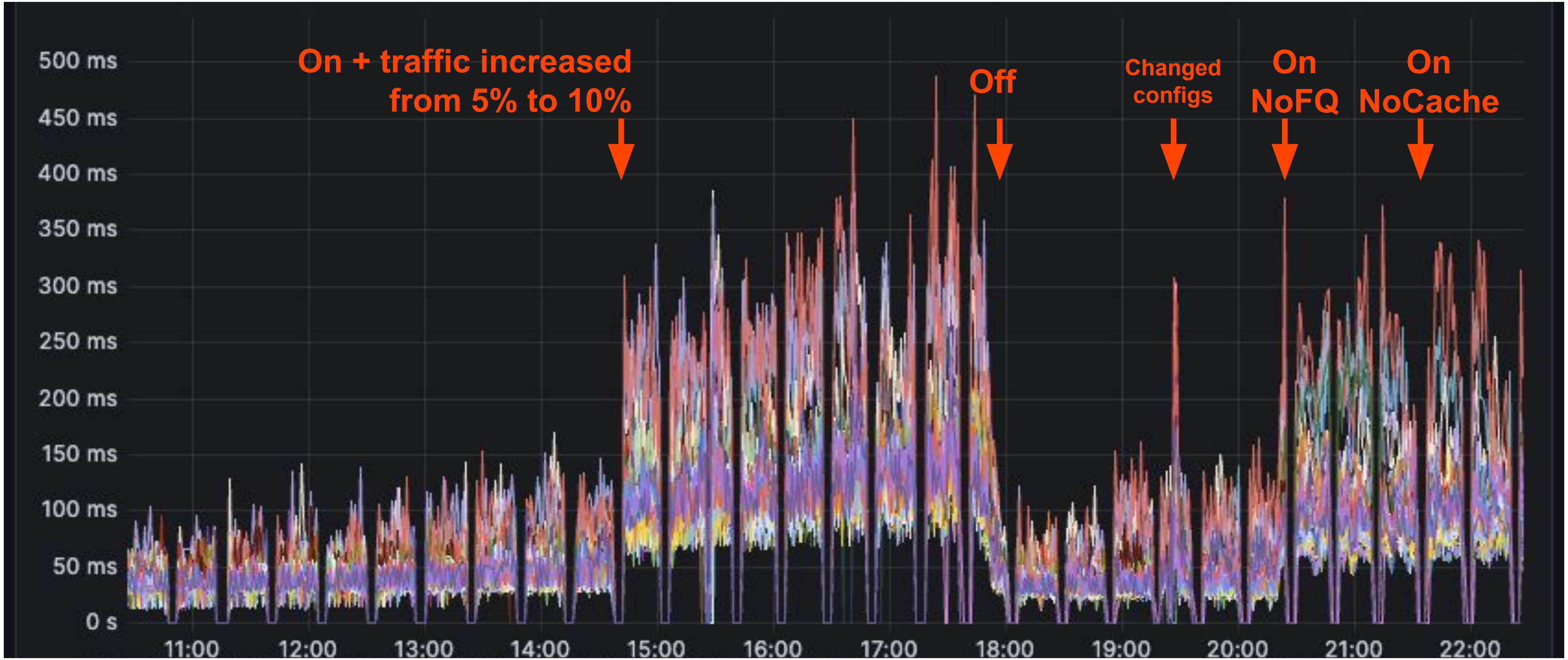
Should this be cached?

Should we set `cache=false` **?**

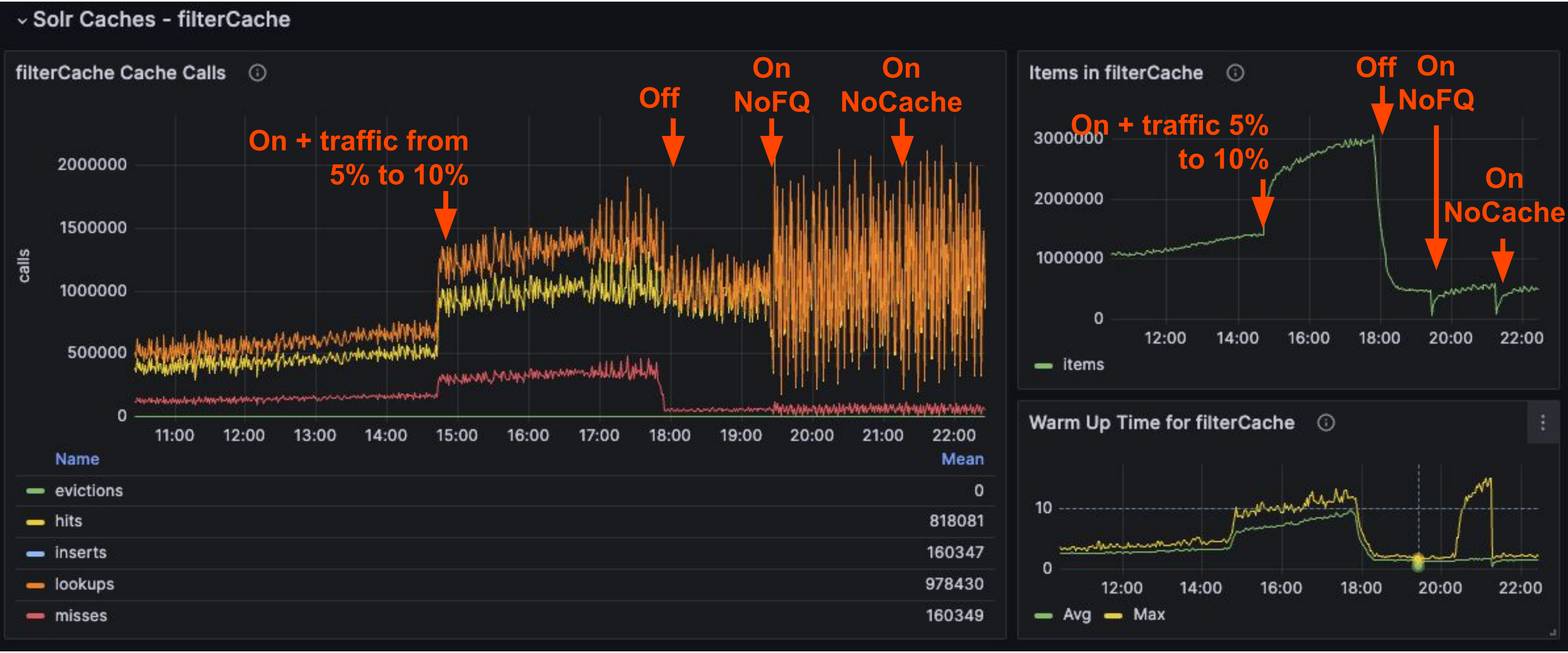
Let’s test a few configurations

On	Re-rank with no changes
Off	No re-ranking
OnNoFQ	Re-rank without FQ features
OnNoCache	Re-rank with non-cached FQ features (`cache=false`)

Garbage Collection time spent



Caching reactions



Solr Caches - filterCache

filterCache Cache Calls

On + traffic from 5% to 10%

Off

On NoFQ

On NoCache

Name	Mean
evictions	0
hits	818081
inserts	160347
lookups	978430
misses	160349

Items in filterCache

Off

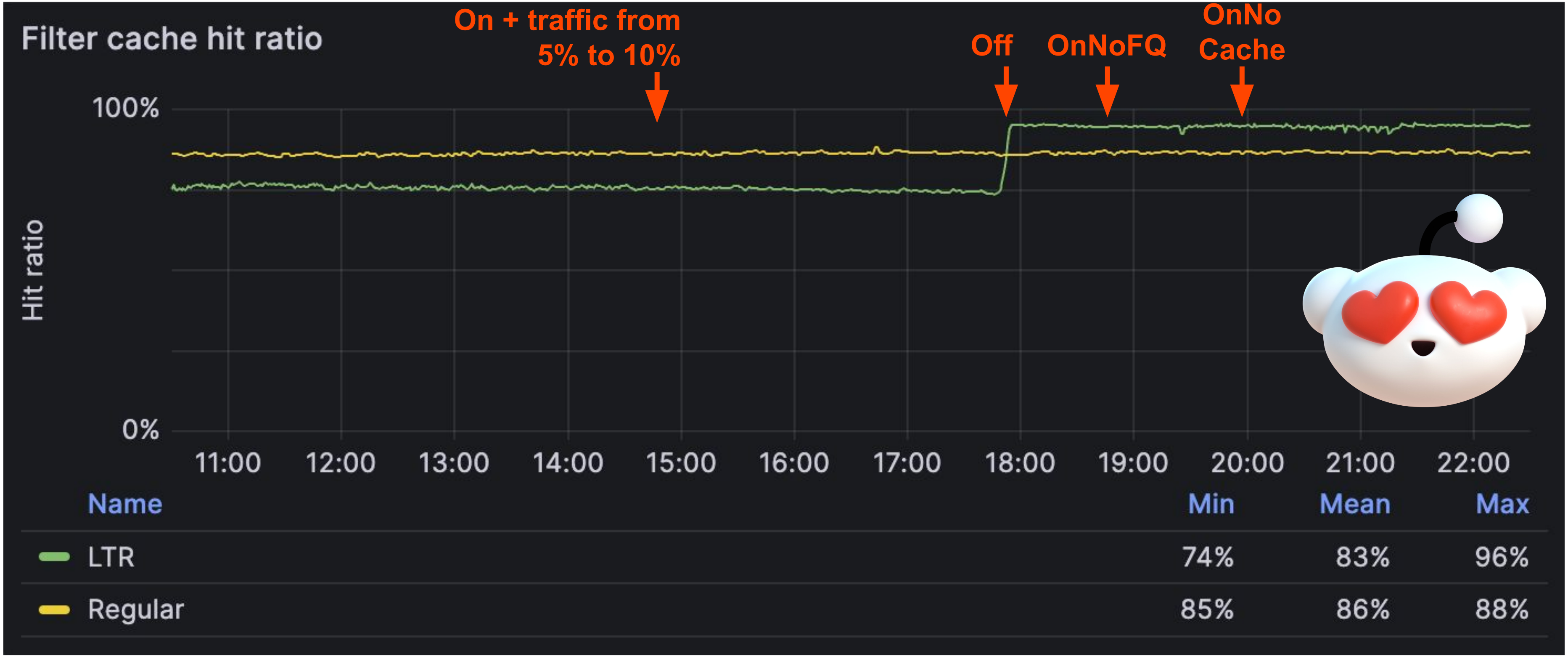
On NoFQ

On + traffic 5% to 10%

On NoCache

Warm Up Time for filterCache

Caching hit rate increased



On + traffic from 5% to 10%

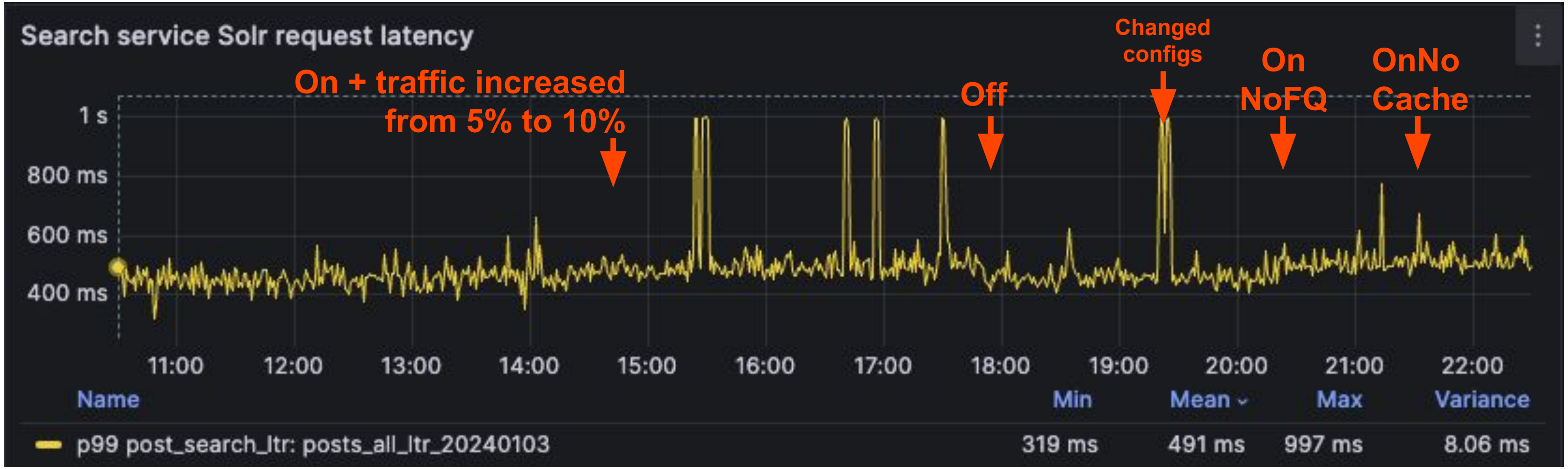
Off

OnNoFQ

OnNo Cache

Name	Min	Mean	Max
🟩 LTR	74%	83%	96%
🟨 Regular	85%	86%	88%

Latency stabilized!



Name	Min	Mean ⌄	Max	Variance
p99 post_search_ltr: posts_all_ltr_20240103	319 ms	491 ms	997 ms	8.06 ms

Tuning takeaways

GC performance is important for Solr stability

Avoiding unnecessary work to optimize performance

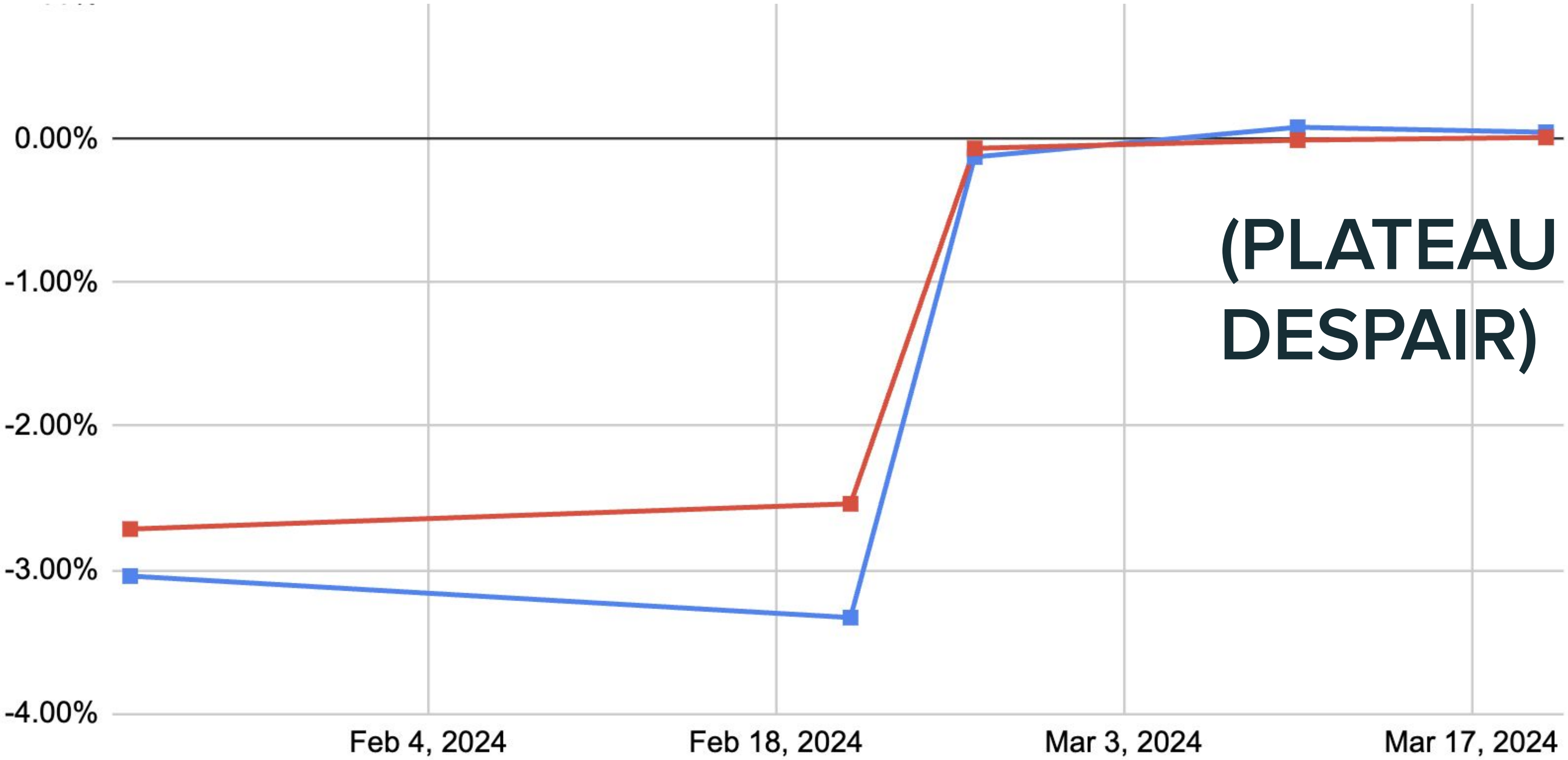
LTR features can be expensive

Yeet to to the moon!

(next steps)

All is well, until...

OFFLINE SAID THESE SHOULD WIN!



(PLATEAU OF DESPAIR)

(A/B Tests LTR vs control ...)

Revisit labels

🤓🔨 Manual relevance

- Some qualitative analysis, more human in the loop
- Weighted avg: NDCG + LGTM
- Can eyeball different types of queries and LGTM

Can be accurate ~80-90% of the time

LTR (Mr. ML Model)

- Model only as smart (or dumb) as labels
- 100% NDCG
- Examples MUST be weighted by frequency

Must be accurate 100% of the time

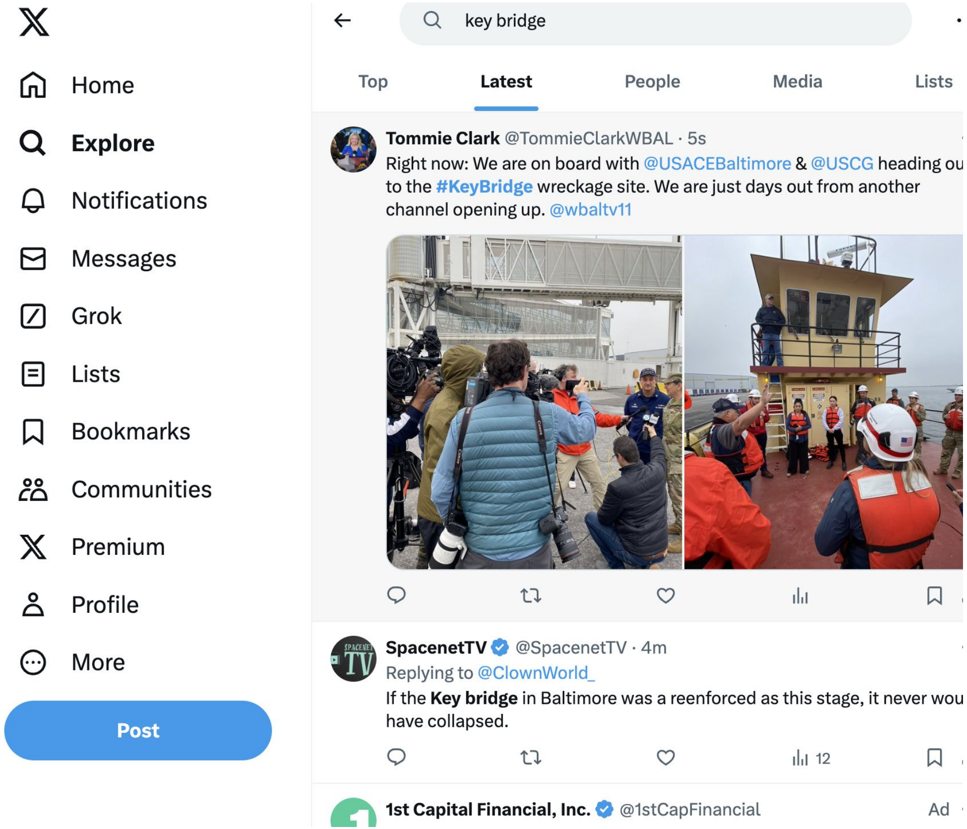
Social search problem - very very changing SERPs

Compared to e-commerce, etc

SERPS change

A LOT!

-> Aggregated labels don’t reflect actual SERPs



Tommie Clark @TommieClarkWBAL · 5s
Right now: We are on board with @USACEBaltimore & @USCG heading ou to the #KeyBridge wreckage site. We are just days out from another channel opening up. @wbaltv11

SpacenetTV @SpacenetTV · 4m
Replying to @ClownWorld_
If the **Key bridge** in Baltimore was a reenforced as this stage, it never wou have collapsed.

12

1st Capital Financial, Inc. @1stCapFinancial Ad

Currently
Human -> Analytic labels

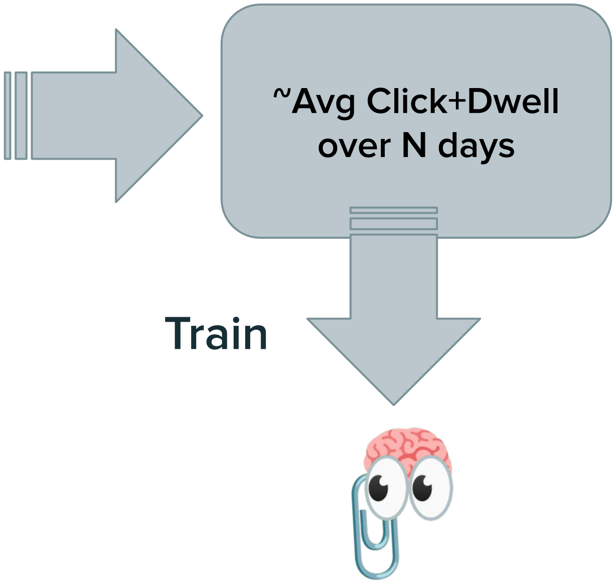
Multiple SERP analytics events

SERP ID	DATE	User Id	Query	Rank	Doc ID	Click+Dwell?
1234	2 days ago	u_124	zoolander	0	abcd	0
1234	2 days ago	u_124	zoolander	1	1212	1

•••

SERP ID	DATE	User Id	Query	Rank	Doc ID	Click+Dwell?
1251	25 days ago	u_110	zoolander	0	1211	0
1251	25 days ago	u_124	zoolander	1	12ab	1

Aggregated to:



~Avg Click+Dwell over N days

Train

Use SERP directly to train?

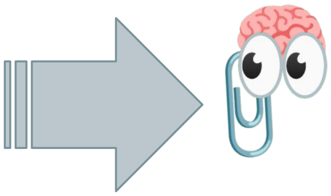
SERP ID	DATE	User Id	Query	Rank	Doc ID	Click+Dwell?
1234	2 days ago	u_124	zoolander	0	abcd	0
1234	2 days ago	u_124	zoolander	1	1212	1

SERP ID	DATE	User Id	Query	Rank	Doc ID	Click+Dwell?
1251	25 days ago	u_110	zoolander	0	1211	0
1251	25 days ago	u_110	zoolander	1	12ab	1

x 100K ? 1m?

Benefits:

- Implicitly weighted
- Handle Changing SERPs
- Features logged at point of search
- Can train on ALL context



Downsides:

- Need to feature log every search
- A lot more data!

Feature Eng - Signals

Trending / recent posts that get engagement for a query

query	post	boost
ace ventura	6785	1.2
zoolander	1234	1.5
zoolander	5678	1.1

Pros / Cons Signals vs an LTR model

🫣 <u>Signals:</u>

“OVERFIT” - not generalized, but a great cheat-sheet for ‘right answer’, but only for queries seen in past

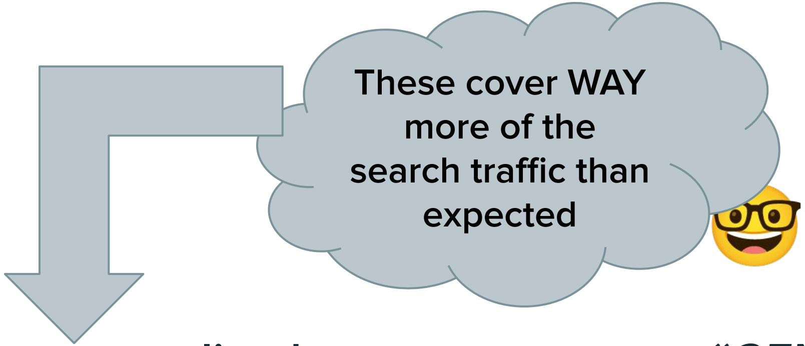
Good for fast changing head queries

🤓 - <u>Model:</u>

“GENERALIZED” - not overfit, general “pattern” can work with query seen rarely / never

Good for torso+tail / not as engaging queries

Signals cover A LOT of the search traffic



These cover WAY more of the search traffic than expected

🫣 **Signals:**

“OVERFIT” - not generalized, but a great cheat-sheet for ‘right answer’, but only for queries seen in past

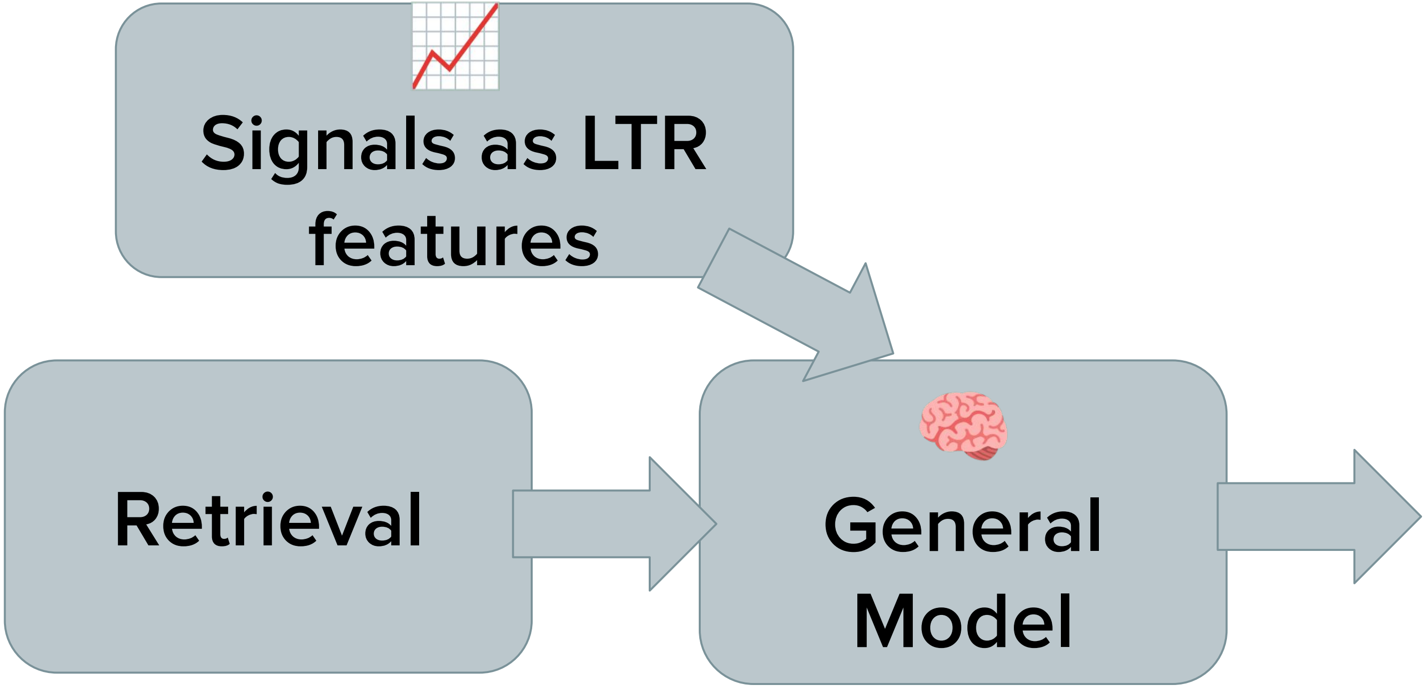
Good for fast changing head queries

🤓 - **Model:**

“GENERALIZED” - not overfit, general “pattern” can work with query seen rarely / never

Good for torso+tail / not as engaging queries

Need to add these to our model



📈 Signals as LTR features

Retrieval

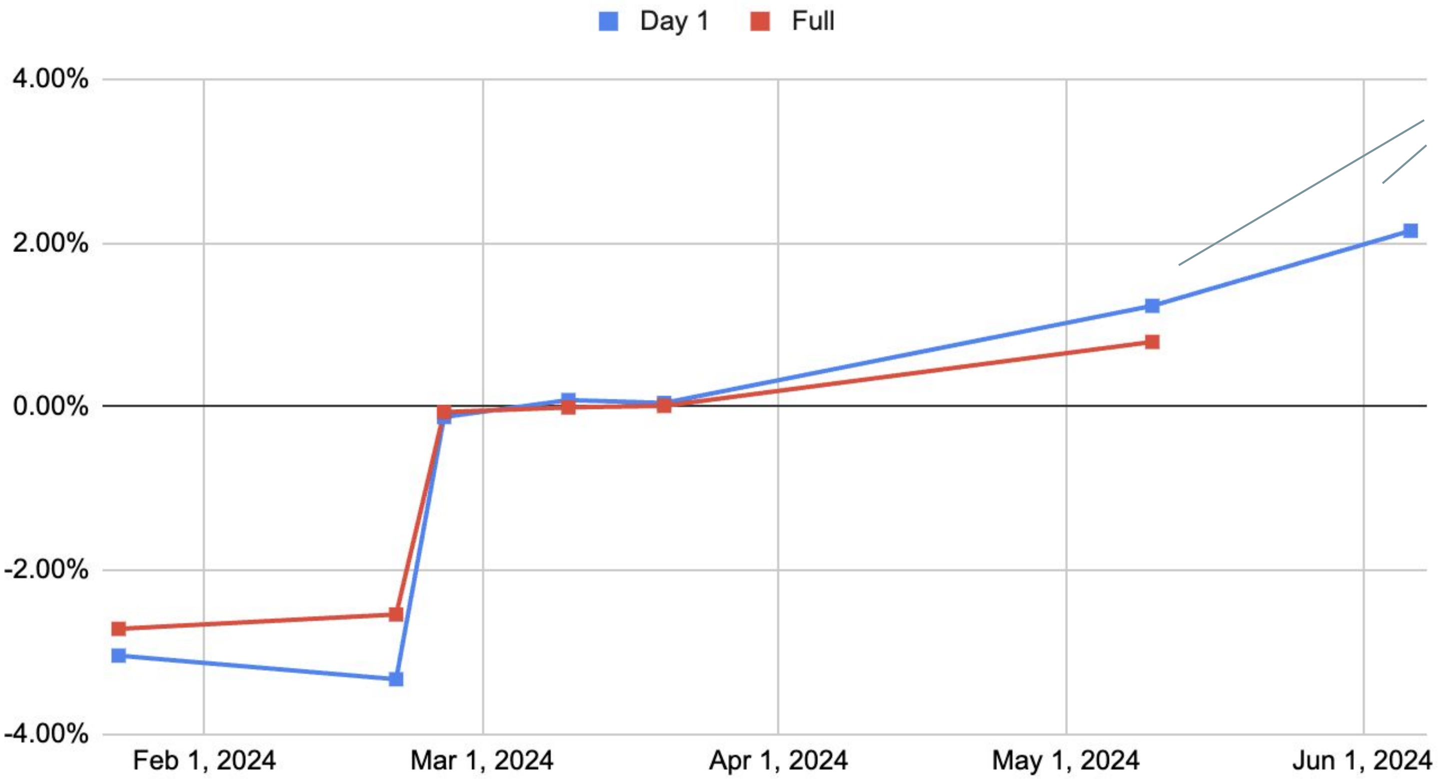
🧠 General Model

Ideal mix of signal + lexical

(simple features / ranking)

(complex features / ranking)

Update!!! Hot off the presses



Variants w / signals adde

Thank you

reddit