We built the Elasticsearch LTR Plugin!
...then came the hard part...

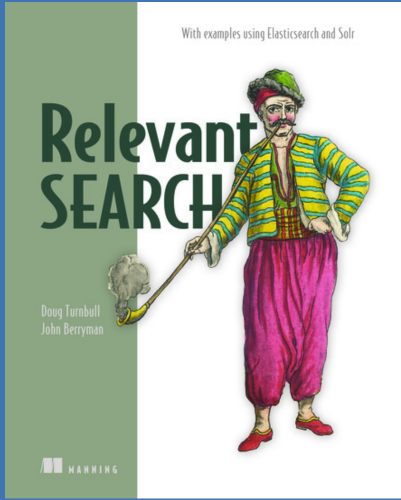
Jason Kowalewski
Sr. Director, Engineering
Snagajob
jason.kowalewski@snagajob.com

Doug Turnbull
Chief Consultant
OpenSource Connections
dturnbull@o19s.com
@softwaredoug

OpenSource Connections

- Solr & Elasticsearch Relevance Consultants

- Specialized Search, Recommendations & Information Retrieval



4.5 million monthly job applications

ONE million new workers added to the site each month

75+ million registered hourly workers

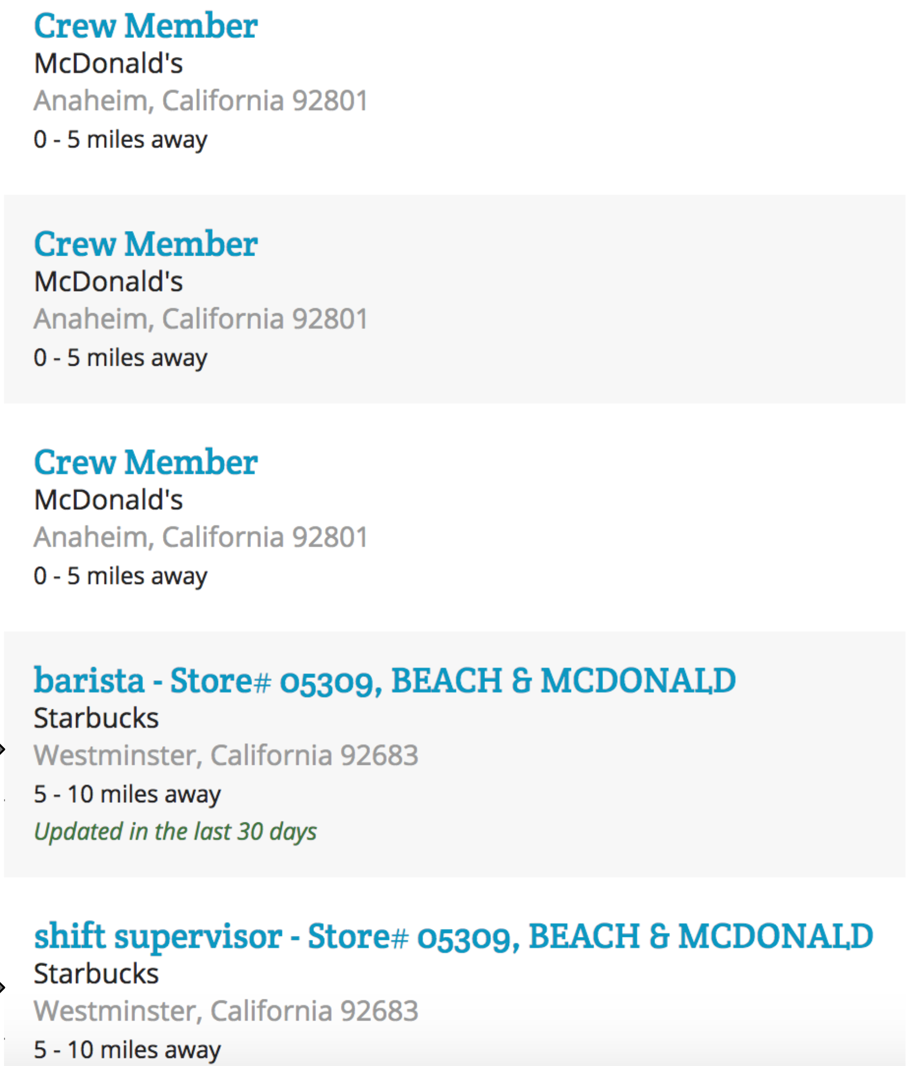
LARGEST
marketplace for hourly work

5.3 million people hired last year

300,000 employer locations

snagajob ENGINEERING

Query: “mcdonalds in 92801”



Title Boost,
“Freshness” Boost

Query: "mcdonalds in 90024"

barista - Store# 05309, BEACH & MCDONALD
Starbucks
Westminster, California 92683
15 - 20 miles away
Updated in the last 30 days

shift supervisor - Store# 05309, BEACH & MCDONALD
Starbucks
Westminster, California 92683
15 - 20 miles away
Updated in the last 30 days

Crew Team Member
McDonald's
CARSON, California 90746
0 - 5 miles away

Shift Manager
McDonald's
CARSON, California 90746
0 - 5 miles away

Department Manager
McDonald's
CARSON, California 90746
0 - 5 miles away

Query: “mcdonalds in 11231”

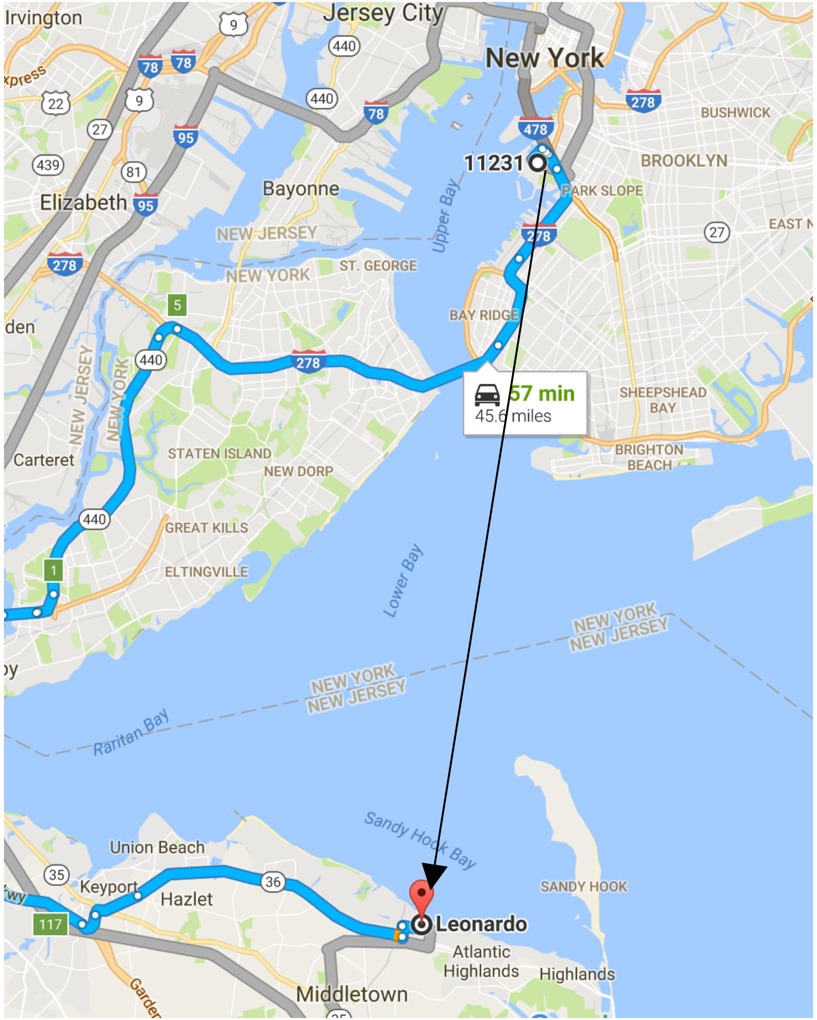
Crew
McDonald's Franchisee
BROOKLYN, New York 11231
0 - 5 miles away

Maintenance Person
McDonald's Franchisee
BROOKLYN, New York 11231
0 - 5 miles away

Crew Member Day Shifts
McDonald's
Leonardo, New Jersey 07716
15 - 20 miles away

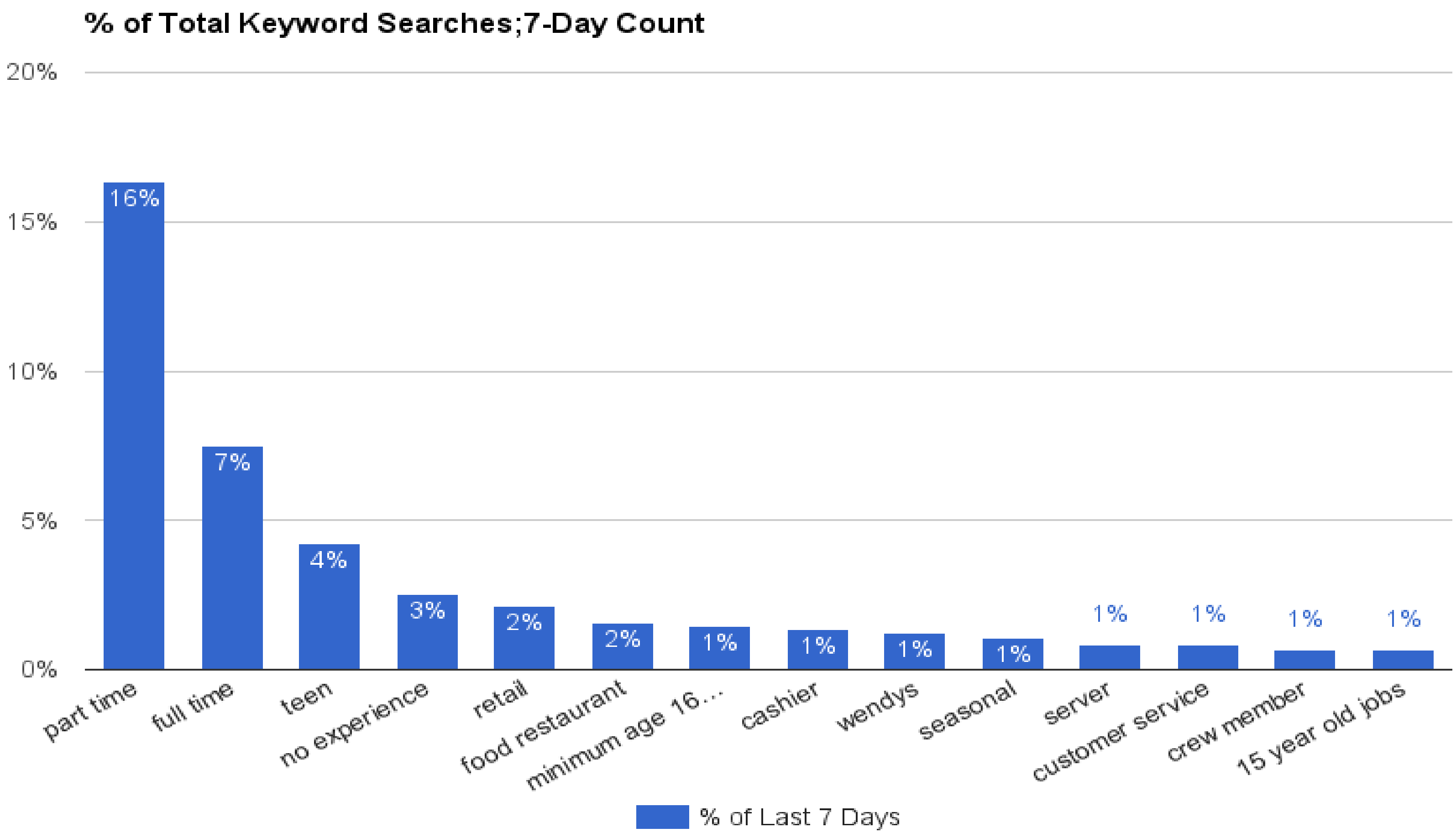
Crew Member Closing Shift
McDonald's
Leonardo, New Jersey 07716
15 - 20 miles away

Shift Manager
McDonald's
Leonardo, New Jersey 07716
15 - 20 miles away



Jersey City
New York
Irvington
Elizabeth
Bayonne
BROOKLYN
BUSHWICK
PARK SLOPE
11231
NEW JERSEY
NEW YORK
ST. GEORGE
Upper Bay
BAY RIDGE
57 min
45.6 miles
SHEEPSHEAD BAY
BRIGHTON BEACH
STATEN ISLAND
NEW DORP
GREAT KILLS
ELTINGVILLE
Carteret
Lower Bay
NEW YORK
NEW JERSEY
Raritan Bay
Sandy Hook Bay
SANDY HOOK
Union Beach
Keyport
Hazlet
Leonardo
Atlantic Highlands
Highlands
Middletown

Our Workers don't know what they want.



Query: “part time in 11231”

Babysitter Needed For 2 Children In Brooklyn
Care.com
Brooklyn, New York 11231
0 - 5 miles away
Updated today

Babysitter Needed To Pick Up 3 Year Old From Summer Camp
Care.com
Brooklyn, New York 11231
0 - 5 miles away
Updated yesterday

Nanny Needed For 2 Children In Brooklyn
Care.com
Brooklyn, New York 11231
0 - 5 miles away
Updated yesterday

Registered Nurse
Sunrise Senior Living
Brooklyn, New York 11201
0 - 5 miles away
Updated in the last 2 weeks

Entry Level Tax Preparers
Liberty Tax Service
Brooklyn Park, New York 11231
0 - 5 miles away

The Real Problem

Hand-tuned relevance does not work for us.

- Field boosts are complex, and difficult to maintain.
- Users are not often searching with a precise keyword set, yet job decisions are very personal.
- Geography plays an important factor in relevance
- Humans are complicated! Relevance factors are non-linear!

snagajob
ENGINEERING

Solution: Learning to Rank

Let our data drive the ranking of an optimal combination of jobs for our workers.

snagajob ENGINEERING

Solution: Learning to Rank

snagajob + =

Elasticsearch
LTR Plugin v. 0.1

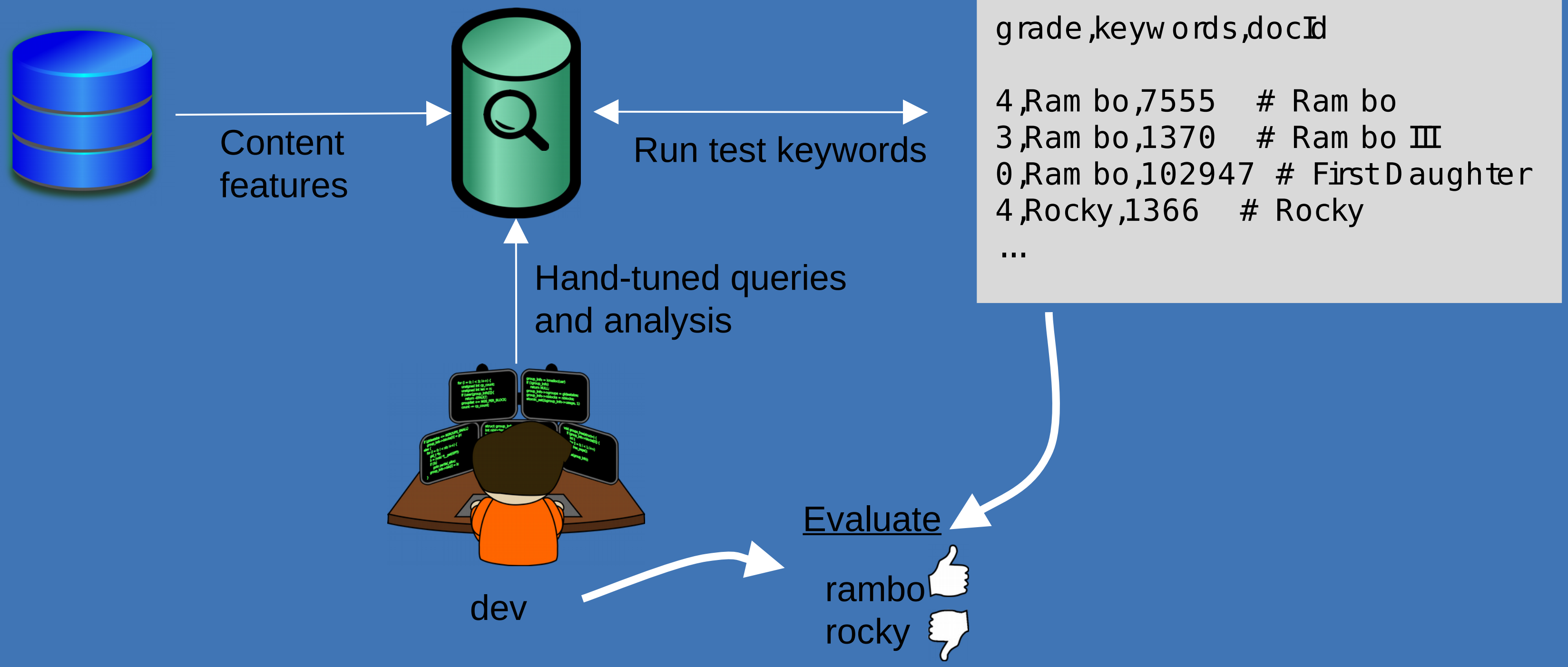
Start w/ Judgment Lists

Quality of doc per query

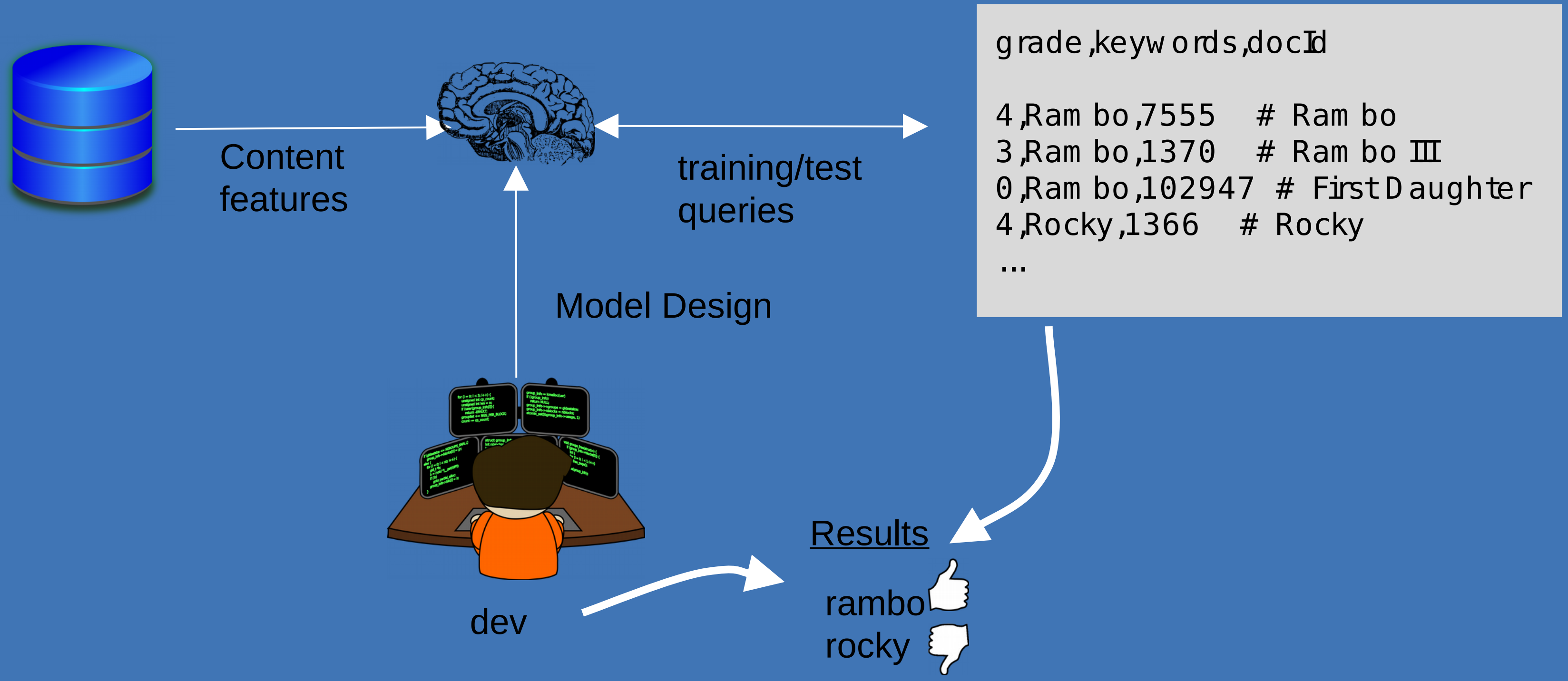
```
grade,keywords,docId

4,Rambo,7555   # Rambo
3,Rambo,1370   # Rambo III
0,Rambo,102947 # First Daughter
4,Rocky,1366   # Rocky
…
```

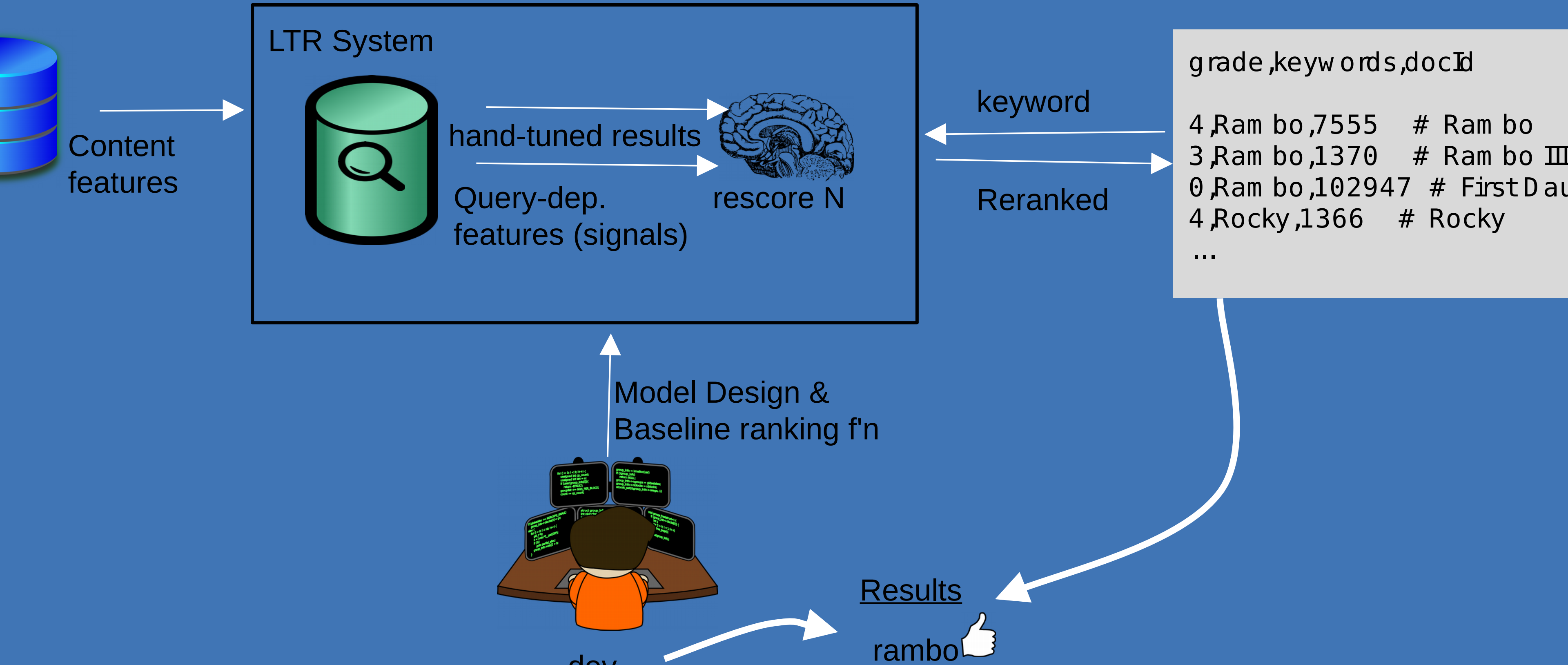
Classic Relevance Tuning



Learning to Rank



Learning to Rank + ES



Judgment List -> Set

```
grade,keywords,docId

4,Rambo,7555 # Rambo
3,Rambo,1370 # Rambo III
0,Rambo,102947 # First Daughter
4,Rocky,1366 # Rocky
…
```

```
grade,queryId,titleScore,bodyScore

4 qid:1 1:0.5 2:24.4
3 qid:1 1:0.76 2:12
0 qid:1 1:10 2:947
4 qid:2 1:4 2:59
…
```

Features: Logged ES Query Scores

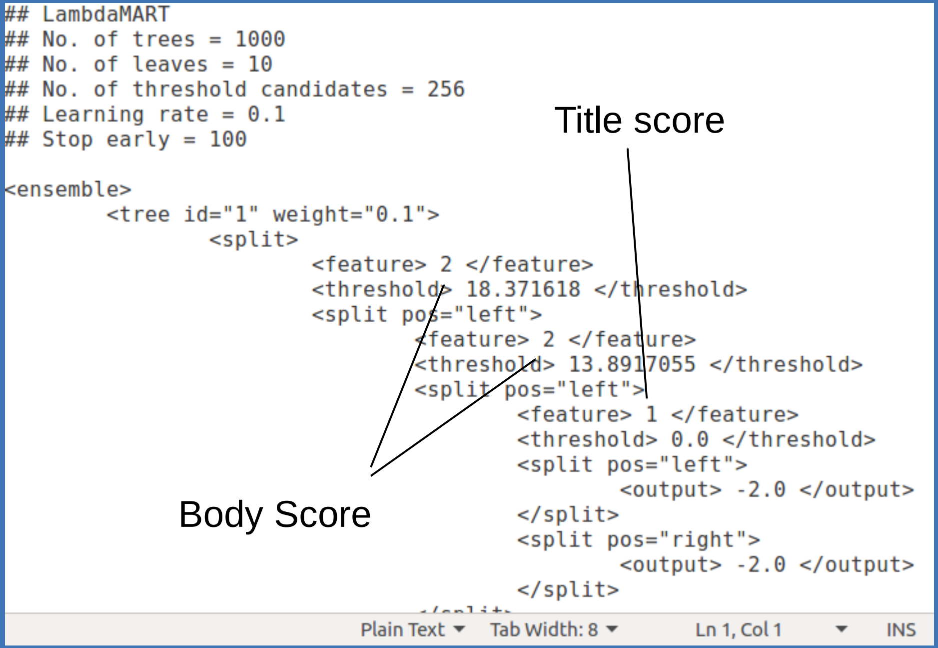
```
{
  "query": {
    "match": {
      "title": "<<keyword>>"
    }
  }
}
```

Training Set -> Model

,titleScore,bodyScore

2:24.4
2:12
2:947
:59

Ranklib or other tool



```
## LambdaMART
## No. of trees = 1000
## No. of leaves = 10
## No. of threshold candidates = 256
## Learning rate = 0.1
## Stop early = 100

<ensemble>
        <tree id="1" weight="0.1">
                <split>
                        <feature> 2 </feature>
                        <threshold> 18.371618 </threshold>
                        <split pos="left">
                                <feature> 2 </feature>
                                <threshold> 13.8917055 </threshold>
                                <split pos="left">
                                        <feature> 1 </feature>
                                        <threshold> 0.0 </threshold>
                                        <split pos="left">
                                                <output> -2.0 </output>
                                        </split>
                                        <split pos="right">
                                                <output> -2.0 </output>
                                        </split>
```


Features: Logged ES Query Scores

```
{
  "query":{
    "match":{
      "title":"<<keyword>>"
    }
  }
}
```

Model -> Elasticsearch

```
= 256

1">

e> 2 </feature>
old> 18.371618 </threshold>
pos="left">
  <feature> 2 </feature>
  <threshold> 13.8917055 </threshold>
  <split pos="left">
          <feature> 1 </feature>
          <threshold> 0.0 </threshold>
          <split pos="left">
                  <output> -2.0 </output>
          </split>
          <split pos="right">
                  <output> -2.0 </output>
          </split>
```


Plugin Functionality 1: "ranklib" scripting language for specifying LTR models:

```
POST _scripts/ranklib/dougs_model
{
  "script": "## LambdaMART\n## No. of trees = 1\n## No. of leaves = 10\n## No. of threshold candidates = 256\n## Learning rate = 0.1\n## Stop early = 100\n\n<ensemble>\n <tree id=\"1\" weight=\"0.1\">\n  <split>\n <feature> 1 </feature>\n   <threshold> 0.45867884 </threshold>\n   <split pos=\"left\">\n    <feature> 1 </feature>\n <threshold> 0.0 </threshold>\n    <split pos=\"left\">\n     <output> -2.0 </output>\n
….
```

Query w/ Model

Plugin Functionality 1: "ranklib" scripting language for specifying LTR models:

```
POST _scripts/ranklib/dougs_model
{
 "script": "## LambdaMART\n## No. of trees = 1\n## No. of leaves = 10\n## No. of threshold candidates = 256\n## Learning rate = 0.1\n## Stop early = 100\n\n<ensemble>\n <tree id=\"1\" weight=\"0.1\">\n  <split>\n   <feature> 1 </feature>\n   <threshold> 0.45867884 </threshold>\n   <split pos=\"left\">\n    <feature> 1 </feature>\n    <threshold> 0.0 </threshold>\n    <split pos=\"left\">\n     <output> -2.0 </output>\n     ….
```

Plugin Functionality 2: "ltr" query that executes a model

```
POST tmdb/movie/_search
{
  "query": {
    "ltr": {
      "model": {
        "stored": "dougs_model"
      },
      "features": [
        {
          "match": {
            "title": "rocky"
          }
        },
        {
          "multi_match": {
            "query": "rocky",
```

Model we just stored

Feature "1" in training data

Feature "2" in training data

You ought to rescore…

unctionality 2: "ltr" query that
s a model

```
/movie/_search

": {
r": {
"model": {
  "stored": "dougs_model"
},
"features": [
  {
    "match": {
      "title": "rocky"
    }
  },
  {
    "multi_match": {
      "query": "rocky",
```

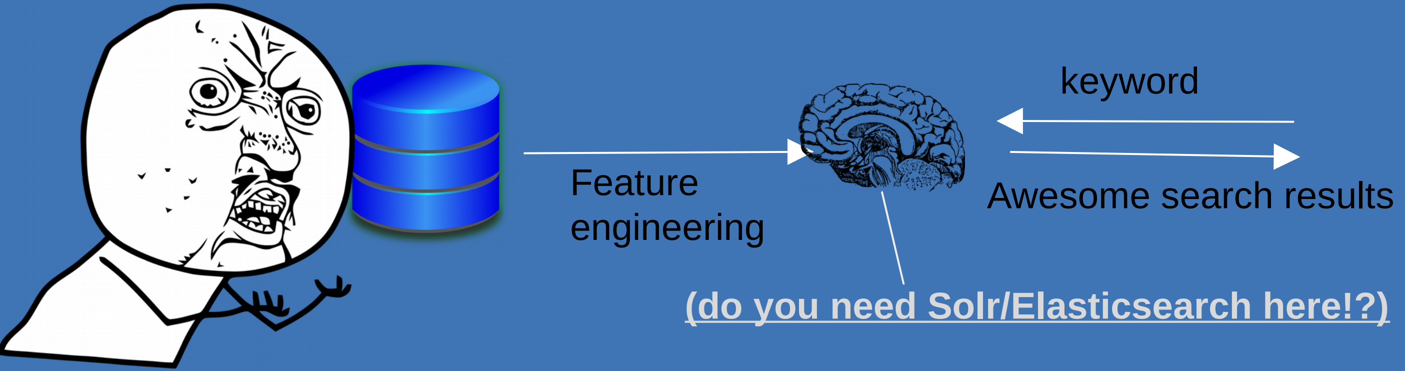
```
POST tmdb/movie/_search
{
  "query": {
    "match": {
      "_all": "rocky"
    }
  },
  "rescore": {
    "window_size": 500,
    "query": {
      "rescore_query": {
        "ltr": {
          "model": {
            "stored": "dougs_model"
          },
          "features": [
```

Baseline query

Rescore top 500

Same ltr query

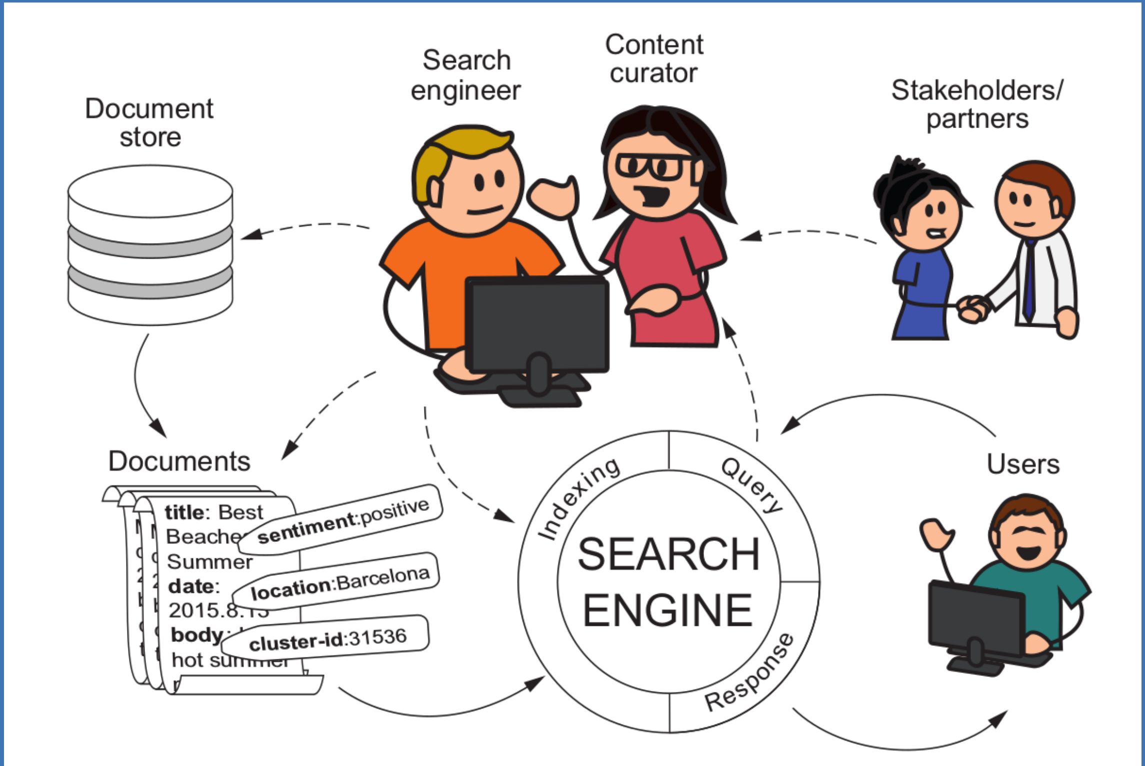
Y u no just use model?!?



Problems w/ just model:

- **Performance**: search engines can prefilter what models score
- **Query-dependent features:** many features depend on keyword (i.e TF*IDF on certain fields) ~ "signals"
- **Business Rules:** influencing ranking beyond user relevance
- **Functionality:** facets, paging, grouping, spell checking, autocomplete, etc etc...

Lesson 1: Judgments == Hard!



Who defines this?

- Domain Experts?
- User analytics?
- Testing w/ Users?
- Devs?
- Sue in marketing?
- HiPPO (highest paid person's opinion?)

Do all these people agree!?

(really go buy my book because this is the *real* hard stuff)

No one size fits all

Consumer-facing

Knowledge-Mgmt

Interpret Analytics (clicks, conversions, etc)

Interpret User Testing (classic judgment lists)

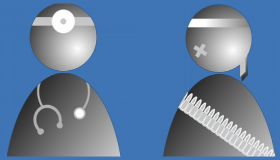
<u>Challenges</u>

Less depth into "why" behind keywords

Poor Info Need differentiation

Cost: Infrastructure/code for analytics

<u>Takeaway:</u> "Interpret" in **both** cases takes domain expertise



<u>Challenges</u>

Less data, low Statistical significance

Complex info needs

Cost: time consuming, paying experts, experts don't have time

Lesson 2: Grade Consistency

Don't make judgment grades keyword relative!

OpenSource Connections Blog Example

- 4 -- article written on keyword topic within last year
- 3 -- article written on topic within last 5 years OR adjacent topic in last year
- 2 -- adjacent topic more than a year old
- 1 -- not relevant
- 0 -- opposite meaning of keywords

OSC's blog has at best a "2" for the keyword "enterprise service bus" (because we have some old Apache Camel articles)

Notice how a global sense of what a "4" is means easier to perform regression to predict the "4s" from many examples

Lesson 3:What should we optimize for?

Precision@n - good stuff near top! — No position bias

NDCG@n - *close* to your best results — Your best might stink!

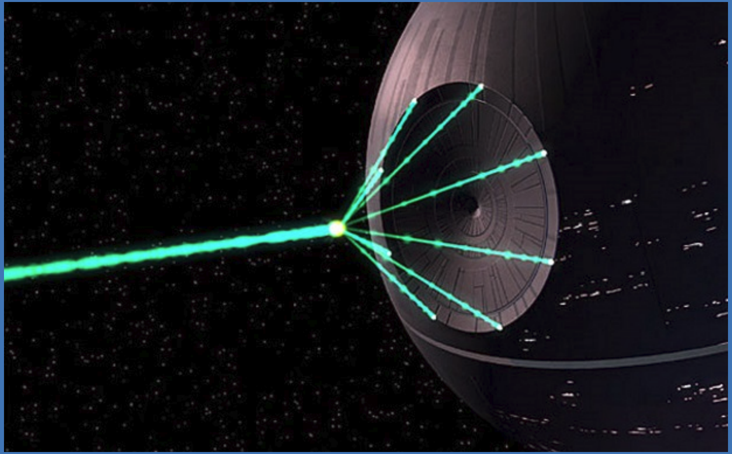
ERR@n - *trust perception* scanning results — Users don't realize there's better out there

Which should we optimize for?

Yes.

Lesson 4: Accuracy vs Speed

When your training infra is...



(big lumbering pile of dedicated compute)

But your search infra is handling….

(less time per 'search requests')

Lesson 5: Model Selection?

Matters *less* than you think

Linear Models: (aka optimizing "boosts") simple use cases, doesn't get "nuance"

Gradient Boosting/SVM/Random Forest: personal experience/preference how much you can understand/debug the model?

Generally: Garbage In/Garbage Out!

Multiple Models?

Prec@500 NDCG@50 ERR@10

Simple/Fast Model:
Crowd top 500 w/
'good stuff'

Optimize best first

First page trustworthy
very accurate but
slow

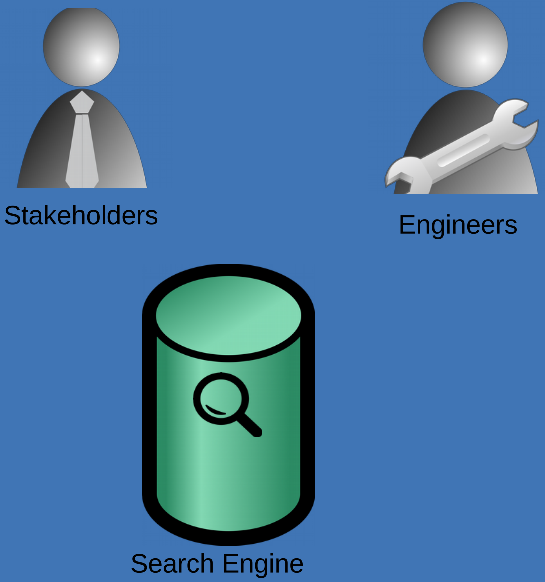
Lesson 6: Quality/Accuracy

- Separate test and training data
- With complex models: I often use simple *best subset selection* on small number of features
- Tree-based systems, often a *mix* of features offers context, so look for best performing mix
- Which combinations features perform best? Did changing features change relative performance offline on test data?

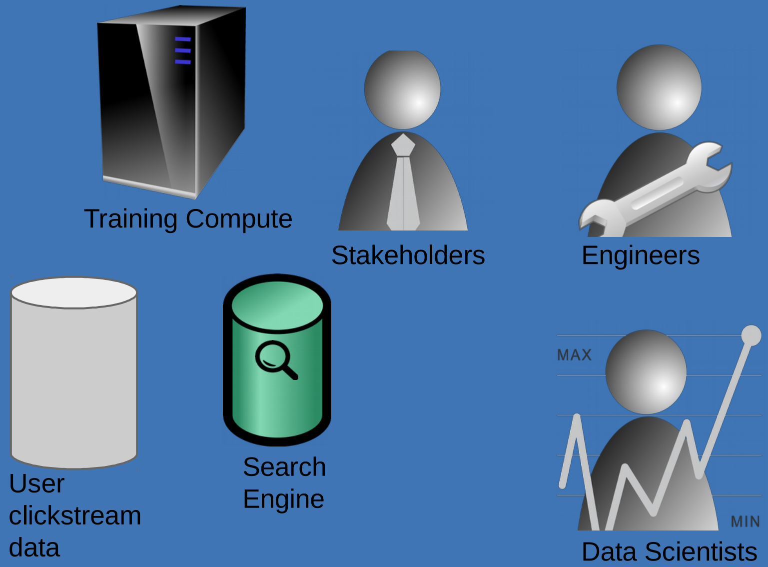
Every machine learning problem can be solved by adding one more for loop!
(or Spark map job)

Lesson 7: This is *harder*

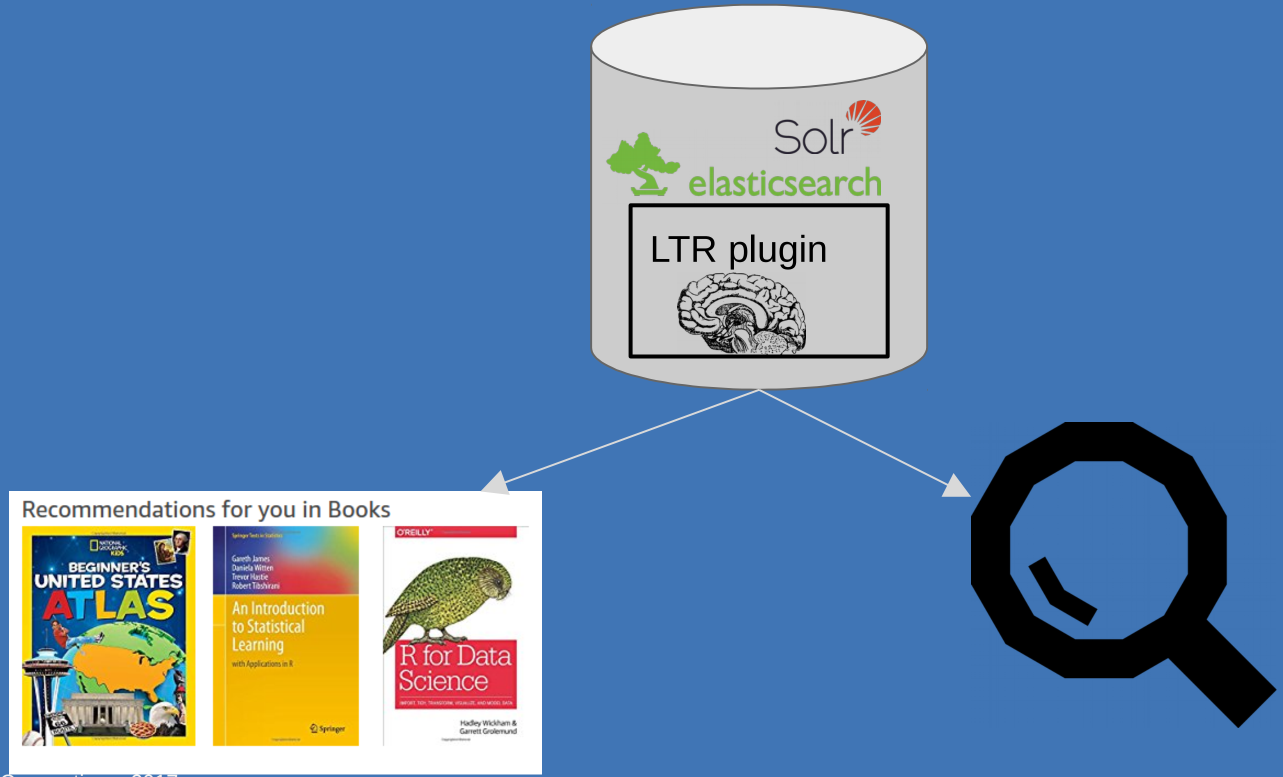
Hand Tuned Team & Infra



LTR Team & Infra



BUT it comes more POOOWER!!
Learning to Rank driven personalized search + recsys



LTR for Job Matching: The Plan

snagajob ENGINEERING

0. Remember this is an iterative process

1.Determine how to measure success

- NDCG@10
- ERR@10

1.Determine how to measure success

Don’t forget UX / UI!

snagajob
ENGINEERING

2. Establish Baseline Ranking Function

“Best shot” at ranking the top-k before rescoring

example: Gaussian distance & freshness decay,
BM25 similarity, geo radius

snagajob
ENGINEERING

2. Establish Baseline Ranking Function

“Real World Considerations”:

- Distance decay wasn’t aggressive enough.
- Freshness decay had the same problem.
- Location facets present interesting edge cases.

snagajob ENGINEERING

2. Establish Baseline Ranking Function

“Real World Considerations”:

- Current thinking is to let baseline ranker handle recall, and the LTR model to optimize precision.

snagajob ENGINEERING

3. Feature Engineering

Start small!

- Brand
- Zipcode
- Title
- Description
- Location

3. Feature Engineering

Then improve constantly….

3. Feature Engineering

Features need not be “search-y” things!

- Content profiles
- Commute distance as a function of roads or transit
- Market forces (supply + demand, etc)

snagajob ENGINEERING

4. Train Models! (LambdaMART)

Data Warehouse

feature values

```
grade,keywords,docId

4,KFC,7555   # KFC
3,KFC,1370   # Pizza Hut
0,KFC,102947 # Uber
4,Macy's,1366 # Macy's
...
```

```
<ensemble>
 <tree id="1" weight="1.0E-4">
  <split>
     <feature>5</feature>
      <threshold> 7.53730
    </threshold>
  <split pos="left">
...
```

4. Train Models! (LambdaMART)

Think about combining latent factor models with your training set!

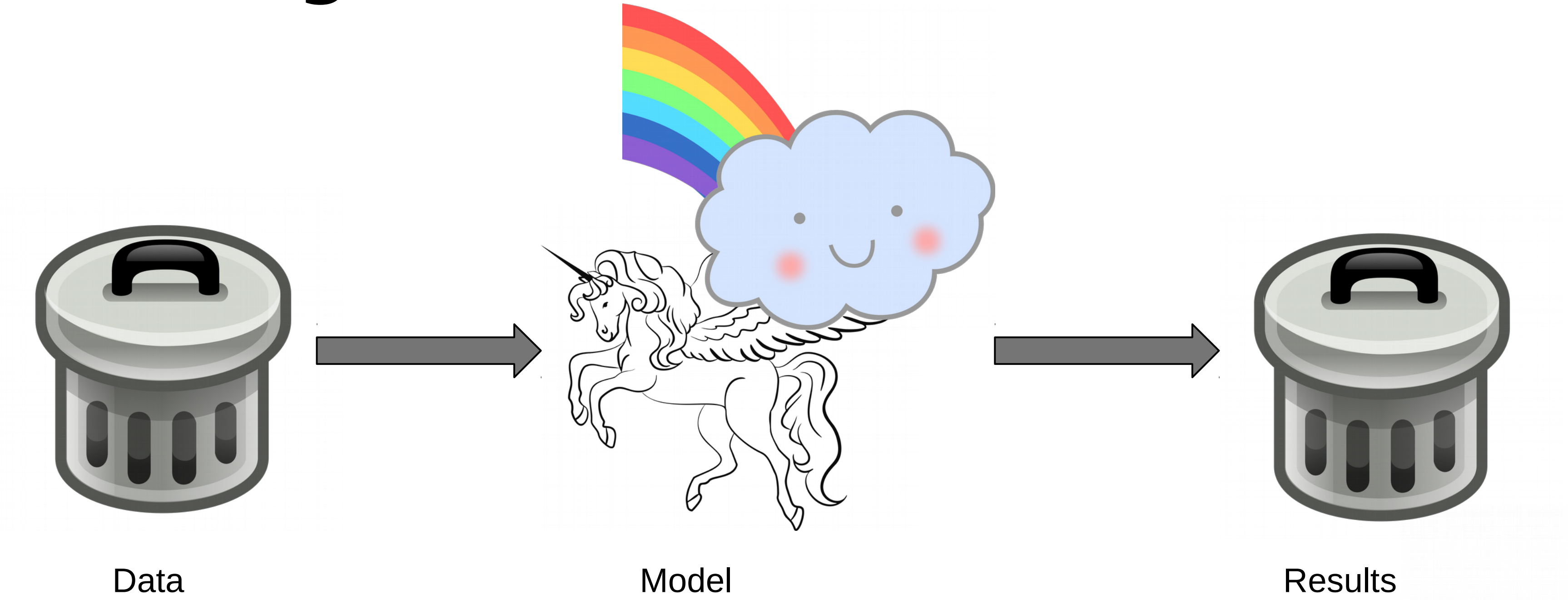
Training - “Real world” considerations

- Ranklib Training performance

Macbook:
2.7MM Judgements - 5 hours (max tree depth of 20)

M4.4Xlarge:
30 minutes - (max tree depth of 100)

Training - “Real world” considerations



snagajob
ENGINEERING

Training - “Real world” considerations

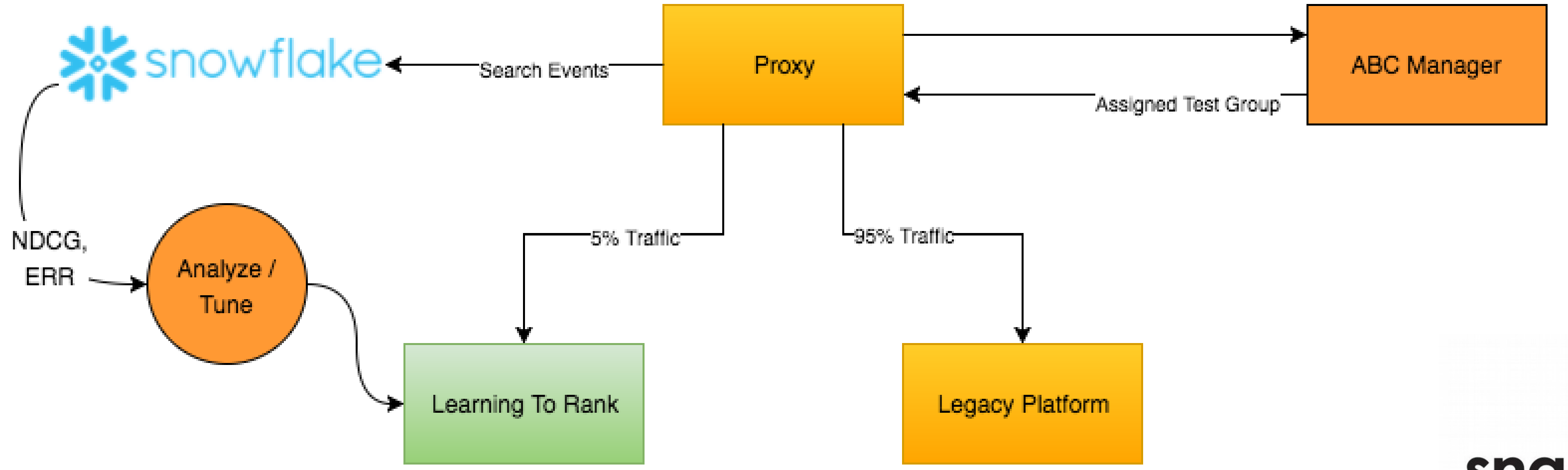
Query dependent features at training time vs. query time.

5. Integrate with an existing platform?

NO.

snagajob
ENGINEERING

5. Integrate with an existing platform?



snagajob
ENGINEERING

6. Profit?

???

snagajob
ENGINEERING

6. Profit?

Model V1: (10 bags, 10 trees, 20 leaves)

NDCG@10: **+20.17%**
ERR@10: **+37.13%**

snagajob
ENGINEERING

Hyperparameters matter!

6. Profit?

Model V2: (10 bags, 50 trees, 100 leaves)

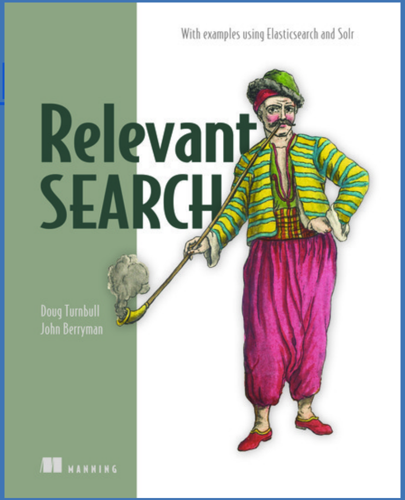
NDCG@10: **+30.17% (+10.17%)**
ERR@10: **+49.06% (+11.93%)**

snagajob
ENGINEERING

Questions…

https://github.com/o19s/elasticsearch-learning-to-rank

Please try it and report bugs!



Discount code **relsearch**