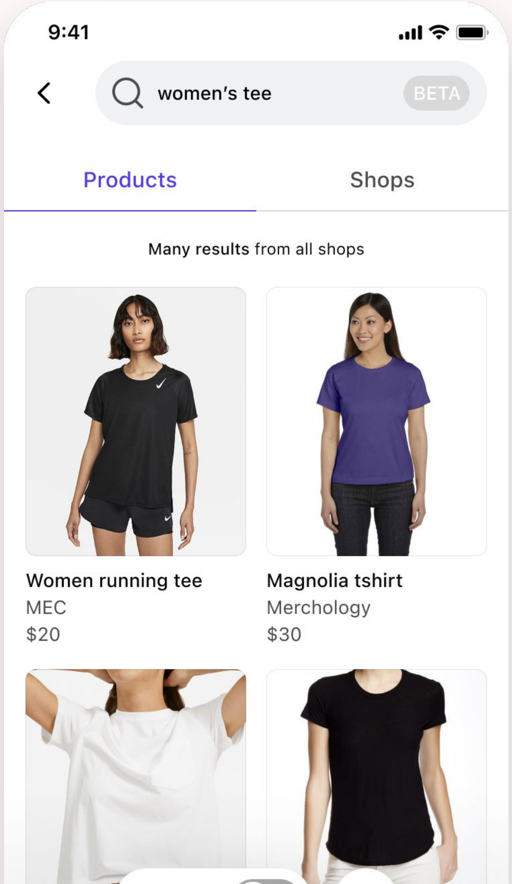
Bayesian Optimization at Shopify

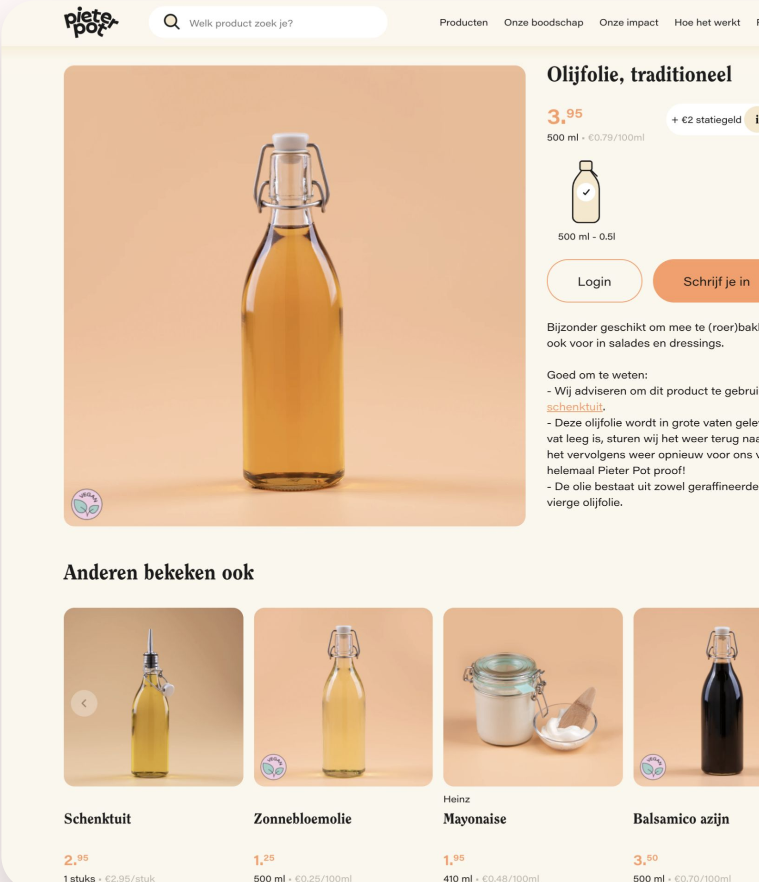
Andy Toulis, Sr. Data Scientist @ Shopify
Doug Turnbull, Relevance Eng @ Shopify

Discovery Experiences

About Discovery Experience at Shopify



- Search and recommendations for millions of merchants
- Empowering merchants by giving them state of the art search and discovery tooling
- Deepening relationships between merchants and their buyers



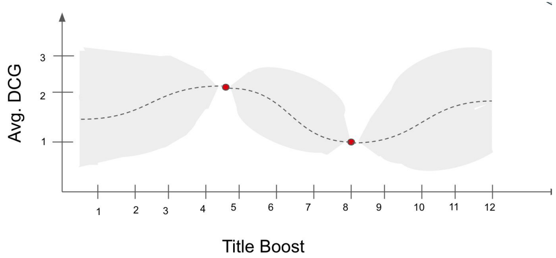
Shoutouts and Citations

- Josh Devins (of Elastic) work with MS Marco using Bayesian optimization
  - https://www.elastic.co/blog/improving-search-relevance-with-data-driven-query-optimization
- Distil.pub's great Bayesian Optimization article - https://distill.pub/2020/bayesian-optimization/
- Live coding Bayesian Optimization from Scratch - https://www.twitch.tv/manningpublications/video/1237632681

NB - https://github.com/o19s/hello-ltr/blob/main/notebooks/elasticsearch/tmdb/bayesian-optimization.ipynb

Motivation

- Jump to LTR with crazy plugin built by mad scientist and lots of infra!!?!?

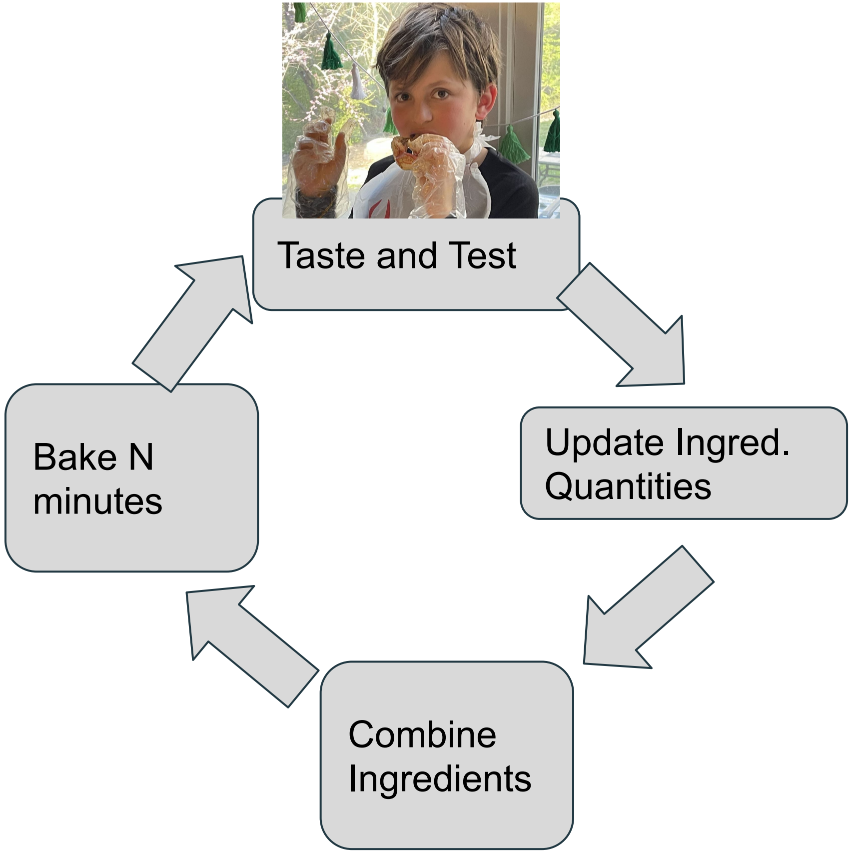


OR

- Just use out of the box Elasticsearch with optimized query and index features?

Let’s bake a cake

<u>Ingredient</u>	<u>Quantity??</u>
Flour	
Baking Powder	
Milk	
Sugar	
Eggs	
Time	

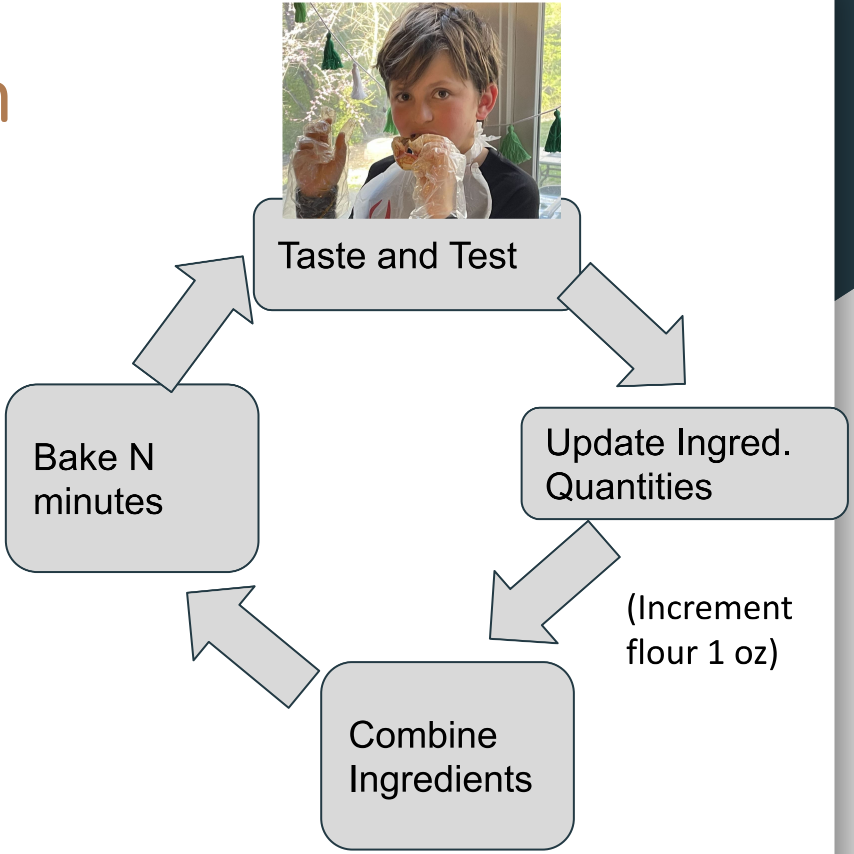


Let’s not do this

One option - grid search

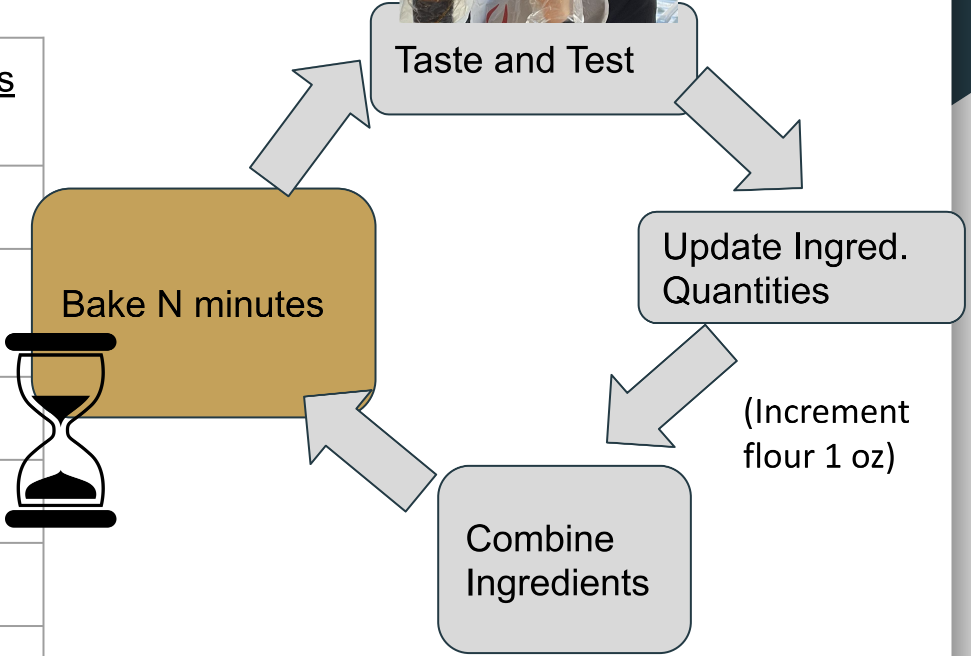
(Try every combination of ingredients)

Ingredient	Quantity Range	Increments
Flour	8 oz – 3 cups	1 oz
Baking Powder	1 tsp - 8 tsp	1 tsp
Milk	8 oz - 3 cups	1 oz
Sugar	1 tsp - 8 tsp	1 tsp
Eggs	1 egg - 4 eggs	1 egg
Time	10 min. - 25 min.	1 min.



Problem - baking time consuming

Ingredient	Quantity Range	Increments
Flour	8 oz – 24 oz	1 oz
Baking Powder	0 tsp - 8 tsp	1 tsp
Milk	8 oz - 24 oz	1 oz
Sugar	0 tsp - 8 tsp	1 tsp
Eggs	0 egg - 4 eggs	1 egg
Time	10 min. - 25 min.	1 min.



Problem - baking time consuming

Ingredient	Quantity Range	Increments	Combinations
Flour	8 oz – 24 oz	1 oz	16
Baking Powder	0 tsp - 8 tsp	1 tsp	9
Milk	8 oz - 24 oz	1 oz	16
Sugar	0 tsp - 8 tsp	1 tsp	9
Eggs	0 egg - 4 eggs	1 egg	5
Time	10 min. - 25 min.	1 min.	15

To find best cake, must Try:

16 * 9 * 16 * 9 * 5 * 15 =

1,555,200!!!

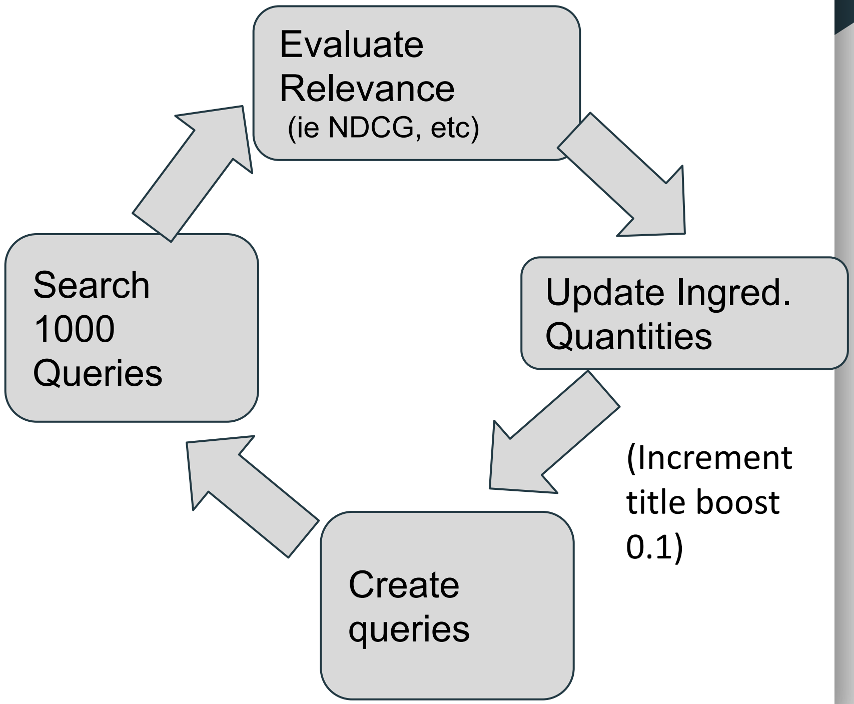
~ 31 million minutes to try every combination

~ 60 years of baking

Search Relevance optimization like baking

(Expensive to try 1000s of queries with one set of boosts)

~~Ingredient~~ **Boost** (or other param)	Quantity Range	Increments
Title	0…20	1
Body	0...20	1
Title k1	0…2	0.1
Title b	0…2	0.1
title min-should-match	0%-100%	10%
…		



Grid search also not ideal...

~~Ingredient~~ **Boost** (or other param)	Quantity Range	Increments	Combinations
Title	0…20	1	21
Body	0...20	1	21
Title k1	0…2	0.1	21
Title b	0…2	0.1	21
title min-should-match	0%-100%	10%	11
…			

To find best cake, must Try:

21 * 21 * 21 * 21 * 21 * 11 =

2,139,291 params!!!

One run = 1 min…

2,139,291 minutes

~ 4 years of compute

Intuition, what if we tracked good / bad combos?

This observation seems pretty good!
Mean NDCG = 0.75

<u>Boost</u> (or other param)	**<u>Quantity</u>**
Title	15
Body	5
Title k1	1.2
Title b	1
title min-should-match	25%
…	

Intuition:

<u>NEARBY PARAMS ALSO LIKELY GOOD</u>
Mean NDCG ~ 0.75

<u>Boost</u> (or other param)	**<u>Updated Quantity</u>**
Title	15 + **1**
Body	5 - **1**
Title k1	1.2 **+ 0.1**
Title b	1
title min-should-match	25%
…	

Next – try ***nearby*** or ***distant*** observation?

EXPLOIT *VS* ***EXPLORE:***

Nearby may be a little bit better?
(but only incrementally)
Mean NDCG > 0.75?

Boost (or other param)	**Updated Quantity**
Title	15 + **1**
Body	5 - **1**
Title k1	1.2 **+ 0.1**
Title b	1
title min-should-match	25%
…	

Farther away may gain more knowledge?
(but with greater downside risk)
Mean NDCG ~ 0.9??

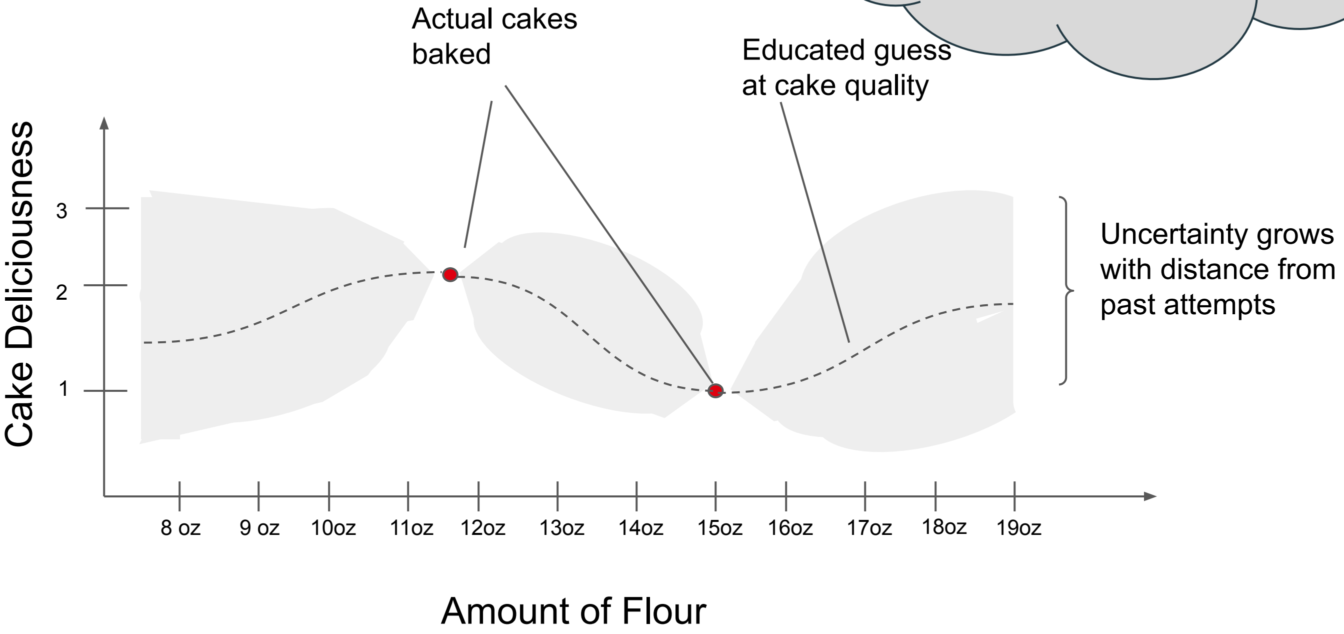
Boost (or other param)	**Updated Quantity**
Title	15 - **5**
Body	5 + **4**
Title k1	1.2 **+ 0.8**
Title b	1 - **0.2**
title min-should-match	25%
…	

Gaussian Regression

Is there enough possible upside to exploring very far?

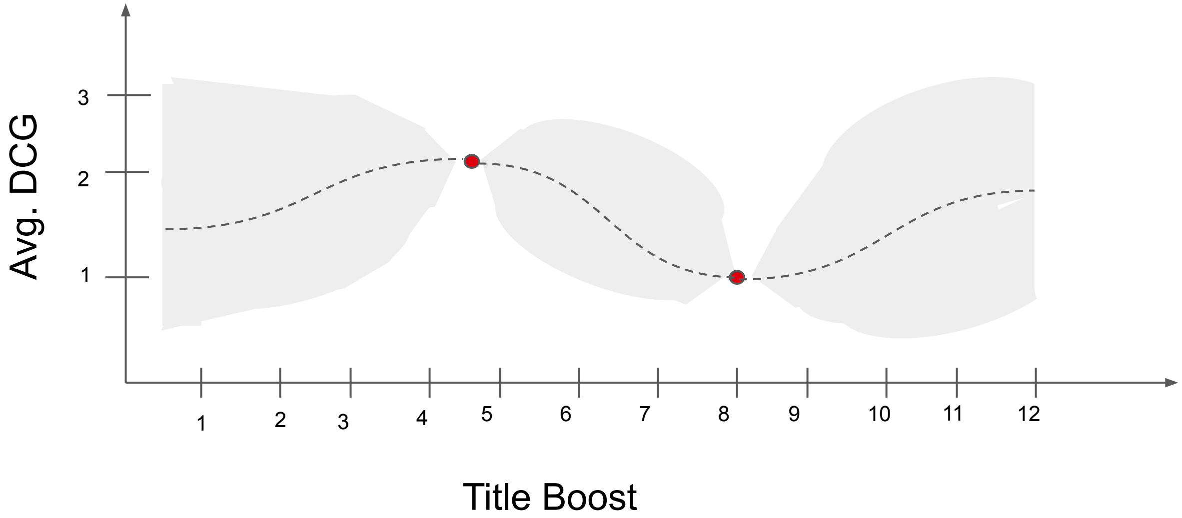
Or better to stay close to existing observation?

Models ***BOTH*** predictions ***AND*** a gaussian distribution of that prediction's likelihood



Gaussian Regression - Search

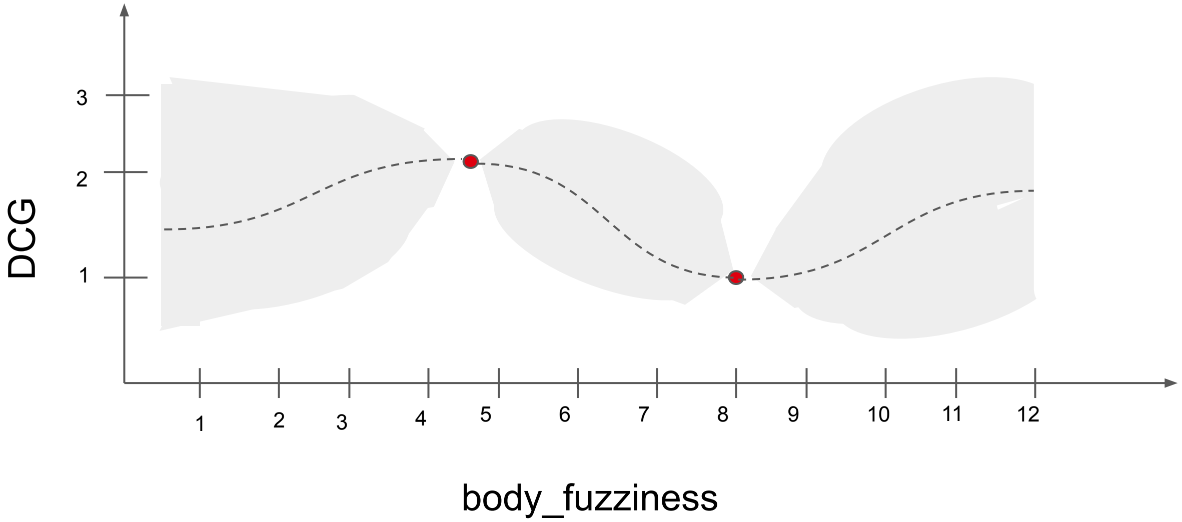
What's the next 'title boost' we should try?



Practically speaking...

Using Mustache Templates

https://www.elastic.co/guide/en/elasticsearch/reference/current/search-template.html

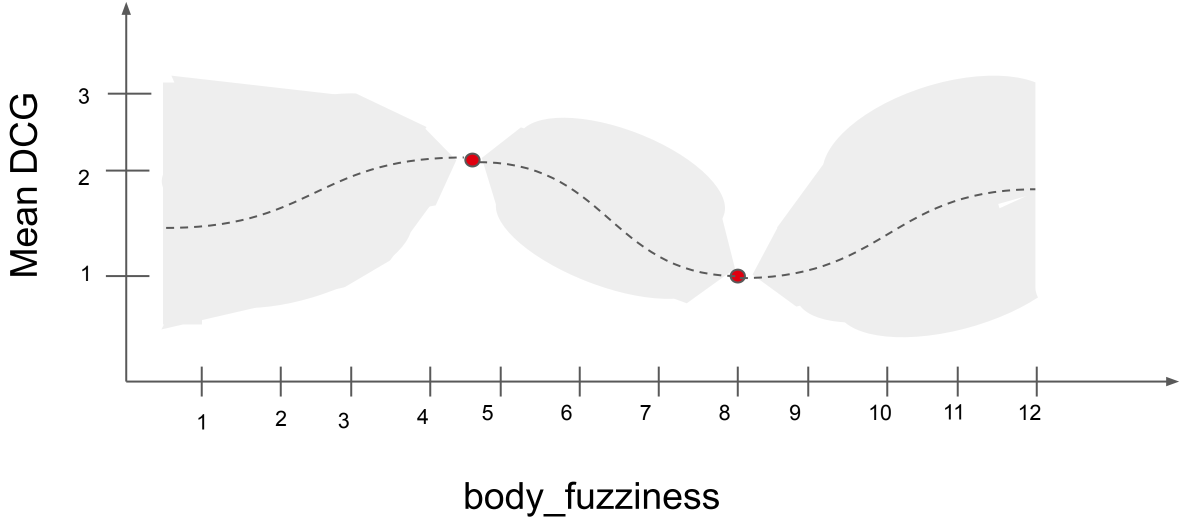



```
{
  "query": {
    "bool": {
      "must": [
        {
          "bool": {
            "should": [
              {
                "match": {
                  "title": {
                    "query": "{{query_term}}",
                    "boost": {{title_boost}},
                  }
                }
              },
              {
                "match": {
                  "body": {
                    "query": "{{query_term}}",
                    "fuzziness": {{body_fuzziness}},
                    "prefix_length": 4,
                    "fuzzy_transpositions": true,
                    "boost": {{fuzzy_body_boost}}
                  }
                }
              },
```

Params
~ baking
ingredients

Choose next candidate

1. Select N random points
2. Probe in Gaussian Process Regressor model
3. Score the best one to explore
4. ~~Bake Cake~~ Try out searches with new params!




```
{
  "query": {
    "bool": {
      "must": [
        {
          "bool": {
            "should": [
              {
                "match": {
                  "title": {
                    "query": "{{query_term}}",
                    "boost": {{title_boost}},
                  }
                }
              },
              {
                "match": {
                  "body": {
                    "query": "{{query_term}}",
                    "fuzziness": {{body_fuzziness}},
                    "prefix_length": 4,
                    "fuzzy_transpositions": true,
                    "boost": {{fuzzy_body_boost}}
                  }
                }
              },
```

(we also use index-time params…)

Training

Open sourced demo in OpenSource Connection’s “Hello LTR” project
https://github.com/o19s/hello-ltr/blob/main/notebooks/elasticsearch/tmdb/bayesian-optimization.ipynb

```
from sklearn.gaussian_process import GaussianProcessRegressor
import pandas as pd

runs_so_far = pd.DataFrame(sorted_by_perf)

y_train = runs_so_far['mean_dcg']
x_train = runs_so_far[['title_boost','fuzzy_body_boost','body_boost']]


gpr = GaussianProcessRegressor()
gpr.fit(x_train.to_numpy(), y_train.to_numpy())
```

Probability of Improvement

Score using the probability a selected point will yield *any improvement*

Current Max Observation
Title Boost = 10, Body Fuzziness = 3, Body Boost = 2
DCG = **2.1**

Title Boost	Body Fuzziness	Body Boost	GPR Pred. DCG	GPR Std Dev	(Expectation - Max) / stddev
0	1	5	1.5	1.0	(1.5 - 2.1) / 1.0 = -0.6
10	3	1	2.3	0.2	(2.3 - 2.1) / 0.2 = 1.0
12	2	4	1.2	0.6	(1.2 - 2.1) / 0.6 = -1.5

Max Candidate Chosen

Numerator ~ ‘opportunity’

https://github.com/o19s/hello-ltr/blob/main/notebooks/elasticsearch/tmdb/bayesian-optimization.ipynb

```
explore_points['opportunity'] = explore_points['prediction'] - best_cg
explore_points
```

Out[20]:

	prediction	std_dev	opportunity
0	4.675000e-01	0.00001	-0.4850
1	4.522243e-142	1.00000	-0.9525

Denominator - is this a sure thing?

https://github.com/o19s/hello-ltr/blob/main/notebooks/elasticsearch/tmdb/bayesian-optimization.ipynb

`norm.cdf` scales this between 0-1 to give us more of a probability.

In [22]:
```
from scipy.stats import norm

std_dev = 0.00000001

norm.cdf( (explore_points['opportunity']) / std_dev)
```

Out[22]:
```
array([0., 0.])
```

In [23]:
```
std_dev = 0.999999

norm.cdf( (explore_points['opportunity']) / std_dev)
```

Out[23]:
```
array([0.313838  , 0.17042149])
```

Moves the mass of the probability distribution above current max

Theta - control explore vs exploit

https://github.com/o19s/hello-ltr/blob/main/notebooks/elasticsearch/tmdb/bayesian-optimization.ipynb

```
explore_points['opportunity'] = explore_points['prediction'] - best_cg - theta
```

High theta = Explore: completely ignore opportunity, choose areas of high std dev

Low theta = Exploit: use a lot of signal from opportunity

Probability of Improvement

Score using the probability a selected point will yield *any improvement*

<u>Current Max Observation</u>
Title Boost = 10, Body Fuzziness = 3, Body Boost = 2
DCG = **2.1; Theta = 20**

Title Boost	Body Fuzziness	Body Boost	GPR Pred. DCG	GPR Std Dev	(Expectation - Max - Theta) / stddev
0	1	5	1.5	1.0	(1.5 - 2.1 - 20) / 1.0 = ***-20.6***
10	3	1	2.3	0.2	(2.3 - 2.1 - 20) / 0.2 = -99
12	2	4	1.2	0.6	(1.2 - 2.1 - 20) / 0.6 = -20.9

Max Candidate Chosen

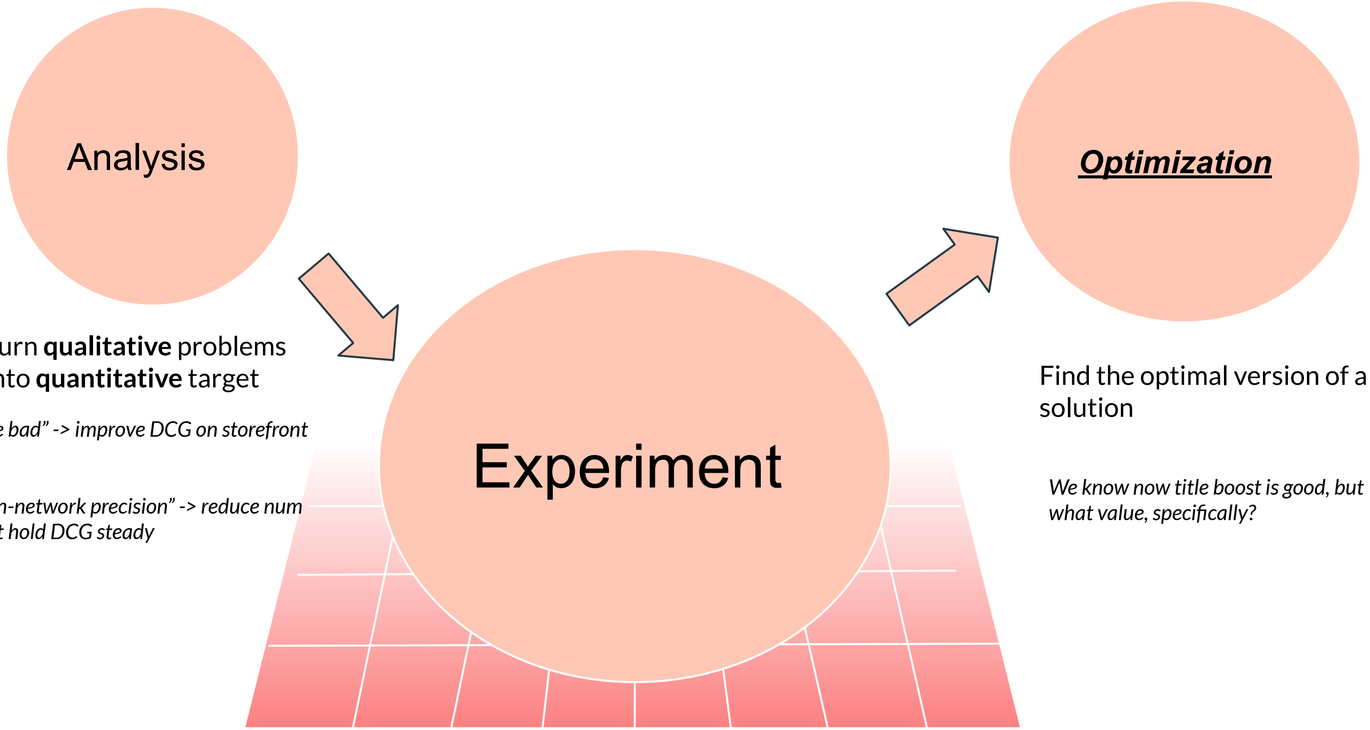
Even better - Expected Improvement

Not just *whether* an improvement will occur but *how much* improvement to expect*!*

$$EI(x) = \begin{cases} (\mu_t(x) - f(x^+) - \epsilon)\Phi(Z) + \sigma_t(x)\phi(Z), & \text{if } \sigma_t(x) > 0 \\ 0, & \text{if } \sigma_t(x) = 0 \end{cases}$$

$$Z = \frac{\mu_t(x) - f(x^+) - \epsilon}{\sigma_t(x)}$$

Boogie - our offline experimentation framework



Turn **qualitative** problems into **quantitative** target

"Relevance bad" -> improve DCG on storefront judgments

"Improve in-network precision" -> reduce num results, but hold DCG steady

Try out different approaches to solve the problem

How to best improve DCG?

Find the optimal version of a solution

We know now title boost is good, but what value, specifically?

Example Walkthrough

Bring your business online

Create an ecommerce website backed by powerful tools that help you find customers, drive sales, and manage your day-to-day.

Explore more examples →

FOOD & DRINK
Alfred

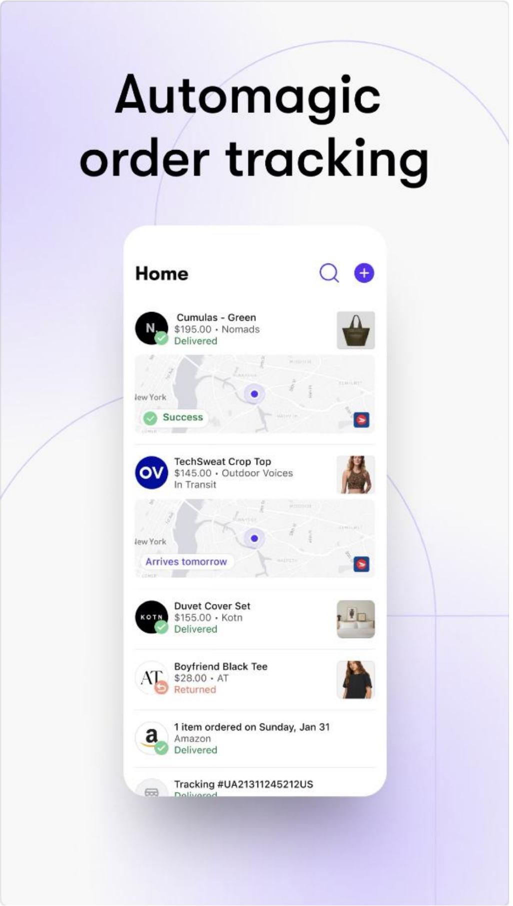
JEWELRY
Corey Moranis

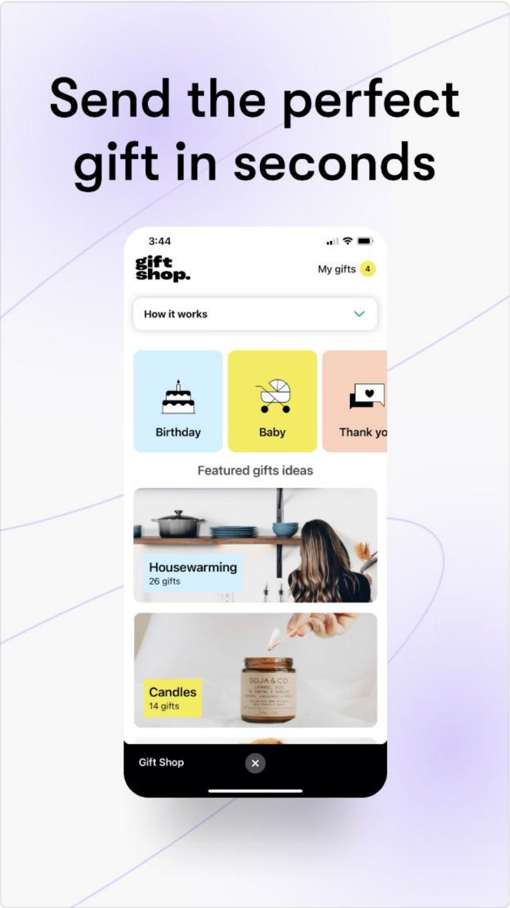
FOOD & DRINK
Detour Coffee

BEAUTY & COSMETICS
Then I Met You

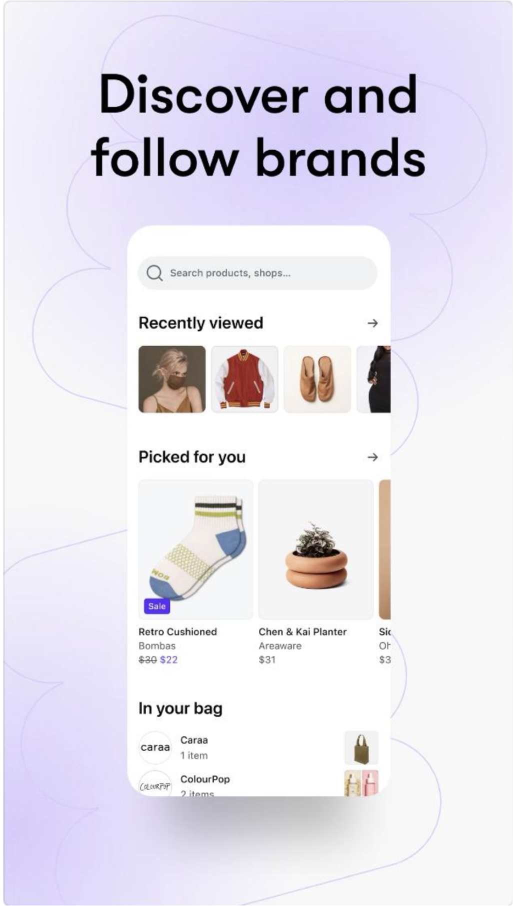
HOME & GARDEN
Miss Boon

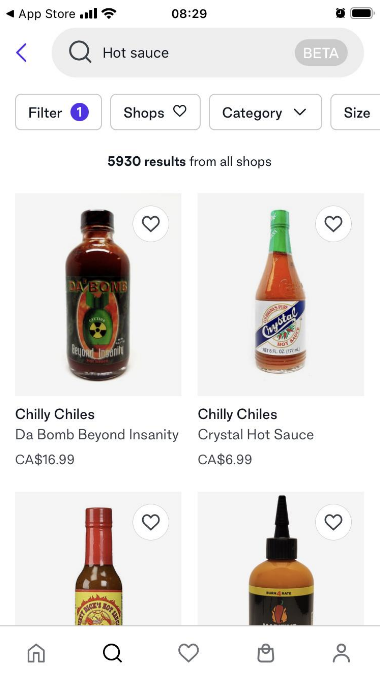
Shop app

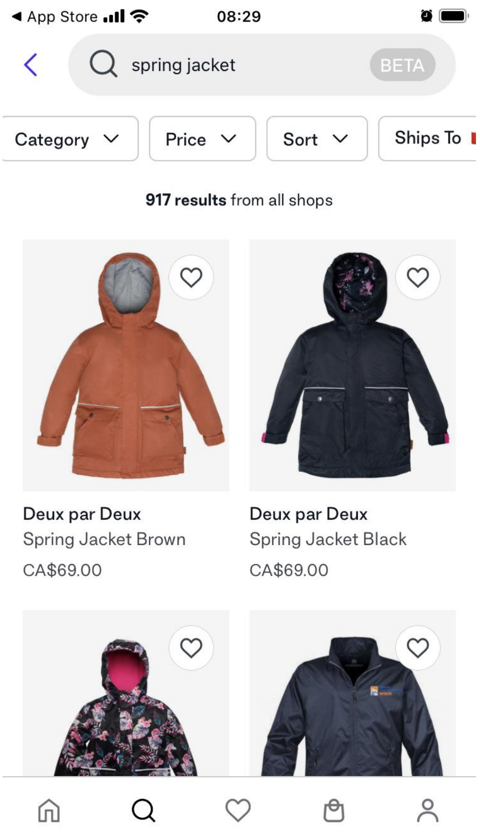

Automagic order tracking
Home
Cumulas - Green
$195.00 · Nomads
Delivered
Success
TechSweat Crop Top
$145.00 · Outdoor Voices
In Transit
Arrives tomorrow
Duvet Cover Set
$155.00 · Kotn
Delivered
Boyfriend Black Tee
$28.00 · AT
Returned
1 item ordered on Sunday, Jan 31
Amazon
Delivered
Tracking #UA21311245212US

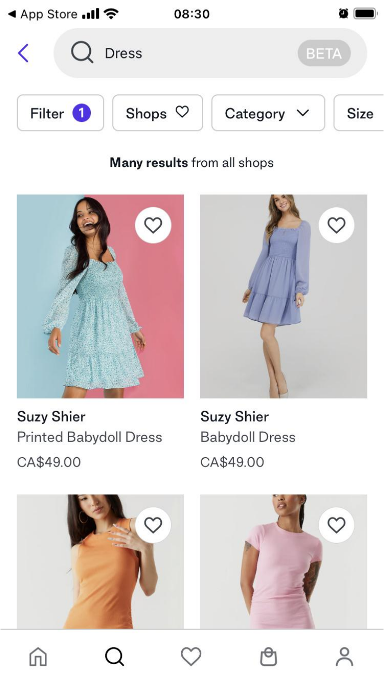

Send the perfect gift in seconds
3:44
gift shop.
My gifts 4
How it works
Birthday
Baby
Featured gifts ideas
Housewarming
26 gifts
Candles
14 gifts
Gift Shop

Shop app


Discover and follow brands
Search products, shops...
Recently viewed
Picked for you
Sale
Retro Cushioned
Bombas
$30 $22
Chen & Kai Planter
Areaware
$31
In your bag
caraa
Caraa
1 item
ColourPop


App Store
08:29
Hot sauce
BETA
Filter 1
Shops
Category
Size
5930 results from all shops
Chilly Chiles
Da Bomb Beyond Insanity
CA$16.99
Chilly Chiles
Crystal Hot Sauce
CA$6.99


App Store
08:29
spring jacket
BETA
Category
Price
Sort
Ships To
917 results from all shops
Deux par Deux
Spring Jacket Brown
CA$69.00
Deux par Deux
Spring Jacket Black
CA$69.00


App Store
08:30
Dress
BETA
Filter 1
Shops
Category
Size
Many results from all shops
Suzy Shier
Printed Babydoll Dress
CA$49.00
Suzy Shier
Babydoll Dress
CA$49.00

Going back

old results for “sailor moon ring”

Going back

old results for “sailor moon ring”

“moon ring”

“sailor moon”

Our setup (at the time)

phrase matching

stemming

minimum
should match

...

we could see plenty of good products lower in search results!

Right setup, wrong parameters

Parameter	**Value**
Match Boost	High?
Phrase Boost	Medium?
Stemmed Match Boost	Medium?
Stemmed Phrase Boost	Low?

“OST x Sailor Moon -
Warrior Silence Grave Ring”

“OST x Sailor Moon -
Beautiful Warrior Moonlight Miss Ring”

“OST x Sailor Moon -
Warrior Sailor Moon Garnet Ring”

“OST x Sailor Moon -
Cutie Sailor Moon Road Ring”

new results for
“sailor moon ring”

Recipe for optimization

Recipe for optimization

1. Set up a **training** sample to optimize on, and a **hold-out** for evaluation

Recipe for optimization

1. Set up a **training** sample to optimize on, and a **hold-out** for evaluation

Training

“jeans”, “wallet”,
“t-shirt”, “k pop”, …

Hold-out

“washi tape”, “notebook”,
“jacket”, “airpods”,
“shorts”, “candy”,
“sailor moon ring”, …

Recipe for optimization

1. Set up a **training** sample to optimize on, and a **hold-out** for evaluation

Query	Product	Grade
jeans	original jeans	0.42
	grey jeans	0.13
	jean outfit	0.06
	jean jacket	0.02
Goal: optimize DCG		

judgments for the query “jeans”

Recipe for optimization

1. Set up a training sample to optimize on, and a hold-out for evaluation

2. Plan which **parameters** you will optimize

Recipe for optimization

1. Set up a training sample to optimize on, and a hold-out for evaluation

2. Plan which **parameters** you will optimize

Parameter
Match Boost
Phrase Boost
Stemmed Match Boost
Stemmed Phrase Boost

Recipe for optimization

1. Set up a training sample to optimize on, and a hold-out for evaluation

2. Plan which **parameters** you will optimize

Parameter
Match Boost
Phrase Boost
Stemmed Match Boost
Stemmed Phrase Boost
Minimum Should Match

share parameters where possible!

Recipe for optimization

1. Set up a training sample to optimize on, and a hold-out for evaluation

2. Plan which parameters you will optimize

3. Set reasonable **ranges** for parameters

Recipe for optimization

1. Set up a training sample to optimize on, and a hold-out for evaluation

2. Plan which parameters you will optimize

3. Set reasonable ranges for parameters

4. **Run the optimizer** on your training sample

Recipe for optimization

1. Set up a training sample to optimize on, and a hold-out for evaluation

2. Plan which parameters you will optimize

3. Set reasonable ranges for parameters

4. **Run the optimizer** on your training sample

   ✓ works for non-linear setups
   (example: multiplied scores)

Recipe for optimization

1. Set up a training sample to optimize on, and a hold-out for evaluation

2. Plan which parameters you will optimize

3. Set reasonable ranges for parameters

4. Run the optimizer on your training sample

5. **Explore** the best models

Recipe for optimization

1. Set up a training sample to optimize on, and a hold-out for evaluation
2. Plan which parameters you will optimize
3. Set reasonable ranges for parameters
4. Run the optimizer on your training sample
5. Explore the best models
6. **Iterate if needed!**
   (training data, metric to optimize, parameter ranges, optimizer settings)

Recipe for optimization

✓ Applied each relevance cycle

1. Set up a training sample to optimize on, and a hold-out for evaluation
2. Plan which parameters you will optimize
3. Set reasonable ranges for parameters
4. Run the optimizer on your training sample
5. Explore the best models
6. Iterate if needed!

Experiment time!

Parameters

```
from boogie.optimize import Param

params = [
    Param(name="title_k1",
          param_type="index",
          param_min=0.2,
          param_max=3.0,
          param_step=0.2,
          default=1.2),
    Param(name="title_b",
          param_type="index",
          param_min=0.2,
          param_max=1.0,
          param_step=0.1,
          default=0.5),

    Param(name="match_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
    Param(name="phrase_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
    Param(name="stemmed_match_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
    Param(name="stemmed_phrase_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
]
```

Index parameters

Query parameters

Parameters

```
from boogie.optimize import Param

params = [
    Param(name="title_k1",
          param_type="index",
          param_min=0.2,
          param_max=3.0,
          param_step=0.2,
          default=1.2),
    Param(name="title_b",
          param_type="index",
          param_min=0.2,
          param_max=1.0,
          param_step=0.1,
          default=0.5),

    Param(name="match_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
    Param(name="phrase_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
    Param(name="stemmed_match_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
    Param(name="stemmed_phrase_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
]
```

only focusing on
product titles

Parameters

BM25

- b

- k

```
from boogie.optimize import Param

params = [
    Param(name="title_k1",
          param_type="index",
          param_min=0.2,
          param_max=3.0,
          param_step=0.2,
          default=1.2),
    Param(name="title_b",
          param_type="index",
          param_min=0.2,
          param_max=1.0,
          param_step=0.1,
          default=0.5),

    Param(name="match_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
    Param(name="phrase_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
    Param(name="stemmed_match_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
    Param(name="stemmed_phrase_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
]
```

BM25 parameters in their typical ranges

Parameters

unstemmed
- match
- match_phrase

```
from boogie.optimize import Param

params = [
    Param(name="title_k1",
          param_type="index",
          param_min=0.2,
          param_max=3.0,
          param_step=0.2,
          default=1.2),
    Param(name="title_b",
          param_type="index",
          param_min=0.2,
          param_max=1.0,
          param_step=0.1,
          default=0.5),

    Param(name="match_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
    Param(name="phrase_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
    Param(name="stemmed_match_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
    Param(name="stemmed_phrase_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
]
```

Boosts ranging between 0.1 and 1.0

Parameters

stemmed

- match

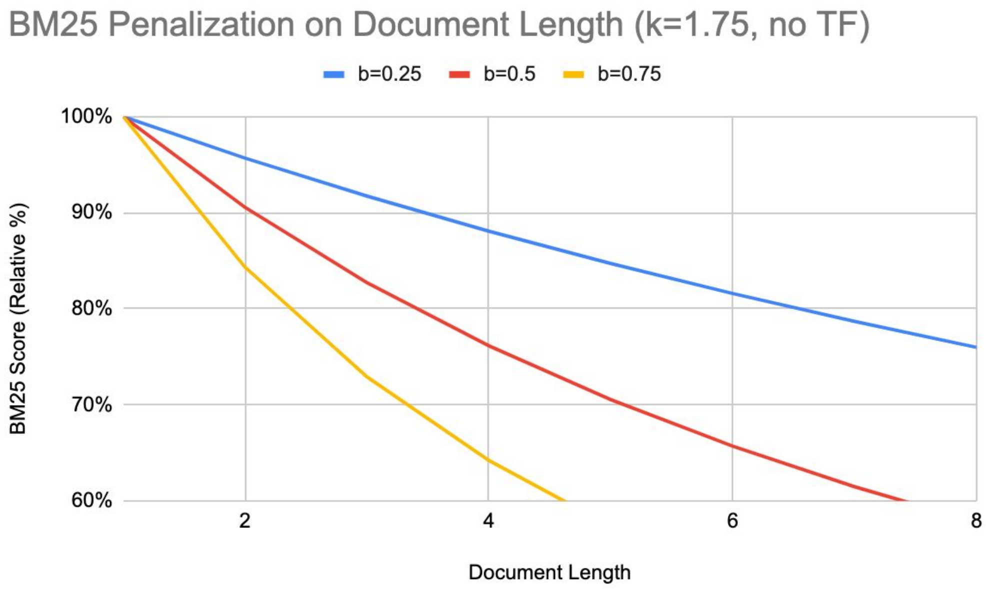
- match_phrase

```python
from boogie.optimize import Param

params = [
    Param(name="title_k1",
          param_type="index",
          param_min=0.2,
          param_max=3.0,
          param_step=0.2,
          default=1.2),
    Param(name="title_b",
          param_type="index",
          param_min=0.2,
          param_max=1.0,
          param_step=0.1,
          default=0.5),

    Param(name="match_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
    Param(name="phrase_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
    Param(name="stemmed_match_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
    Param(name="stemmed_phrase_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
]
```

Same idea, but for stemmed product titles

(BM25 detour)

Role of "b"

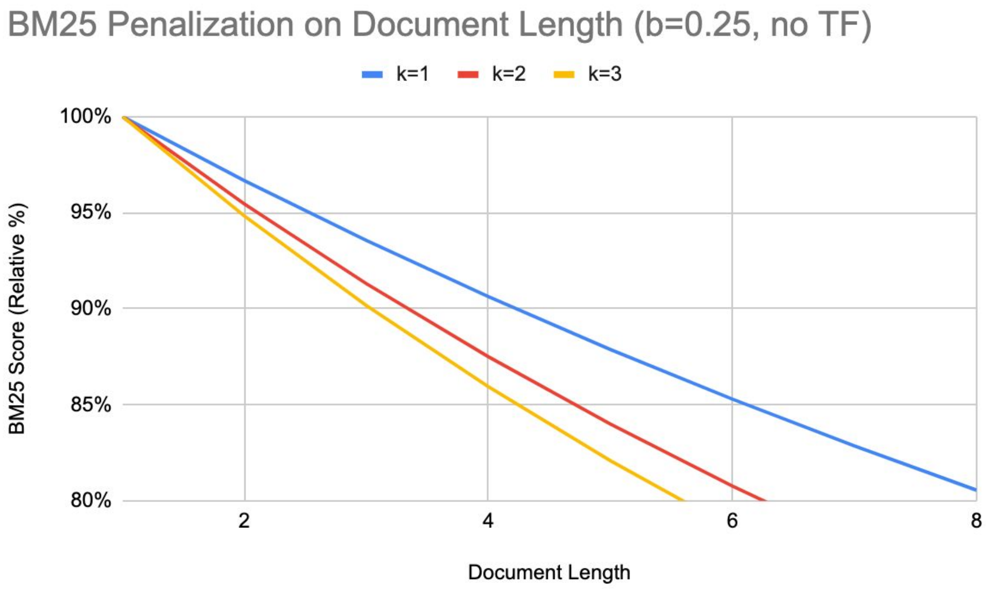


BM25 Penalization on Document Length (k=1.75, no TF)
b=0.25
b=0.5
b=0.75
100%
90%
80%
70%
60%
BM25 Score (Relative %)
2
4
6
8
Document Length

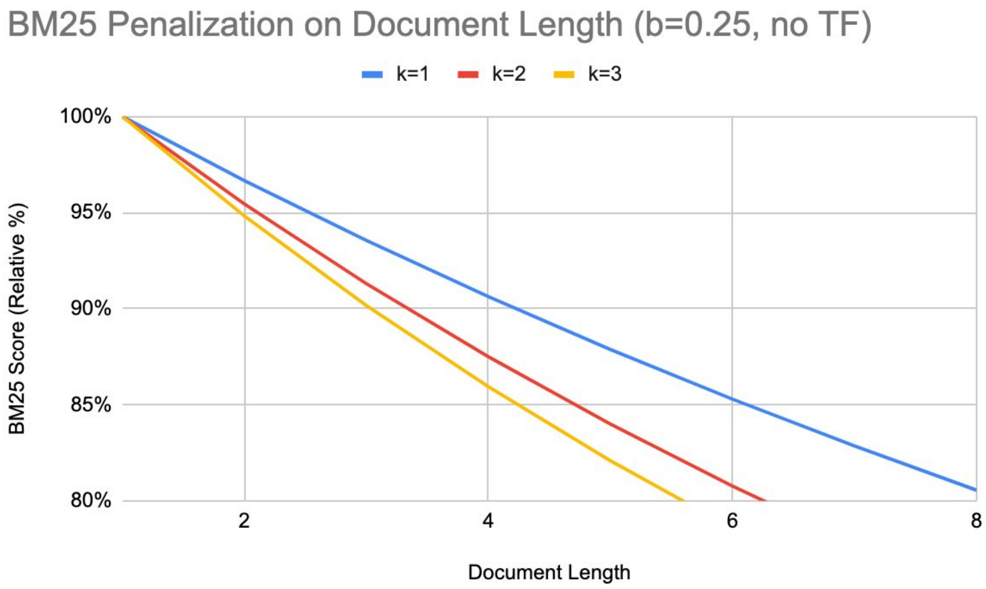
Role of "k"

$$\frac{1}{1 + k\left[(1-b) + \left(b \cdot \frac{\text{length}_{doc}}{\text{length}_{avg}}\right)\right]}$$

no term frequencies



Role of “k”


BM25 Penalization on Document Length (b=0.25, no TF)
k=1
k=2
k=3
100%
95%
90%
85%
80%
BM25 Score (Relative %)
2
4
6
8
Document Length

Back to the experiment

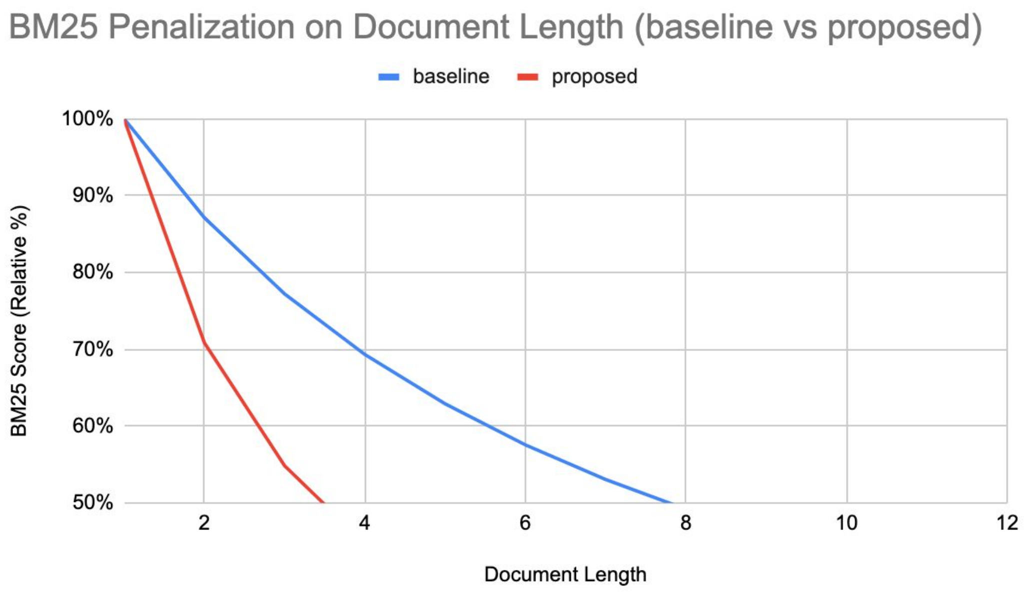
Results!

Parameter	Value
b (BM25)	High
k (BM25)	High
Match Boost	High
Phrase Boost	High
Stemmed Match Boost	Low
Stemmed Phrase Boost	High

Results?

Parameter	Value
b (BM25)	High
k (BM25)	High
Match Boost	High
Phrase Boost	High
Stemmed Match Boost	Low
Stemmed Phrase Boost	High

Results?


BM25 Penalization on Document Length (baseline vs proposed)
baseline
proposed
100%
90%
80%
70%
60%
50%
BM25 Score (Relative %)
2
4
6
8
10
12
Document Length

product titles provide critical information!

“OST x Sailor Moon -
Beautiful Warrior Moonlight Miss Ring”

First lesson: optimizers are lazy

“shirt”

First lesson: optimizers are lazy

"shirt"

if there is a loophole, the optimizer will find it

First lesson: optimizers are lazy

(source: https://dragonflyai.co)

Attempt two

```
from boogie.optimize import Param

params = [
    Param(name="title_k1",
          param_type="index",
          param_min=0.2,
          param_max=3.0,
          param_step=0.1,
          default=1.2),
    Param(name="title_b",
          param_type="index",
          param_min=0.10,
          param_max=0.40,
          param_step=0.05,
          default=0.2),

    Param(name="match_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
    Param(name="phrase_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
    Param(name="stemmed_match_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
    Param(name="stemmed_phrase_boost",
          param_type="query",
          param_min=0.1,
          param_max=1.0,
          param_step=0.1,
          default=1.0),
]
```

Forcing exploration around lower “b” values

Optimization

Next explorations:

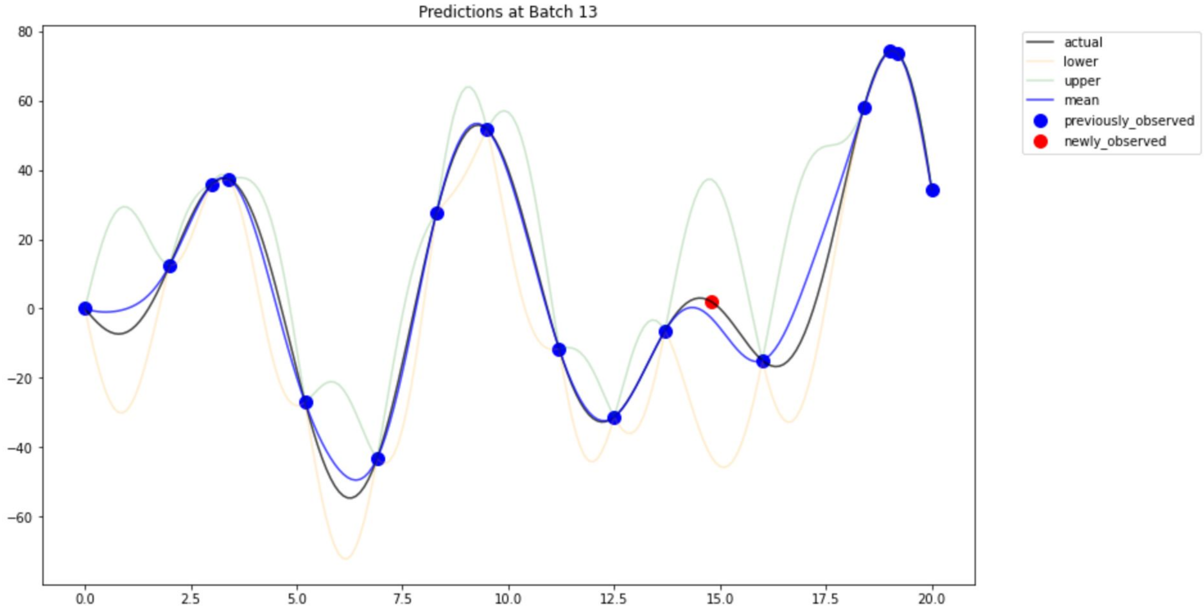
title_k1	title_b	match_boost	phrase_boost	stemmed_match_boost	stemmed_phrase_boost
2.8	0.2	1.0	0.2	0.1	1.0
2.8	0.2	1.0	0.3	0.1	1.0
2.8	0.3	1.0	0.2	0.1	1.0
2.8	0.2	1.0	0.2	0.2	1.0
2.8	0.3	1.0	0.2	0.2	1.0
2.8	0.3	1.0	0.3	0.1	1.0
2.8	0.2	1.0	0.3	0.2	1.0
2.8	0.2	1.0	0.1	0.1	1.0
2.6	0.2	1.0	0.2	0.1	1.0
2.6	0.2	1.0	0.3	0.1	1.0
2.8	0.3	1.0	0.3	0.2	1.0
2.8	0.3	1.0	0.1	0.1	1.0
2.8	0.2	1.0	0.1	0.2	1.0
2.8	0.4	1.0	0.2	0.1	1.0
2.8	0.3	1.0	0.1	0.2	1.0
2.8	0.4	1.0	0.2	0.2	1.0

Optimization

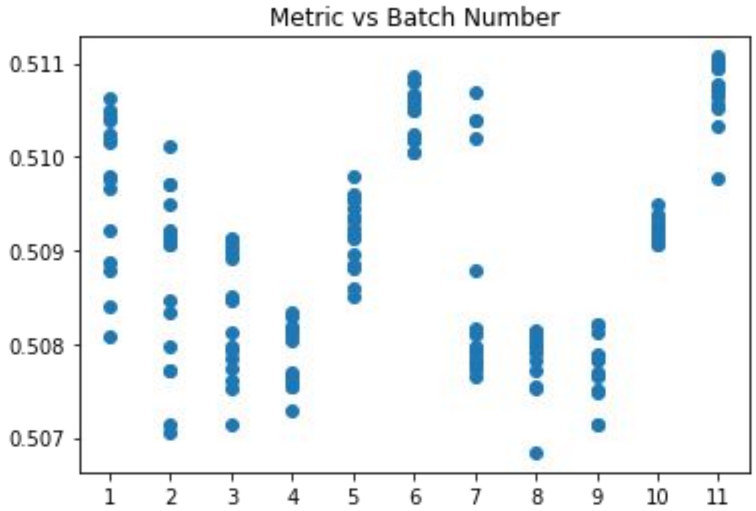
Next explorations:

title_k1	title_b	match_boost	phrase_boost	stemmed_match_boost	stemmed_phrase_boost	prediction	prediction_stddev	opportunity	expected_improvement
2.8	0.2	1.0	0.2	0.1	1.0	0.535540	0.000518	0.024276	0.023276
2.8	0.2	1.0	0.3	0.1	1.0	0.535465	0.000509	0.024202	0.023202
2.8	0.3	1.0	0.2	0.1	1.0	0.535433	0.000502	0.024170	0.023170
2.8	0.2	1.0	0.2	0.2	1.0	0.535374	0.000476	0.024111	0.023111
2.8	0.3	1.0	0.2	0.2	1.0	0.535326	0.000460	0.024063	0.023063
2.8	0.3	1.0	0.3	0.1	1.0	0.535322	0.000493	0.024059	0.023059
2.8	0.2	1.0	0.3	0.2	1.0	0.535299	0.000467	0.024036	0.023036
2.8	0.2	1.0	0.1	0.1	1.0	0.535270	0.000530	0.024007	0.023007
2.6	0.2	1.0	0.2	0.1	1.0	0.535262	0.000501	0.023999	0.022999
2.6	0.2	1.0	0.3	0.1	1.0	0.535218	0.000491	0.023955	0.022955
2.8	0.3	1.0	0.3	0.2	1.0	0.535215	0.000450	0.023952	0.022952
2.8	0.3	1.0	0.1	0.1	1.0	0.535182	0.000516	0.023919	0.022919
2.8	0.2	1.0	0.1	0.2	1.0	0.535112	0.000488	0.023849	0.022849
2.8	0.4	1.0	0.2	0.1	1.0	0.535104	0.000491	0.023841	0.022841
2.8	0.3	1.0	0.1	0.2	1.0	0.535080	0.000473	0.023817	0.022817
2.8	0.4	1.0	0.2	0.2	1.0	0.535076	0.000449	0.023813	0.022813

If we were higher-dimensional beings



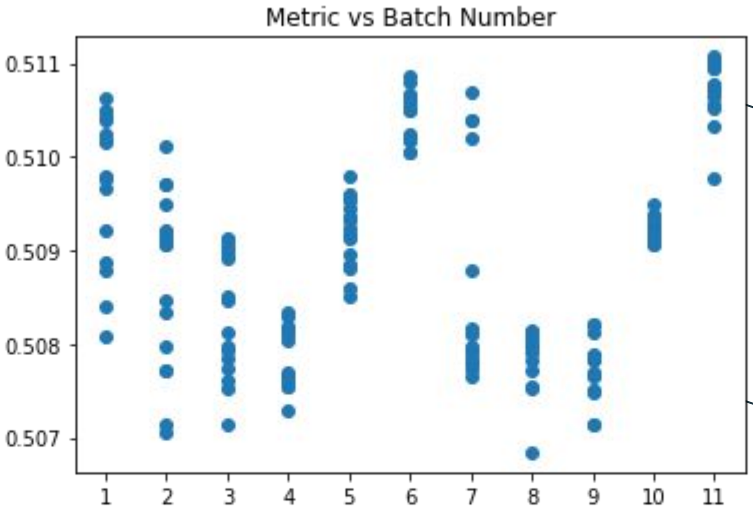
Measuring progress



DCG
@
20

Batch Number

Measuring progress



DCG
@
20

Batch Number

Exploitation

Exploration

Exploring results

"Best" model

Parameter	Value
b (BM25)	Low
k (BM25)	Medium
Match Boost	High
Phrase Boost	High
Stemmed Match Boost	Low
Stemmed Phrase Boost	High

Exploring results

Parameter	Value
b (BM25)	Low
k (BM25)	Medium
Match Boost	High
Phrase Boost	High
Stemmed Match Boost	Low
Stemmed Phrase Boost	High

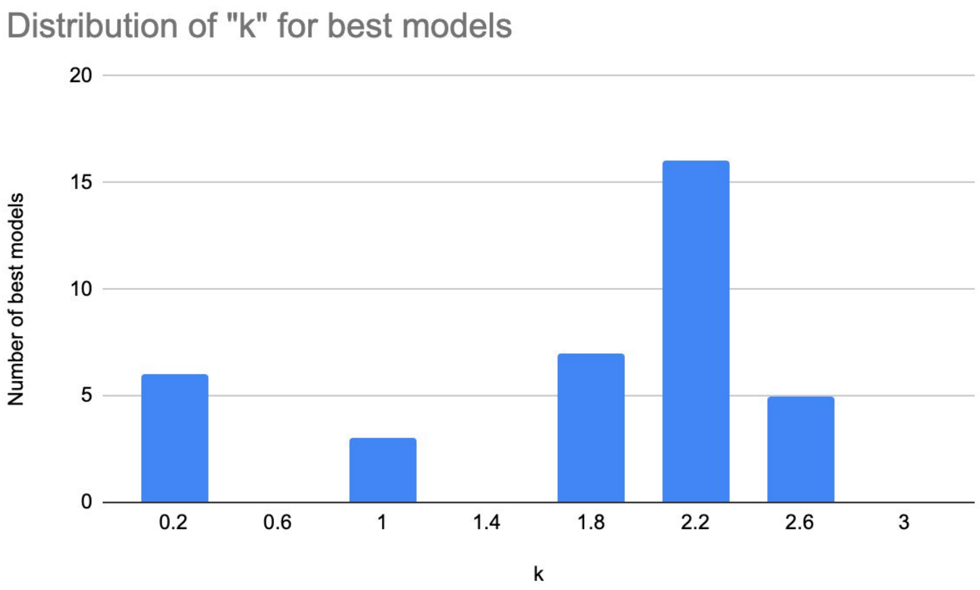
Less emphasis
on the lengths of titles

Exploring results

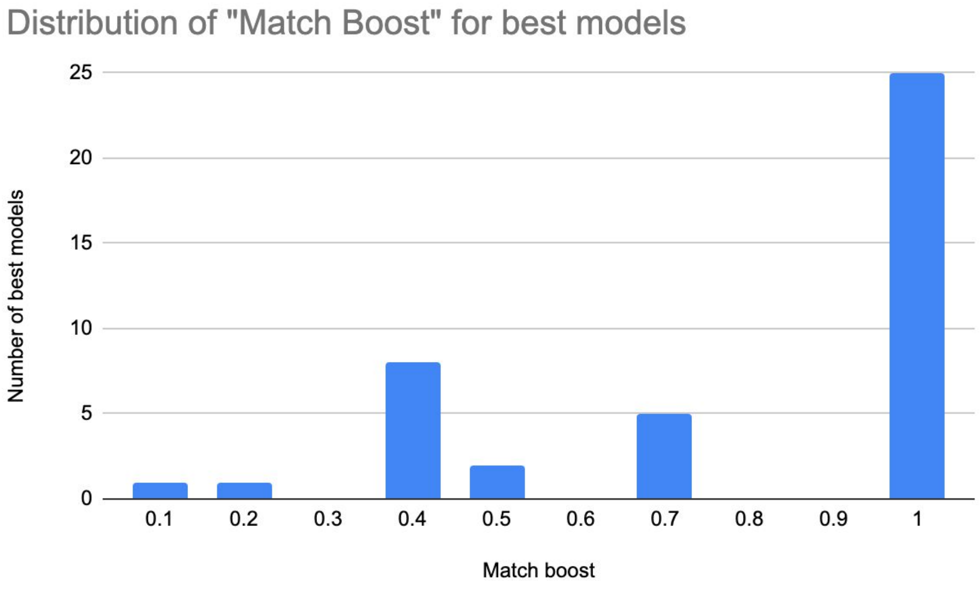
Parameter	Value
b (BM25)	Low
k (BM25)	Medium
Match Boost	High
Phrase Boost	High
Stemmed Match Boost	Low
Stemmed Phrase Boost	High

Phrase matches have strong boosts

Second lesson: look at the distribution



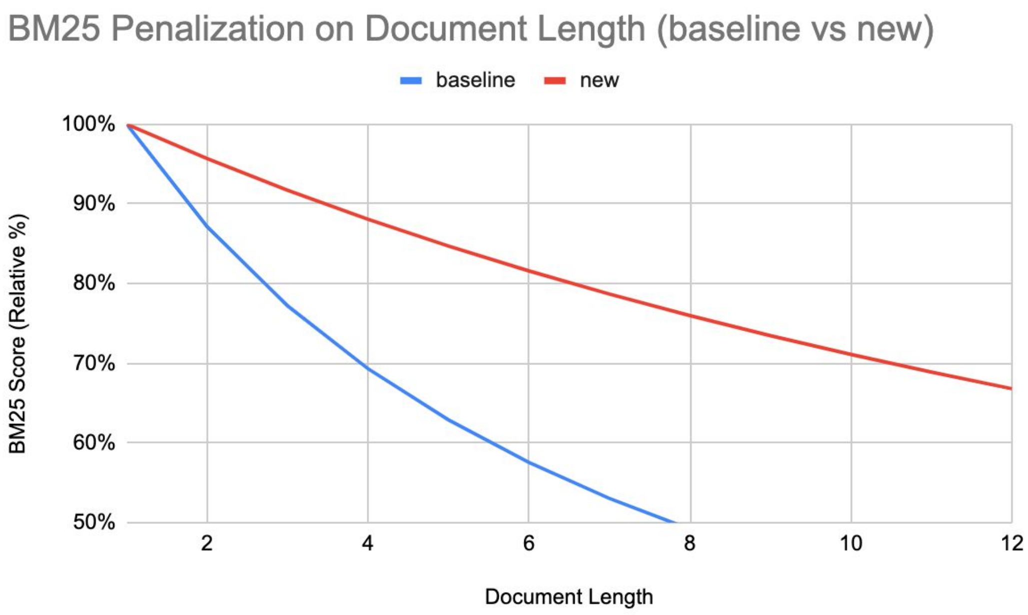
Second lesson: look at the distribution



Summary of changes

Parameter	**Change**	**Effect**
b (BM25)	High → Low	Less penalization on document lengths
Stemmed Match Boost	High → Low	Stemmed matches come lower in search results. Still boosts recall
Stemmed Phrase Boost	Low → High	Doubling-down on high-precision phrases

BM25: before and after



Evaluation

downsample_name	ds_num_docs	ds_num_queries	test_dcg_at_20	control_dcg_at_20	dcg_test_minus_control	change_in_dcg_relative	significance
grid_no_noise_bigger	36389	1440	0.51050	0.51025	0.000	0.0%	False
brokers_prod_small	151771	215	0.22815	0.22047	0.008	3.5%	True
manual_with_noise	123871	1879	0.41519	0.39553	0.020	5.0%	True

Evaluation

downsample_name	ds_num_docs	ds_num_queries	test_dcg_at_20	control_dcg_at_20	dcg_test_minus_control	change_in_dcg_relative	significance
grid_no_noise_bigger	36389	1440	0.51050	0.51025	0.000	0.0%	False
brokers_prod_small	151771	215	0.22815	0.22047	0.008	3.5%	True
manual_with_noise	123871	1879	0.41519	0.39553	0.020	5.0%	True

Evaluation

downsample_name	ds_num_docs	ds_num_queries	test_dcg_at_20	control_dcg_at_20	dcg_test_minus_control	change_in_dcg_relative	significance
grid_no_noise_bigger	36389	1440	0.51050	0.51025	0.000	0.0%	False
brokers_prod_small	151771	215	0.22815	0.22047	0.008	3.5%	True
manual_with_noise	123871	1879	0.41519	0.39553	0.020	5.0%	True

Takeaways

1. Your search engine might have the right strategies

Takeaways

1. Your search engine might have the right strategies
   - ✓ Sometimes all that's needed is a re-balance of signals

Takeaways

1. Your search engine might have the right strategies
   - ✓ Sometimes all that's needed is a re-balance of signals

2. Parameter optimization is both a science and an art

Takeaways

1. Your search engine might have the right strategies
   - ✓ Sometimes all that’s needed is a re-balance of signals

2. Parameter optimization is both a science and an art
   - ✓ Evaluate on the right data
   - ✓ Optimize the right metric
   - ✓ Share parameters where possible
   - ✓ Constrain parameters using your judgment
   - ✓ Explore the distribution of good models

Future work

1. Better understanding and tuning the optimizer

2. Comparing to baselines like random grid search and LTR / LTB

Future work

1. Better understanding and tuning the optimizer

2. Comparing to baselines like random grid search and LTR / LTB

3. Lots more!

- The value of theta for controlling the level of exploration
- Enforcing randomness across parallel jobs (otherwise they perform similar explorations)
- The best way to initialize the optimization with random samples
- Which regression to use, which kernels to use

So when do you use LTR instead?

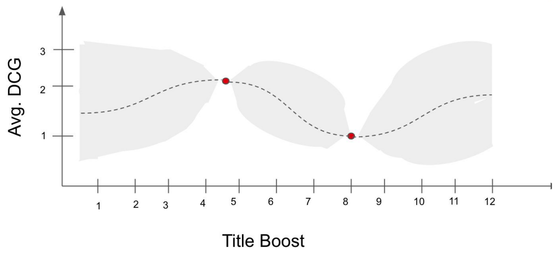
Pro: Arbitrary functional form of the features

Con: More infrastructure and (re)training

Elasticsearch Learning to Rank GO

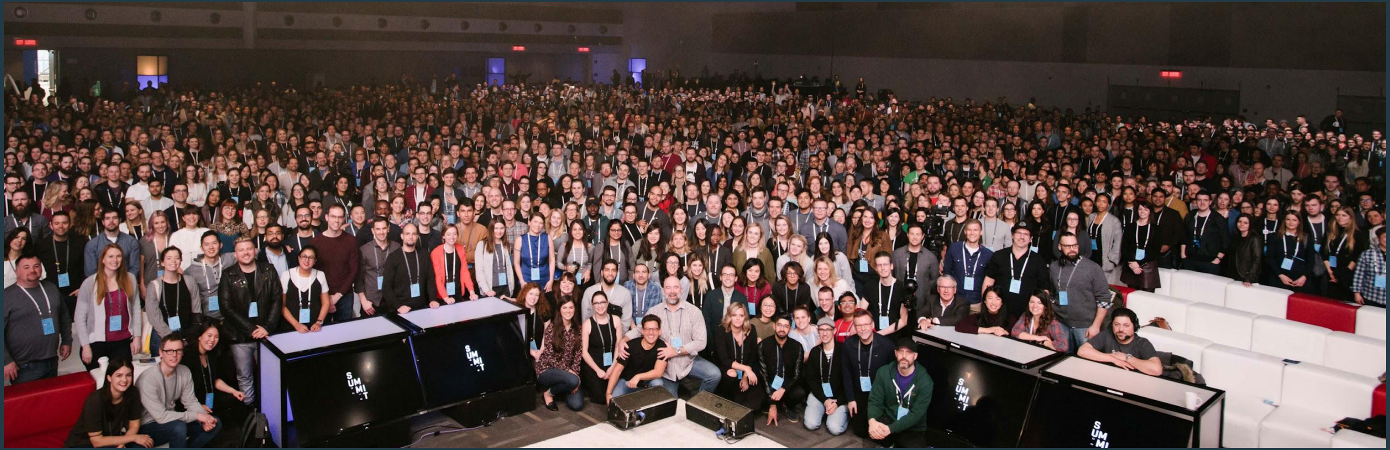
Con: Static functional form (your Elasticsearch query)

Pro: Simple infrastructure



We're growing

shopify.com/careers



Our 3,000+ employees at our annual Summit

Thank You!