Feb 18, 2020

Search Relevance Engineering:
Query Understanding & Ranking

Dipak Parmar
Principal Engineer

Shipt

Agenda	
1	Introduction
2	Search Relevance Engineering - Text analysis - Query understanding - Ranking
3	Overview of Search Platform @ Shipt - Shipt search platform technologies stack - Search indexing pipeline - Search services - Elasticsearch clusters - Lessons learned and best practices
4	Q & A

About Shipt

- Shipt launched in the summer of 2014 in the heart of the Magic City, Birmingham, AL.
- Shipt connects members to fresh groceries and everyday essentials. Saving time, fuel and headspace, next-hour, same day grocery delivery is quickly becoming an everyday necessity for people looking for an extra few hours and intentional food choices.

5,000+
cities covered

80%
of households covered nationwide

100+
Retailers

400K
shoppers

$175M+
and counting in shopper tips

70
NPS Score

CVS

HIBBETT SPORTS

COSTCO WHOLESALE

BEST BUY

BED BATH & BEYOND

Party City

TARGET

FINE WINE ABC & SPIRITS

Beyond Grocery...

Office DEPOT

Sur La Table

BevMo!

buy buy BABY

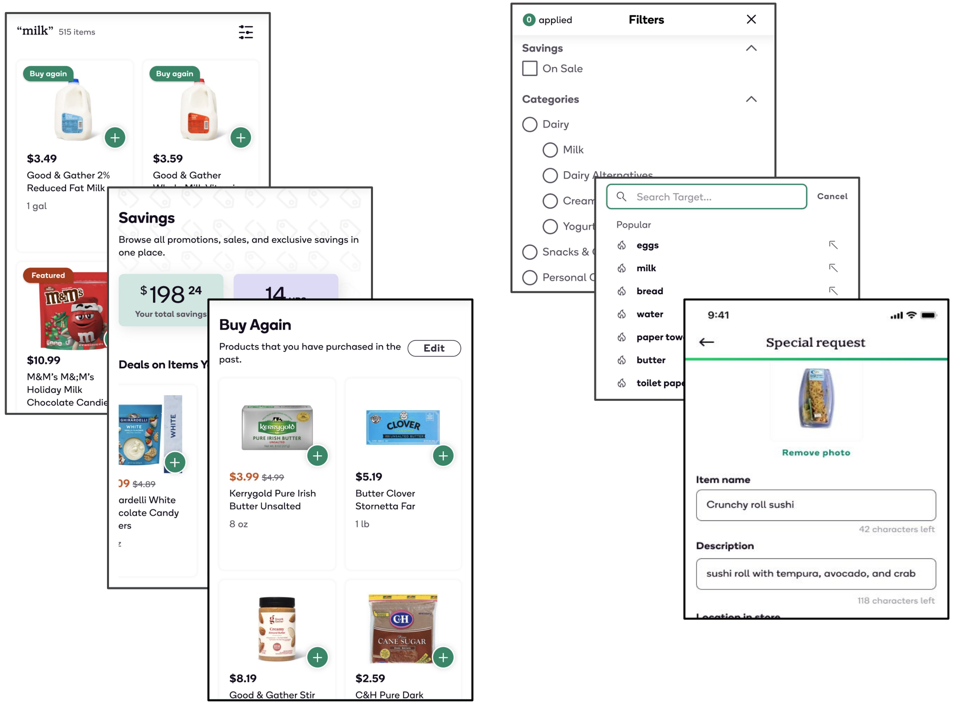
Total Wine & MORE

petco

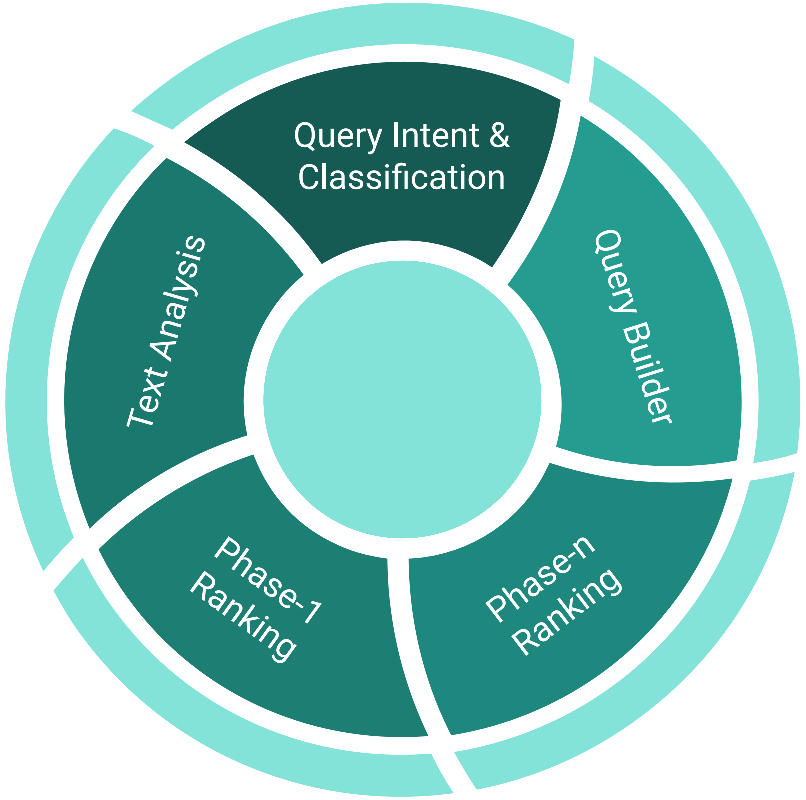
meijer

Search @ Shipt

- 100+ retailers, 1000+ store locations
- 50% of searches come from just top 1000 unique searches
- 30% of all purchases are repeat buys
- Search keywords distribution
  - 20% single token
  - 50% two tokens
  - 30% three+ tokens



Search Relevance Engineering



Text Analysis & Query Understanding

Text Analysis

The QUICK brown foxes jumped over the dog!

[“quick”, “fast”, “brown”, “fox”, “jump”, “over”, dog”]

Ngrams [“q”, “qu”, “qui”, “quic”, “quick”, ..]

Text Analysis

strawberry ice cream

↓

Stemming: [“starwberii”, “ic”, “cream”]

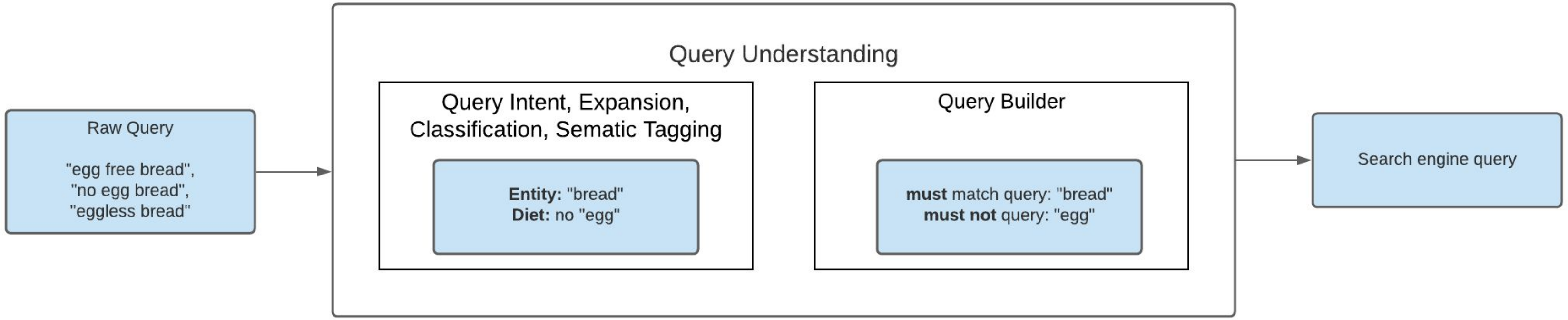
Shingles: [“strawberry ice”, “ice cream”]

Shingle concat: [“strawberryice”, “icecream”]

Query Understanding

As per Wikipedia

"Query understanding is the process of inferring the intent of a search engine user by extracting semantic meaning from the searcher's keywords. Query understanding methods generally take place before the search engine retrieves and ranks results"



Query Understanding

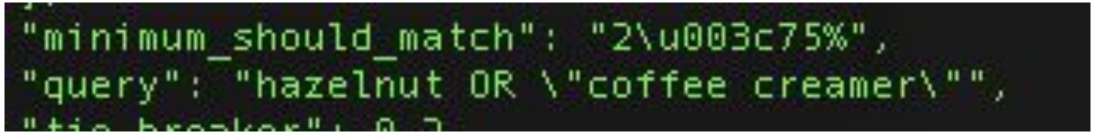
01	good & gather veggie pizza	● Brand: good & gather ● Dietary preference: veggie ● Entity: “pizza”
02	1 gal 2% milk 2% gallon milk	● Size: 1 gallon ● Modifier: 2% ● Entity: “milk”
03	16 oz sour cream sourcream 16oz	● Size: 16 oz ● Entity: “sour cream” (compound word)
04	half and half coffee pods ½ & ½ coffee pods	● Modifier: “half & half” ● Entity: “coffee pods” (compound word)
05	fresh red onion	● Entity: “onion” ● Modifier: “red” ● Optional: “fresh”

Query Understanding

- Detecting intent with classifiers
    - Size, brands, nutritional/dietary keywords
    - Identify color, pattern, modifiers, flavors, taste, shape, etc.
    - Optional keywords
    - Negative intent: “egg free pizza”
- Replacements, query expansions, synonyms, hypernyms
- Linguistic analysis
- ML, AI, NLP techniques

Query Builder

- Build query based on intent
  - Partial match
  - Phrase search
  - Word drop
  - Minimum match criteria
  - Fuzzy search
  - Filtering
  - Boost
- Ranking based on intent



```
"minimum_should_match": "2\u003c75%",
"query": "hazelnut OR \"coffee creamer\"",
```

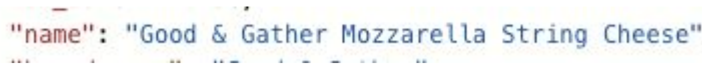
Ranking

Ranking

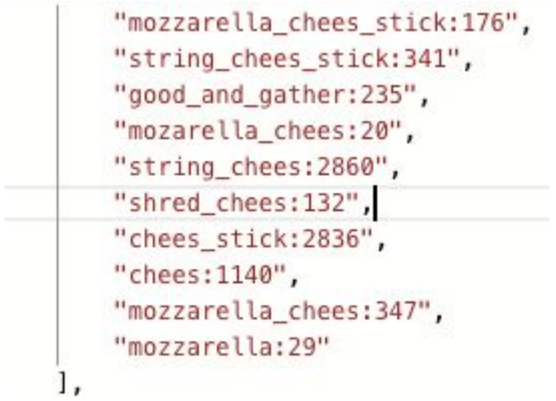
- 5 phases of ranking
  - Phase-1: Base query ranking based on query intent (BM25)
  - Phase-2: Boosting (BM25)
  - Phase-3: Custom ranking with other signals (script score or custom plug-in)
  - Phase-4: Re-rank top x documents (LTR / vector similarity / other signals)
  - Phase-5: Post process ranking (outside search engine)

Ranking

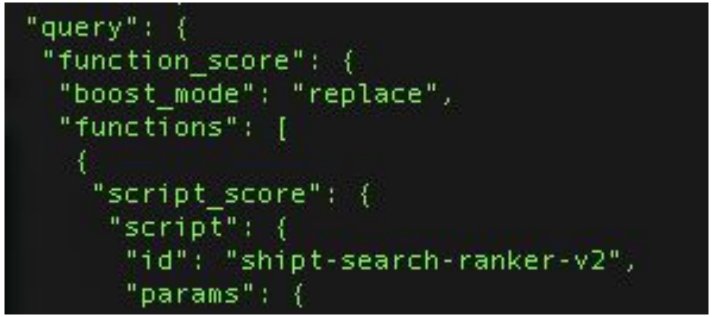
- Ingest external signals
  - Offline process to generate popularity scores at product level
  - Signals at neighborhood level can be generated
  - Product to converted keywords mapping can be maintained
- Use external signals along with BM25 score in deriving final score
- “Painless” script in Elasticsearch can be used for ranking



```
"name": "Good & Gather Mozzarella String Cheese"
```



```
        "mozzarella_chees_stick:176",
        "string_chees_stick:341",
        "good_and_gather:235",
        "mozarella_chees:20",
        "string_chees:2860",
        "shred_chees:132",
        "chees_stick:2836",
        "chees:1140",
        "mozzarella_chees:347",
        "mozzarella:29"
],
```



```
"query": {
 "function_score": {
  "boost_mode": "replace",
  "functions": [
   {
    "script_score": {
     "script": {
      "id": "shipt-search-ranker-v2",
      "params": {
```

Ranking

Final score = (BM25 score * *weight*) +

(signal-1 * *multiplier* * *weight*) +

(signal-2 * *multiplier* * *weight*) + …

Example:

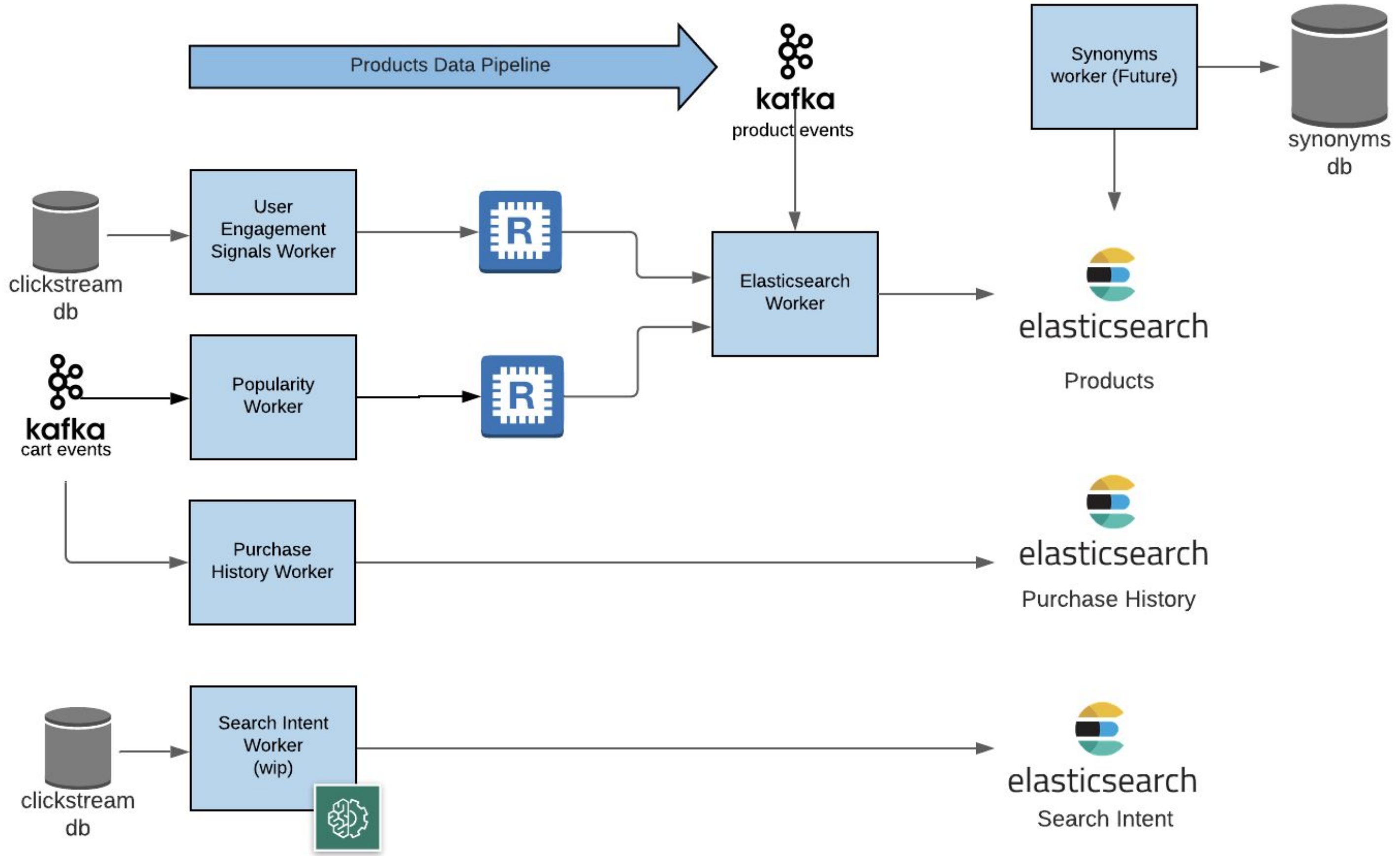
Final score = (BM25 score * *0.50*) + (log(popularity) * *100* * *0.30*) + (geographic trend * *0.10*) + ..

Overview of Search Platform @ Shipt

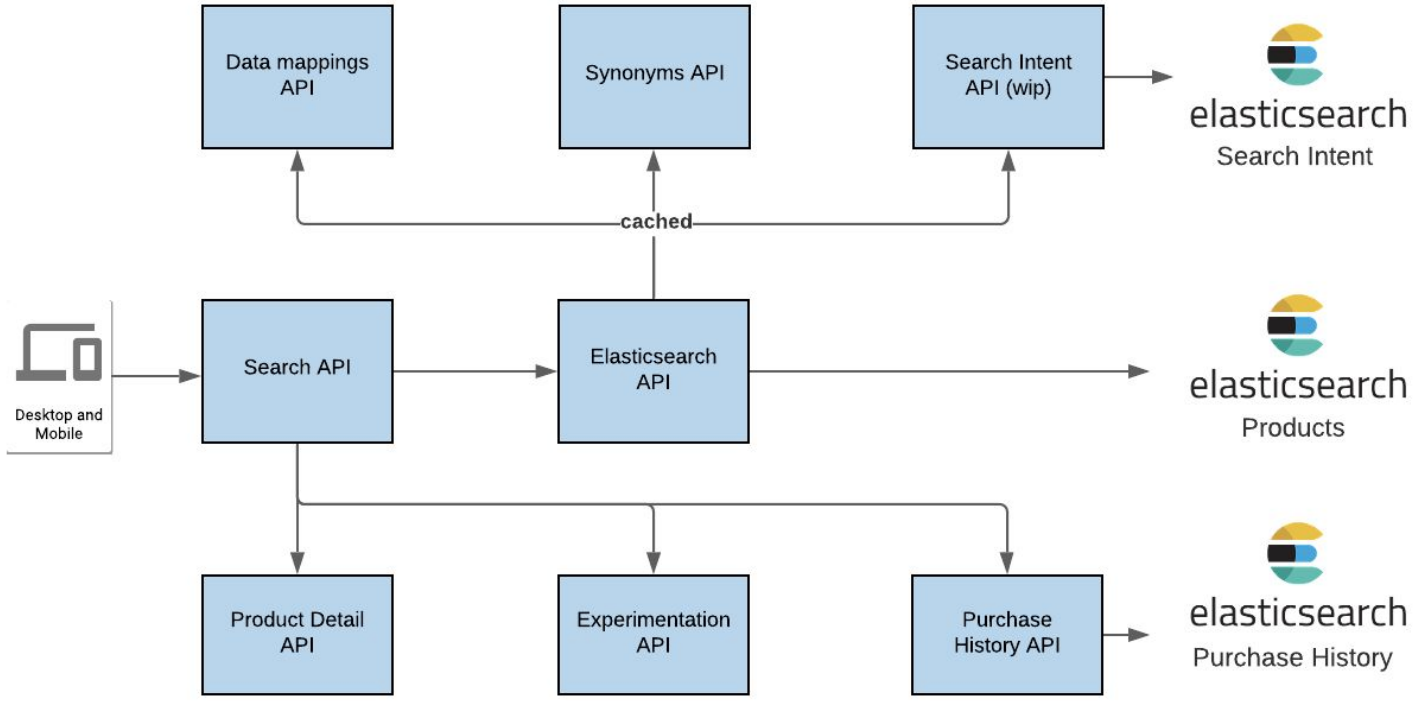
Shipt Search Platform Technologies Stack

- Language:
  - Golang
- Datastores:
  - Elasticsearch, Redis, PostgreSQL
- Messaging:
  - Kafka
- Infrastructure / Observability:
  - Docker, Kubernetes, Rollbar, Grafana
- Cloud hosting:
  - AWS, GCP, ElasticCloud, Confluent

Search Indexing Pipeline @ Shipt



Search Services @ Shipt



Elasticsearch clusters

- Write heavy indexing pipeline
- Multiple groups of clusters based on search traffic, indexing traffic and number of documents
- High availability, multiple replicas
- Single query approach, p99 of 70ms

Lessons learned, best practices

- Single-Responsibility Principle. Don’t do too many things in one micro-service
- Separation of concerns. Modularize and re-use, avoid duplicate logic
- Don’t try to solve everything with search engine!
- Don’t try to solve everything with machine learning!
- High refresh interval setting in Elasticsearch improves response time p99
- Start with optimizing head keywords followed by torso and tail keywords
- Act fast, evaluate a/b test fast, learn from a/b tests, fail fast
- Parent-child relationship in Elasticsearch could degrade performance
- Duplication of data is just fine in most cases in search engine
- Excessive use of Fuzzy search could lead to performance overhead. It’s better not to apply fuzzy search for already correctly spelled keywords

Questions

Thank you!

Special thanks to OpenSource Connections for organizing this event!