ENTER ELASTIC NINJA

Product Search with Elastic Stack, Embeddings, and Learning-to-Rank

Charin Polpanumas
Lead Data Scientist @ Central

LEAD DATA SCIENTST @ CENTRAL

Provides search, ranking, recommendation, customer targeting in online and offline retail across 6+ business units

DATA SCIENTIST @ BRIDGEASIA

Implements abnormality screening model for chest-xrays at one of Thailand's largest public hospital chain, cutting radiologists' time in half

DATA SCIENTIST @ LAZADA

Implements ads optimization and fraud detection models, saving XM USD annually in marketing budget

Central Retail Corporation

Fashion, food, electronics, office supplies and so on

CENTRALRETAIL

CENTRAL

ROBINSON

RINASCENTE

cmg

Power Buy

ไทวัสดุ
THAI WATSADU

baan
& BEYOND

SUPERSPORTS

GO!

NK

LANCHI MART

KIS

AUTO 1

CENTRAL FOOD HALL

Tops market

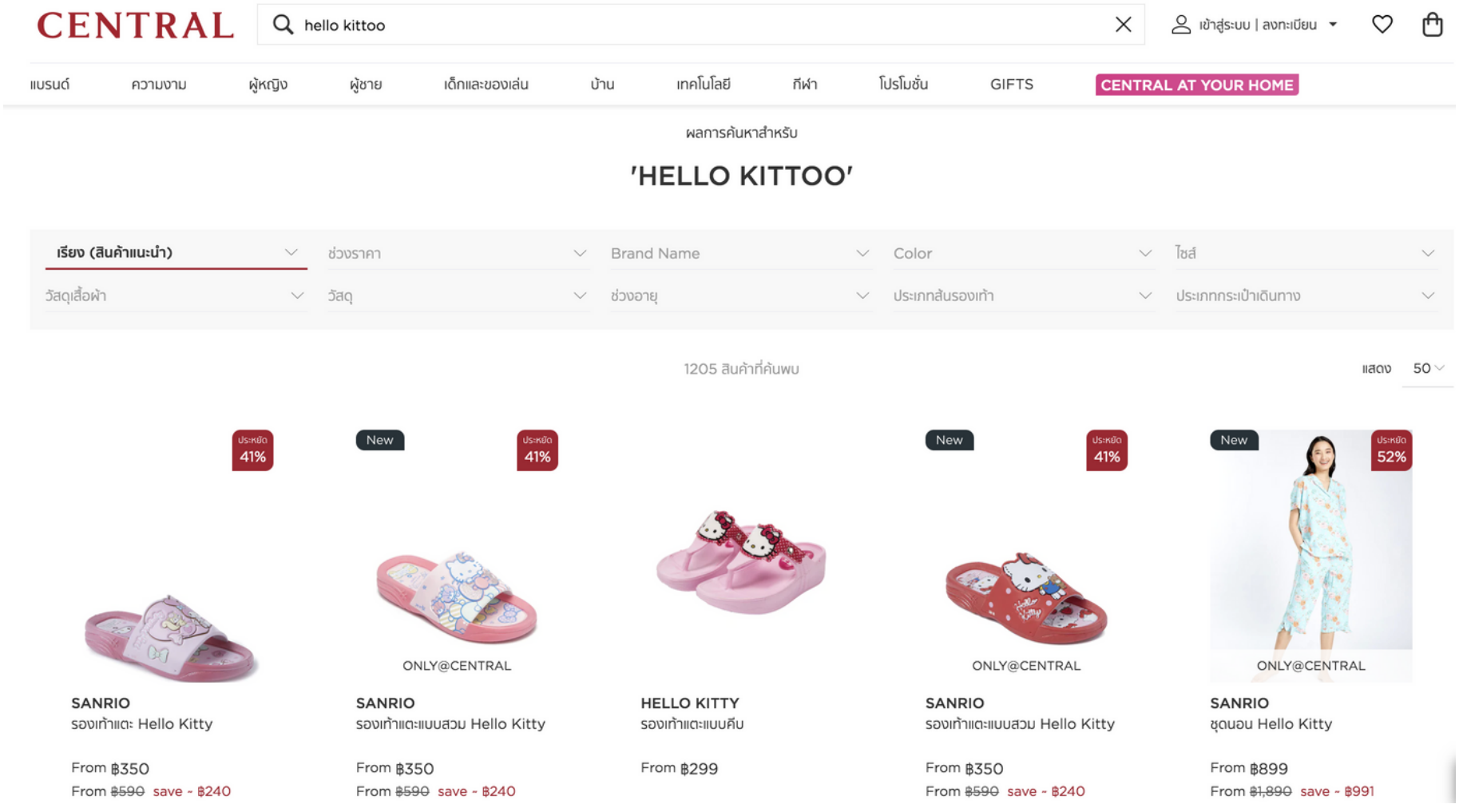
Tops daily

FamilyMart

薬

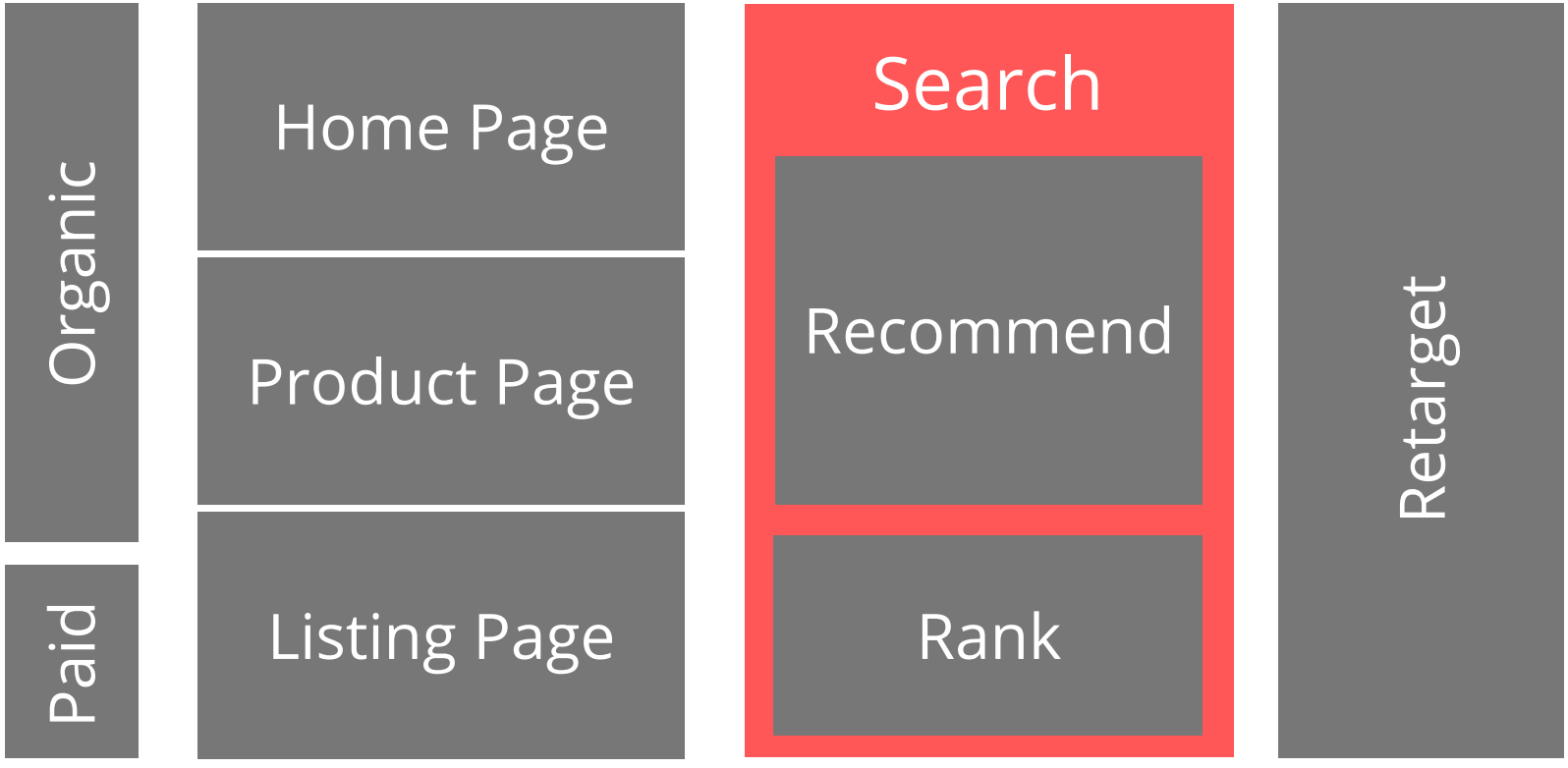
Product search in one sentence

Give a search term, show the products that user will most likely buy



Why people care about product search

Search is 10-20% of traffic but often has 5-10x conversion rate vs average



Layers of product search

What we will specifically cover today

Cold-start logic

Multi-armed bandits and maybe reinforcement learning to shuffle

Re-ranking Model

Models that re-rank full-text search results to optimize for CTR/CR

Full-text Search

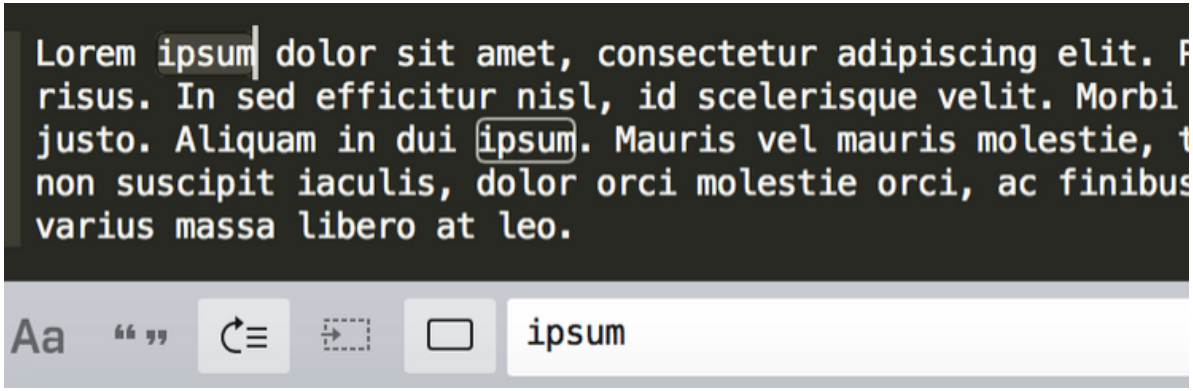
Match products purely based on texts of names, brand names, categories and other possible metadata

Part I

Full-text Search

RECALL THE MOST RELEVANT PRODUCTS BASED ON TEXTS ALONE

Github: https://github.com/cstorm125/esninja



Why use Elasticsearch instead of regex or sklearn

Distributed, open source search and analytics engine for all types of data

Expression

```
/([A-Z])\w+/g
```

Text Tests NEW

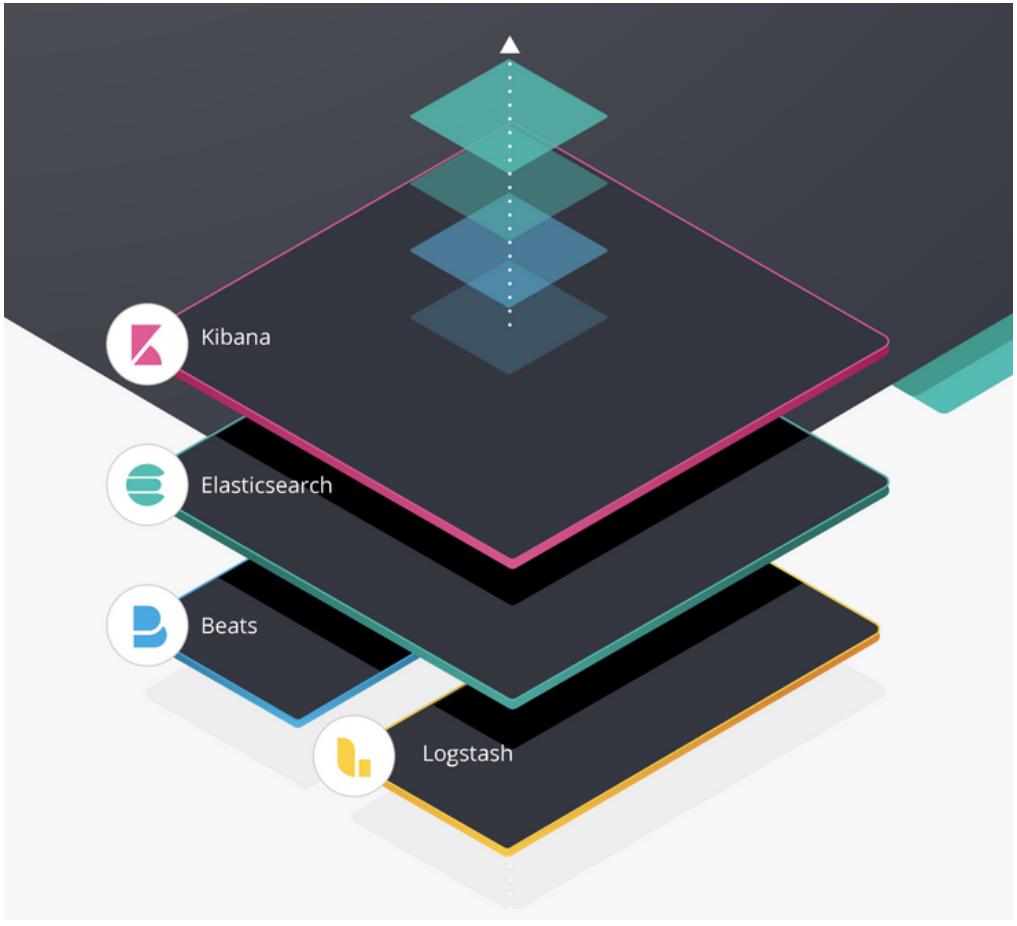
```
RegExr was created by gskinner.c

Edit the Expression & Text to se
are supported. Validate your exp

The side bar includes a Cheatshe
create or favorite in My Pattern

Explore results with the Tools b
expression in plain English.
```

```
from sklearn.feature_extraction.text import
corpus = [
    'This is the first document.',
    'This document is the second document.'
    'And this is the third one.',
    'Is this the first document?',
]
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(corpus)
print(vectorizer.get_feature_names())
nd', 'document', 'first', 'is', 'one', 'seco
print(X.shape)
9)
```

- Natively Distributed; built on Apache Lucene
- Fast
- Scalable
- Resilient
- Rich in features
- Platform independent with REST APIs
- Excellent documentation and ease of use
- Largest community of any full-text search software

What is the Elastic Stack

Logstash and Beat, Elasticsearch, Kibana



The Elastic stack is obviously useful more much more than full-text product search but we will use only these components for our purpose today:

- Logstash - ingest data (csv, json, ...)
- Elasticsearch - full-text search engine
- Kibana - console and visualization

See more at https://www.elastic.co/what-is/elk-stack

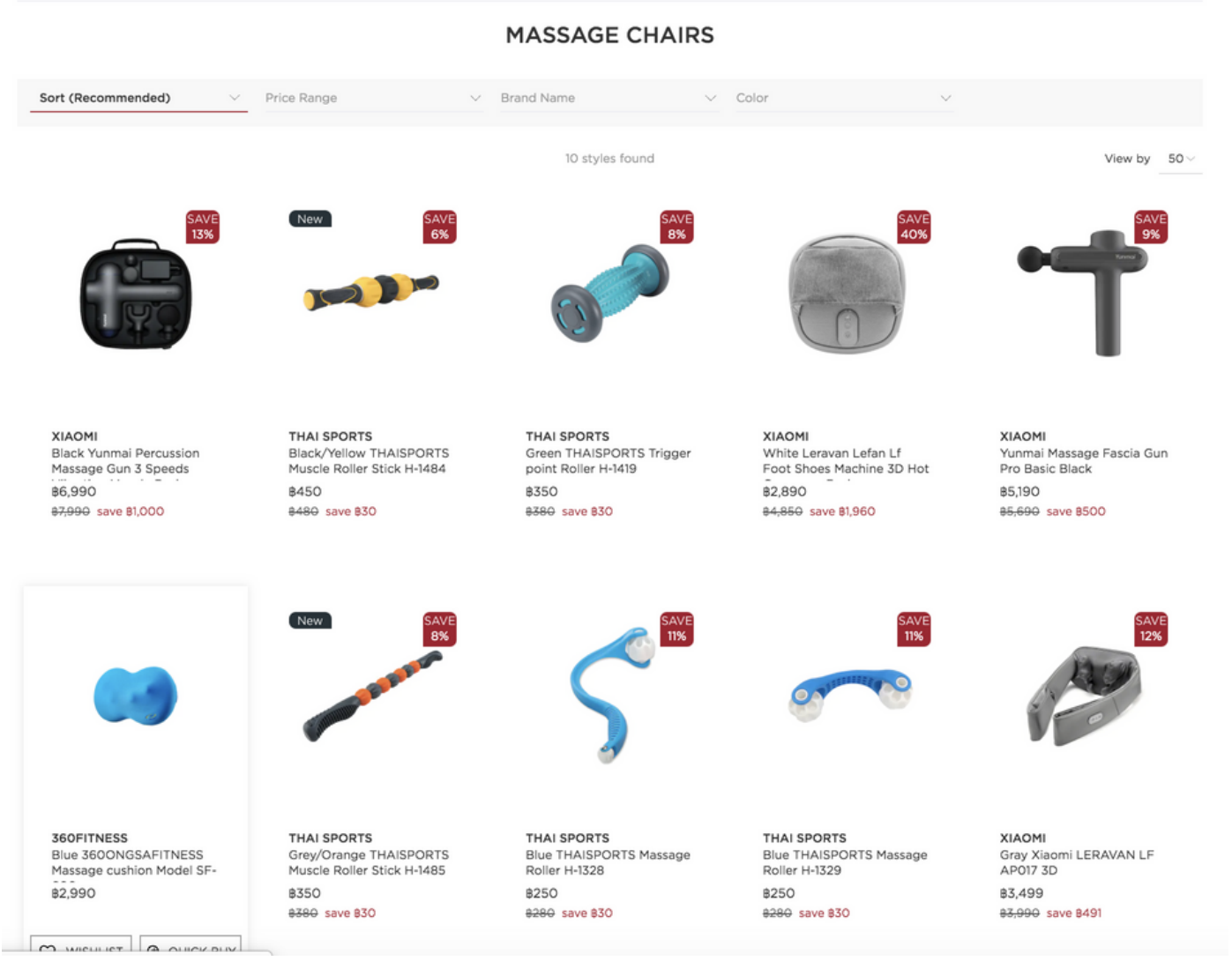
Always perform quality check before ingestion

Example from TOPS Supermarket; see tops_sample_qc.ipynb

	name_en	brand_en	category_en	subcategory_en	class_en	long_desc_en
0	Chicken Pie(B	MY CHOICE	Fresh Food & Bakery	Bakery	Danish & Puff Pastry	พายไส้ไก่ หอม อร่อย เ...
1	Heinz Pickled Onions 440g.	HEINZ	Fruit & Vegetables	Preserved Fruit & Ve...	Preserved Vegetables	heinz pickled onions...
2	PC Tuna Salmon Flake 70g.	PC TUNA	Pantry & Ingredients	Canned Food	Instant Meals	nan
3	Malee Pasteurized Mandarin Orange Ju...	MALEE	nan	nan	nan	nan
4	Frontline Plus Kills Fleas And Tick For B...	FRONTLINE	Household & Pet	Pet Care	Pet Health Care	nan
5	ST. Ives Even and Bright Pink Lemon a...	ST.IVES	Health & Beauty Care	Facial Care	Facial Cleanser	nan
6	Sengheng Fresh Tofu Skin 100g.	SENGHENG	Fresh Food & Bakery	Tofu & Fresh Noodle	Tofu	เส่งเฮงฟองเต้าหู้ คุณภ...
7	Brands Essence of Chicken Original 42...	BRANDS	Beverages	Health Tonics	Essence of Chicken	แบรนด์ซุปไก่สกัดรสต้น...
8	Tops Brand Fish Maw 50g.	TOPS	Pantry & Ingredients	Dried Ingredients	Dried Soup Ingredients	nan
9	My Choice Thai Seasoned & Rolled Cut...	MY CHOICE THAI	Meat & Seafood	Marinated Meat	Processed Seafood	มายช้อยส์ไทยปลาหมึ...
10	Cathy Doll Tsum Tsum Oil Control Pact ...	CATHY DOLL	International Products	KOREA	Health & Beauty Care	nan
11	Dr. Hiratake Mushroom 150g.	DOCTOR VEGETAB...	Fruit & Vegetables	Vegetables	Mushroom	ชิตาเกะ หรือเห็ดหอม เ...
12	Healthy Boy Black Soy Sauce 400g.	HEALTHY BOY	Pantry & Ingredients	Sauces	Soy Sauce	nan
13	Tops Spicy Stir Fried Vegetarian Protein...	TOPS	Fresh Food & Bakery	Frozen Food	Frozen Meals	ท็อปส์ข้าวกระเพราเจ ผ...
14	Attack Easy Quick Happy Love Powder...	ATTACK	Household & Pet	Laundry	Powder Detergent	แอทแทคอีซี่ควิกแฮปปี้เ...
15	Boontiang Toffee 200g.	BOONTIANG	Fruit & Vegetables	Preserved Fruit & Ve...	Fruit Desserts	บุญเที่ยงท๊อฟฟี่กะทิใส่...

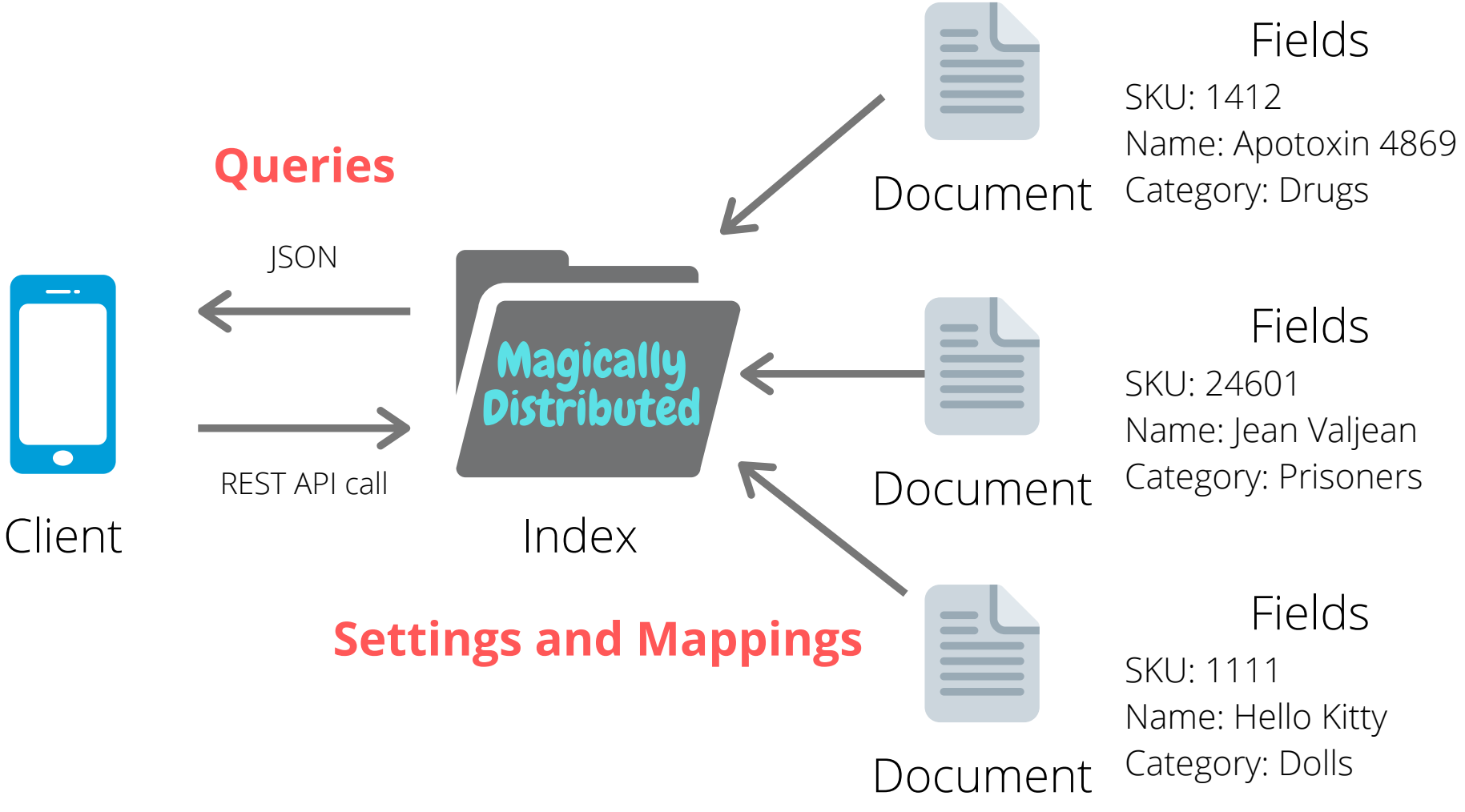
Content curation is extremely underrated

Sometimes people will just put random stuff in `massage chairs`



Elastic stack vocabulary

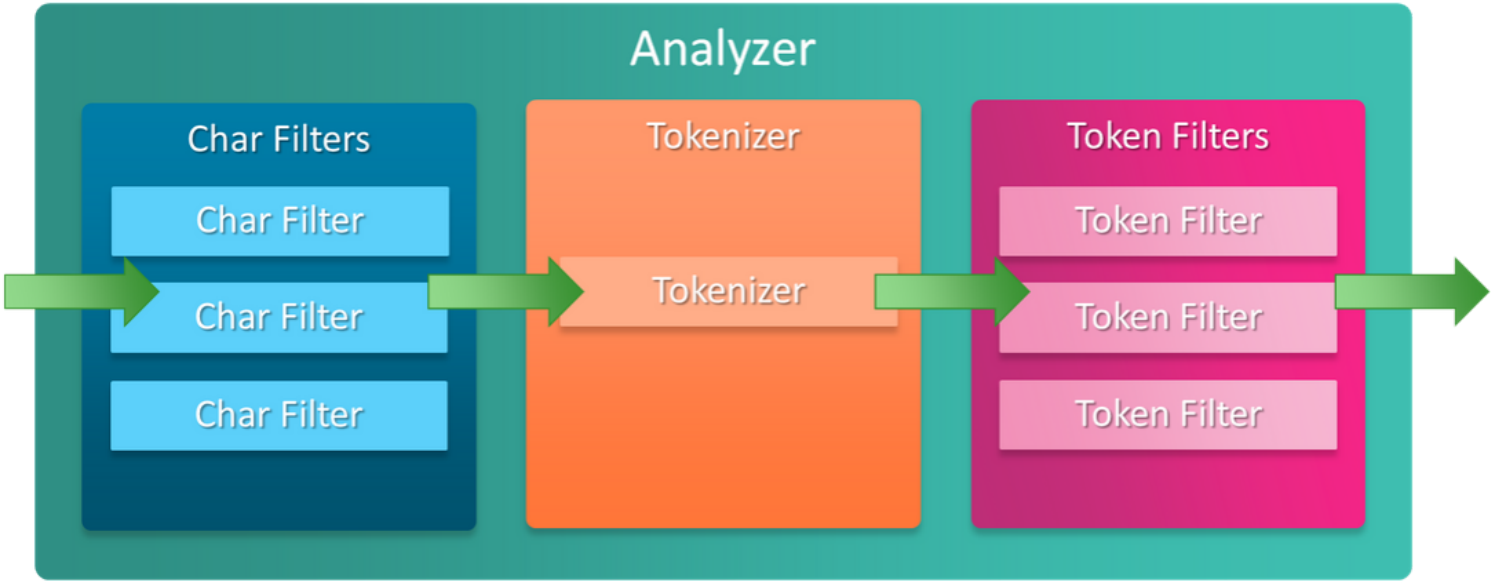
What you need to know to configure full-text search like a ninja



Settings

The tools that elasticsearch can use to process texts

- Analyzer - a text processing function comprise of:
  - Character Filter - process at character level
  - Tokenizer - tokenize characters to tokens
  - (Token) Filter - process at token level



See more at https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-custom-analyzer.html

What tokenizers to use

`thai`, `standard`, `icu_tokenizer`, or none of them

Built-in tokenizers of Elasticsearch

- **standard** - cannot tokenize Thai words at all
- **thai** - pretty bad e.g. `คลิปดำ` (black clips) to [`ค`,`ลิ`,`ป`,`ดำ`] (k, li, p, black)
- **icu_tokenizer** - mostly correct but does not take into account the needs for product search e.g. `หูฟัง` (earphones) to [`หู`,`ฟัง`] (ear, listen) so it will match also [`ต่าง`,`หู`] (earrings tokenized as different, ear)

The solution is to have either

- **Custom dict-based tokenizer plugin in Java** - faster, easier to maintain
- **Tokenize all Thai texts before indexing** with a special character such as pipe - adds another moving part to your system

Analyzer that almost always work

If you have no idea what to do, just start with this

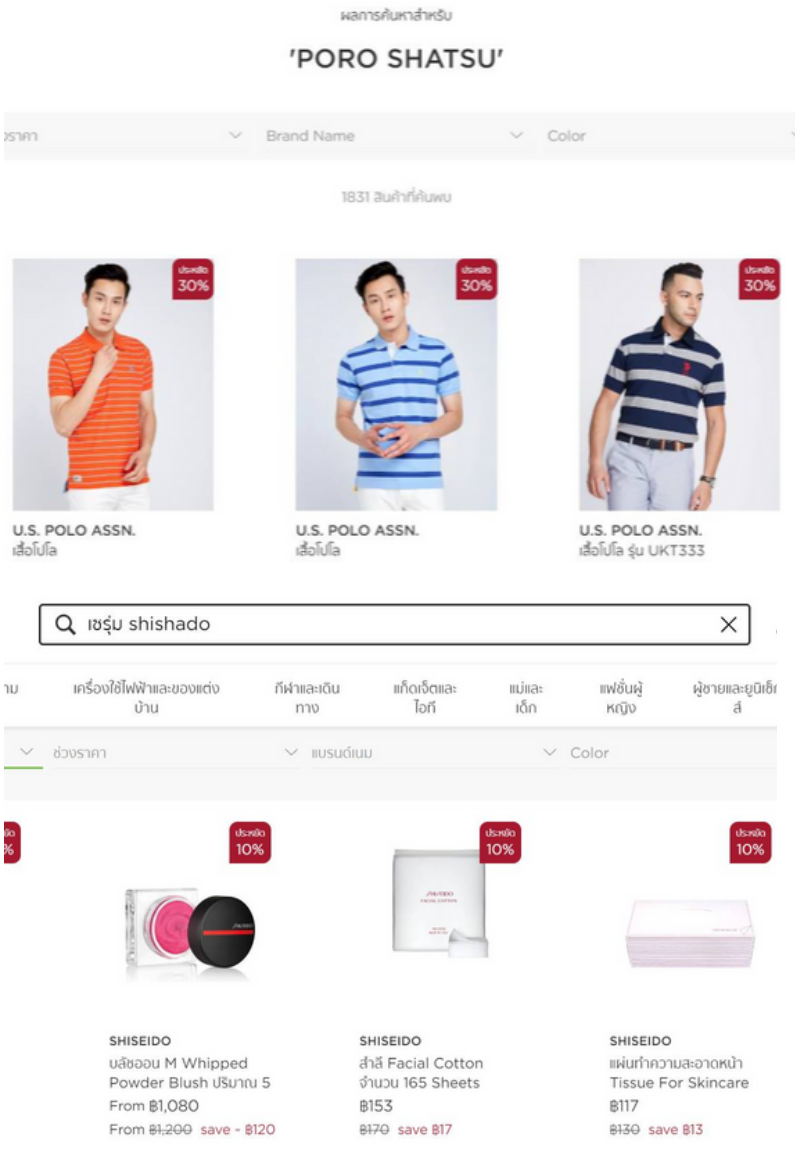
```
"word_stem_anl": {
  "char_filter": [
    "html_strip",
    "connector_flt",
    "numeric_delim_flt"
  ],
  "filter": [
    "lowercase",
    "asciifolding",
    "trim",
    "decimal_digit",
    "snowball_flt",
    "synonym_flt"
  ],
  "tokenizer": "icu_tokenizer"
},
```

A generic analyzer that works for most product search use cases have:

- Tokenization by ICU or better your own custom tokenizer
- `lowercase`
- `asciifolding` to handle things like `Lancôme`
- `trim` to get rid of surrounding spaces
- `decimal_digit` in case someone wants to use Thai numerals
- Snowball stemming to match words at their root forms
- Synonyms

List of things this usually solve

If you have no idea what to do, just start with this

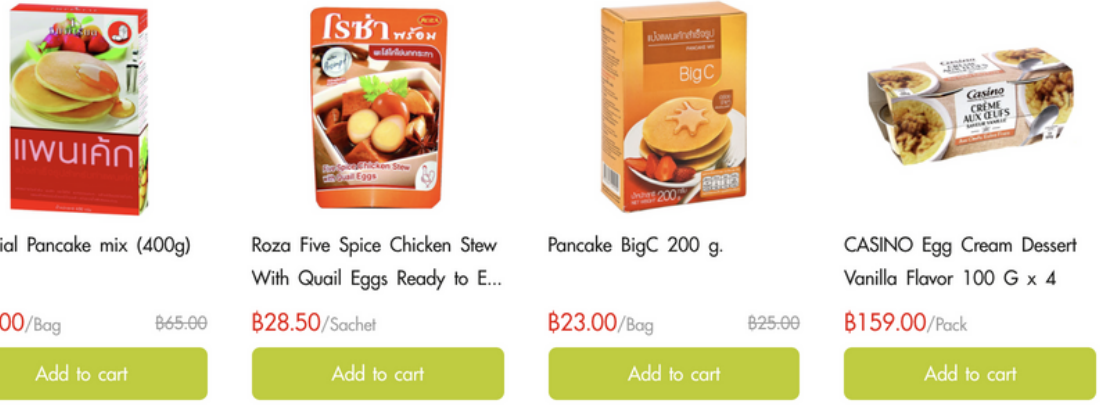


- misspell wrong character (dior, diox)
- missing character (dior, dio)
- extra character (lipstick, lipsticky)
- hyphens and co (baby-g, babyg, baby g; l'oreal, loreal)
- regular plural (bags, bag; classes, class)
- irregular plural (shelves, shelf) partial
- match for sku number (CDS1234, 1234)
- linguistic synonyms (rat, mouse)
- use-case synonyms (coke, coca-cola)
- word delimiters (pm2.5 to [pm,2.5])

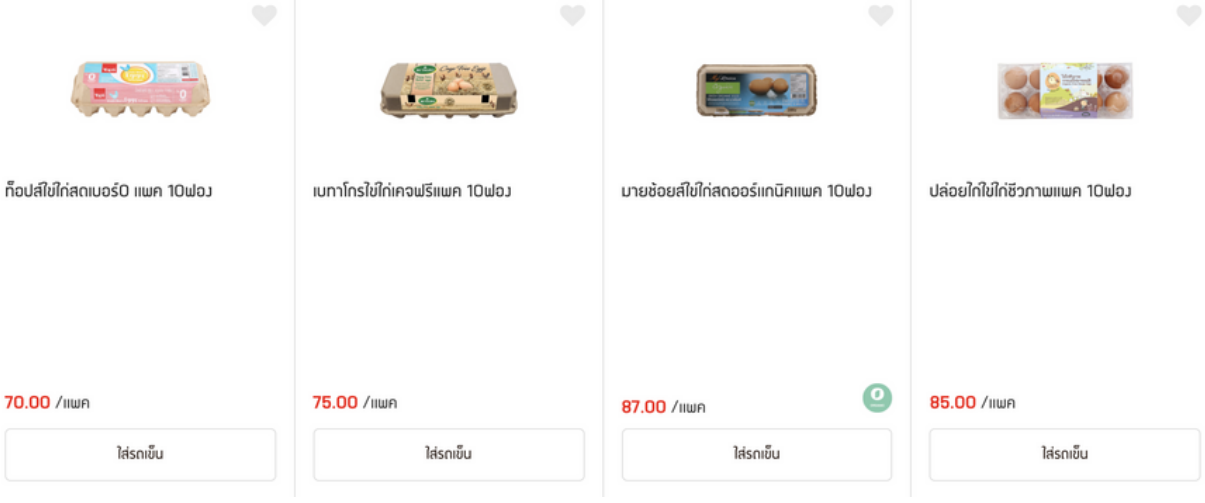
Queries

Why it is an extremely bad idea to do simple match query on product name

`eggs` on Big C



`eggs` on Tops



Multi-match query

Your best friend in e-commerce full-text search

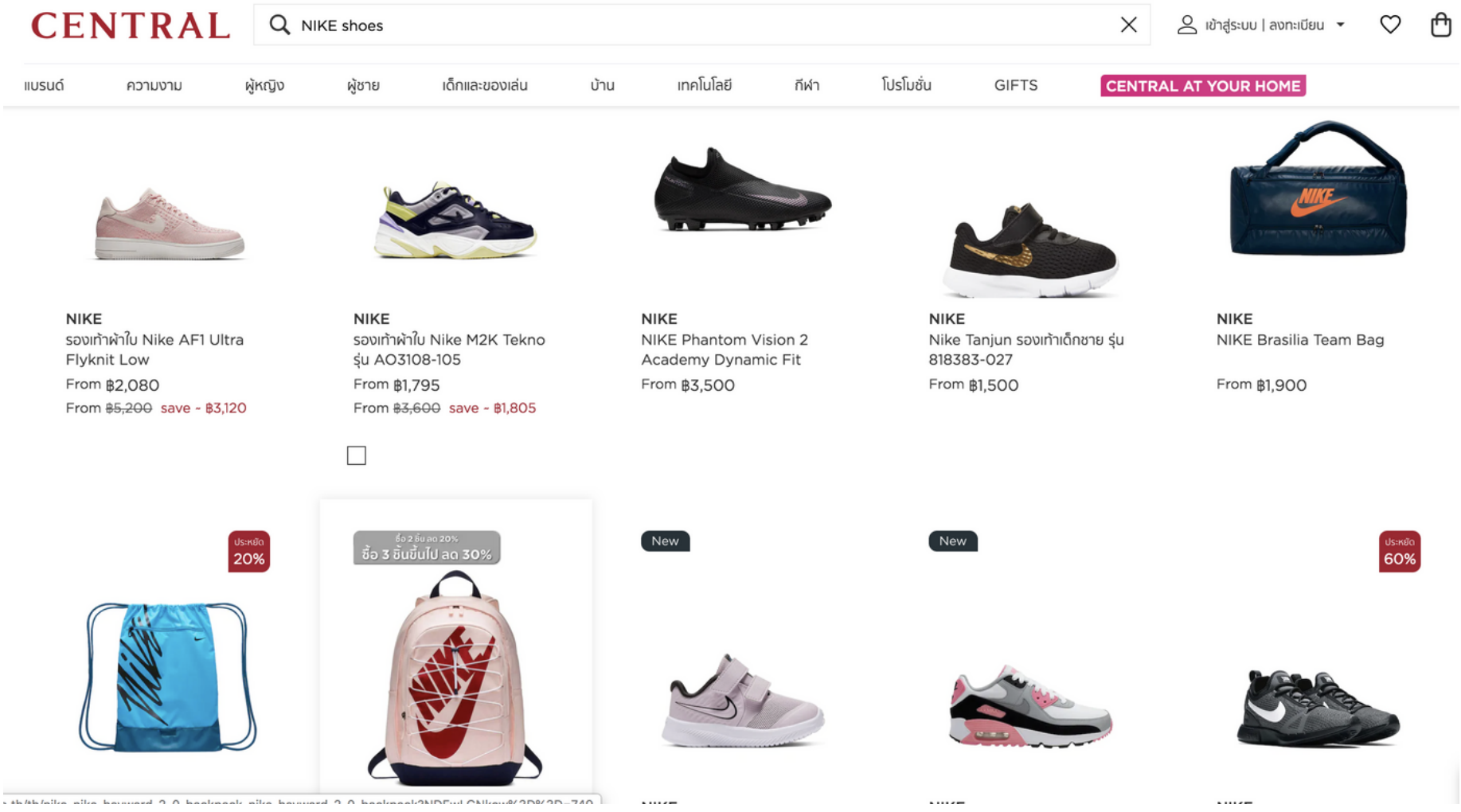
With multi-match queries, you can simultaneously search multiple fields at the same time. I prefer `most_fields` option because it **allows you to give each field a specific weight** and **allows fuzziness**. (not available in `cross_fields`)

`best_fields`	(**default**) Finds documents which match any field, but uses the `_score` from the best field. See `best_fields`.
`most_fields`	Finds documents which match any field and combines the `_score` from each field. See `most_fields`.
`cross_fields`	Treats fields with the same `analyzer` as though they were one big field. Looks for each word in **any** field. See `cross_fields`.

https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-multi-match-query.html

Multi-match query with operator `and`

When you want `nike shoes` to not return `nike shirts`



Putting everything together

Boolean queries and boosting to get desired hierarchy of search results

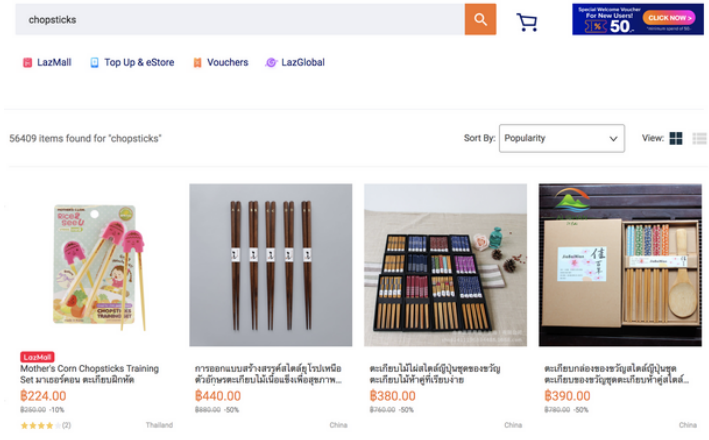
We can combine what we have learned so far with boosting function to allow the search to perform the following hierarchy:

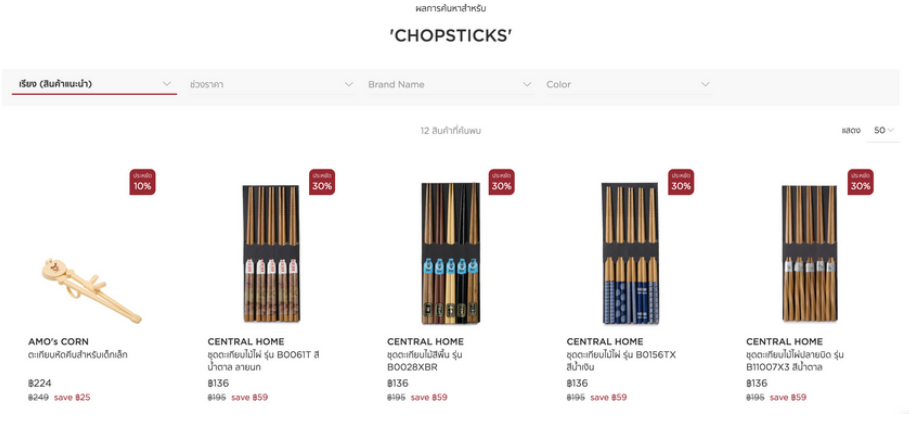
1. Match tokens without stemming
2. Match the entire tokens of the search term
3. Fuzzily match 1. and 2.
4. Go over 1., 2. and 3. but also allows partial match of some tokens of the search term e.g. `eggs` will match `fresh eggs` category

Keep in mind for full-text search

Finetuning elasticsearch mappings and queries is a balancing act

- **Relevance and speed** might be at odds e.g. four multi-matches might take twice the time to run compared to a simple multi-match; use the search profiler
- Generally we prefer **recall over precision** since we will re-rank the results with a model later anyways
- Content team needs to care; random stuff in `massage chairs`
- Assortment >>>>>> good configurations; Lazada has 500k results (only display 40k) for chopsticks vs 12 from Central





Part II

Re-ranking Models

SHOW THINGS PEOPLE WILL BUY

Most relevant texts != Most likely to be bought

People who search for `butter` might want margarine ("fake butter" in Thai)

เนย ✕ Search product name or brand

Login Register

Sale

save 10 baht

Allowrie Unsalted Butter 500g.

- Sale
- Today – 12 May 2020

~~220~~ 210.00 /pcs

Add to cart

Sale

save 6 baht

Orchid Butter Blend Spread Salted 227g.

- Sale
- Today – 12 May 2020

~~80~~ 74.00 /pcs

Add to cart

Sale

save 18 baht

Elle&Vire Unsalted Butter 200g.

- Sale
- Today – 12 May 2020

~~168~~ 150.00 /pcs

Add to cart

Sale

save 18 baht

Elle&Vire Salted French Butter 200g.

- Sale
- Today – 12 May 2020

~~168~~ 150.00 /pcs

Add to cart

Best Foods Margarine 150g.

31.50 /pcs

Add to cart

Allowrie Zero Soft Spreadable Butter Blend Salted 125g.

Sale

save 54 baht

Lurpak Lighter Spreadable Butter 250g.

- Sale
- Today – 12 May 2020

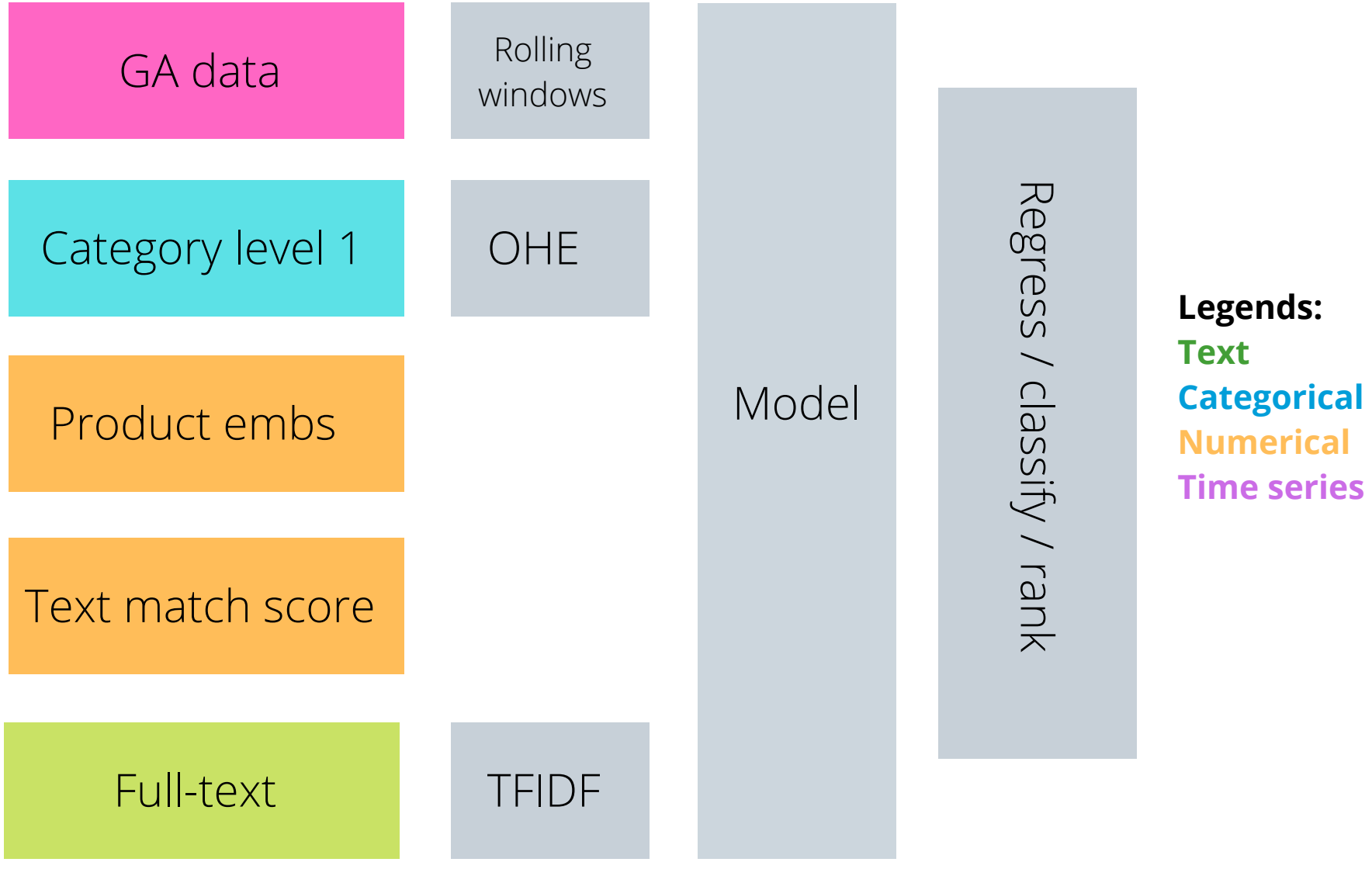
Meadow Lea Salt Reduced Spread 250g.

Orchid Pure Creamery Butter 10g. Pack 8

Party Coated with Butter Caramel Salt Flavored 60g.

Architecture Gameplan

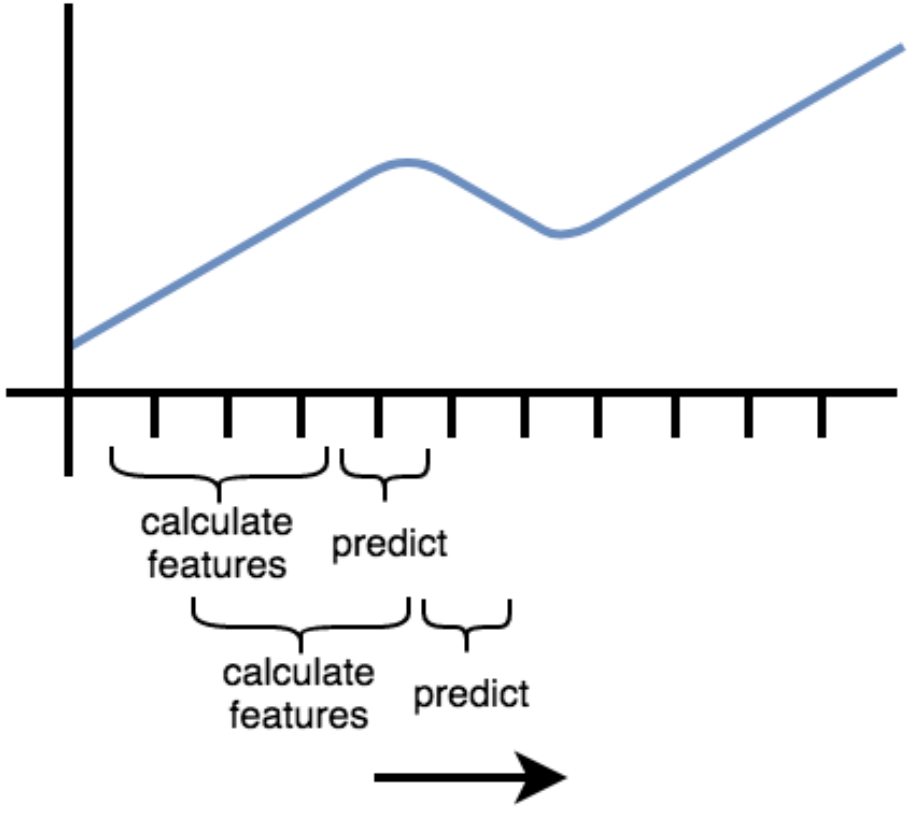
Time series user interactions and product characteristics



Simple Time Series Feature Extraction

Rolling windows of 7, 14, 28 days with some aggregations

What I need



What I think I would do

```
SELECT
    search_term,
    sku,
    date,
    SUM(impressions) OVER
    (PARTITION BY user
    ORDER BY date ROWS
    BETWEEN 28 PRECEDING AND
    CURRENT ROW)  as
    impressions_sum_28
FROM some_random_bigquery_table
```

source: https://tsfresh.readthedocs.io/en/latest/text/forecasting.html

Simple Time Series Feature Extraction

Why it is not so easy and some hacky solutions

Problem

Window functions are usually built for complete rows

	search_term	sku	date
1201404	dove	9300830037803	2019-07-06
2004204	dove	9300830037803	2019-07-08
2039768	dove	9300830037803	2019-07-09
2073097	dove	9300830037803	2019-07-10
2106864	dove	9300830037803	2019-07-11
206329	dove	9300830037803	2019-07-13
232225	dove	9300830037803	2019-07-14
259474	dove	9300830037803	2019-07-15

Possible Solutions

1. Aggregate for 7, 14, and 28 days before as table_7, table_14 and table_28 then join back
2. Left join all search_term-sku pairs to complete date

Both are very hacky and takes a lot of computing resources (5 hours for one month in Python and 50 seconds in BigQuery)

Categories as categorical features

Deeper-level categories are usually too sparse to be useful

Category Level 1

value	cnt	per
health & beauty care	12908	0.288912
pantry & ingredients	6336	0.141815
snacks & desserts	5850	0.130937
household & pet	4827	0.108040
beverages	3339	0.074735
fresh food & bakery	2486	0.055643
beer,wine & spirits	2111	0.047249
xxna	1965	0.043981
fruit & vegetables	1528	0.034200
mom & kids	1261	0.028224
meat & seafood	980	0.021935

30 values

Category Level 2

value	cnt	per
makeup	3975	0.088970
body care	2744	0.061417
facial care	2366	0.052957
hair care	2046	0.045794
xxna	1965	0.043981
biscuits cookies & crackers	1580	0.035364
seasonings & spices	1398	0.031291
wine	1181	0.026434
candies & chewing gum	877	0.019629
kitchen supplies	868	0.019428
chips	858	0.019204

131 values

Category Level 3

value	cnt	per
xxna	1965	0.043981
lips	1275	0.028538
eyes & brows	957	0.021420
biscuit & cookies	884	0.019786
foundation & facial powder	859	0.019226
shampoo	682	0.015265
facial moisturizer	666	0.014907
fresh vegetables	643	0.014392
liquid soap	638	0.014280
red wine	631	0.014123
chocolate	630	0.014101

442 values

All you need is embeddings

Collaborative filtering gives you bought-this-bought-that dense features

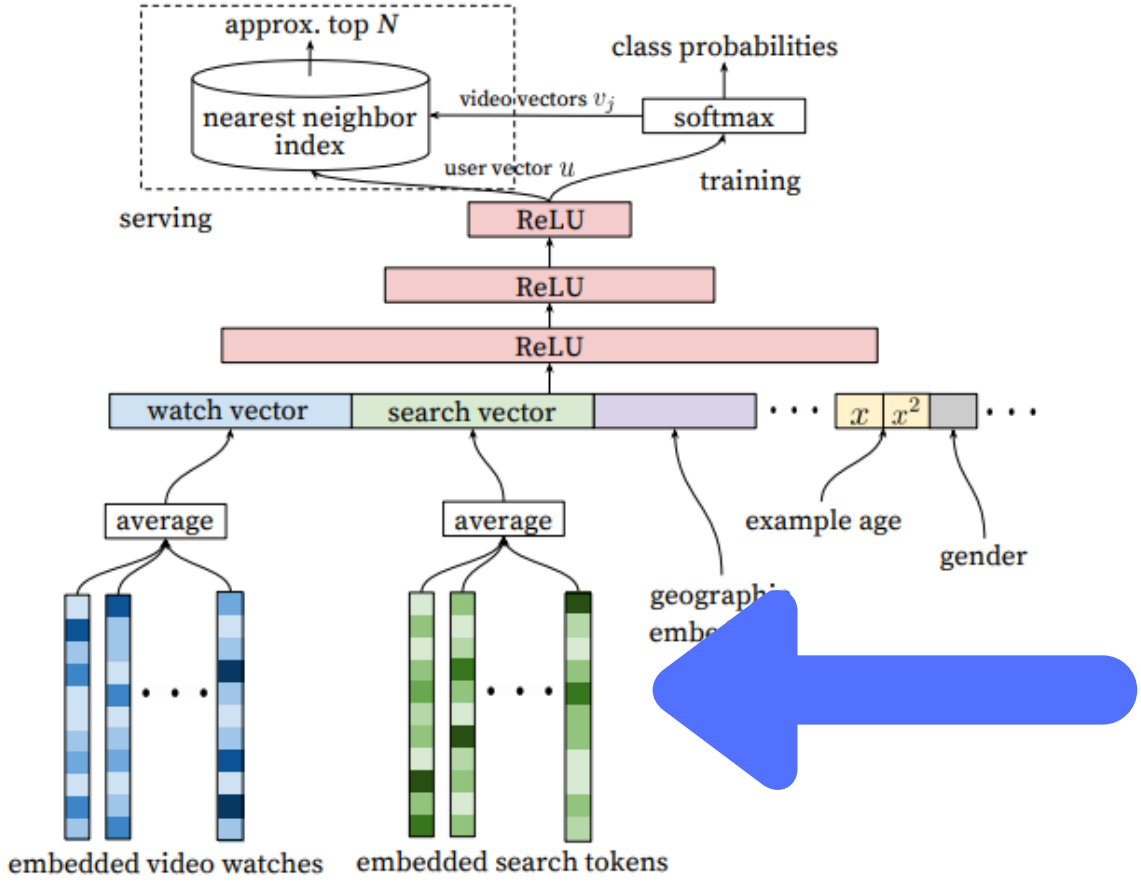


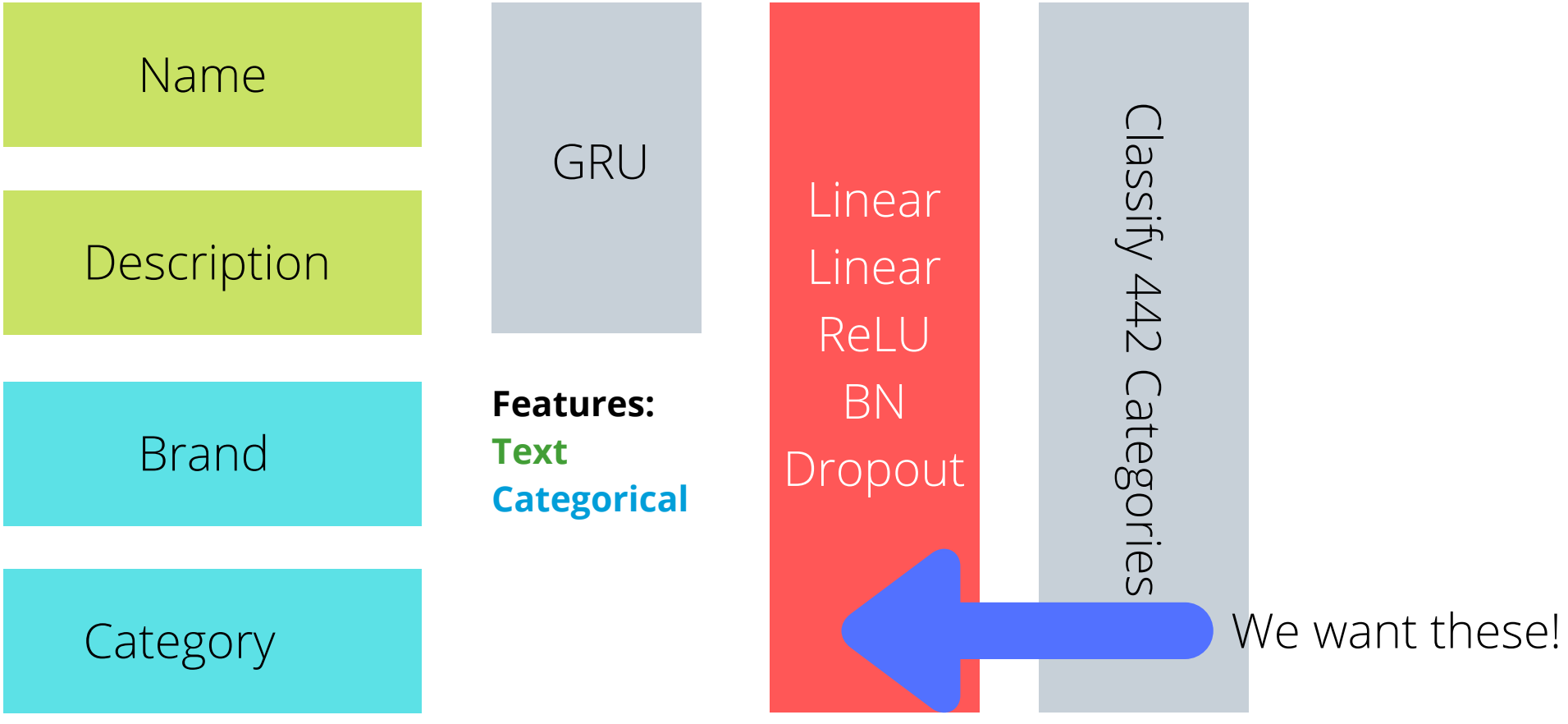
Figure 3: Deep candidate generation model architecture showing embedded sparse features concatenated with dense features. Embeddings are averaged before concatenation to transform variable sized bags of sparse IDs into fixed-width vectors suitable for input to the hidden layers. All hidden layers are fully connected. In training, a cross-entropy loss is minimized with gradient descent on the output of the sampled softmax. At serving, an approximate nearest neighbor lookup is performed to generate hundreds of candidate video recommendations.

Deep Neural Networks for YouTube Recommendations
(Covington, Adams, Sargin; 2016)
Paper: https://bit.ly/2wXInLD
Github Example: https://bit.ly/2Pwrhe4

We want these!

All you need is an encoder

Category prediction to create product characteristic encoder



Why BatchNorm after ReLU:
https://blog.paperspace.com/busting-the-myths-about-batch-normalization/

Product arithmetic word2vec style

Soda + Whiskey = Beer / Soda + Sugar = Soft Drinks

Github: https://github.com/witchapong/product-embedding-viz

```
model.most_similar_cosmul(positive=['จอห์นนี่วอคเกอร์แบลคเลเบิ้ลวิสกี้_1ลิตร','สิงห์โซดาเครื่องดื่ม_325มล_แพค_6'],
                          topn=20)
```

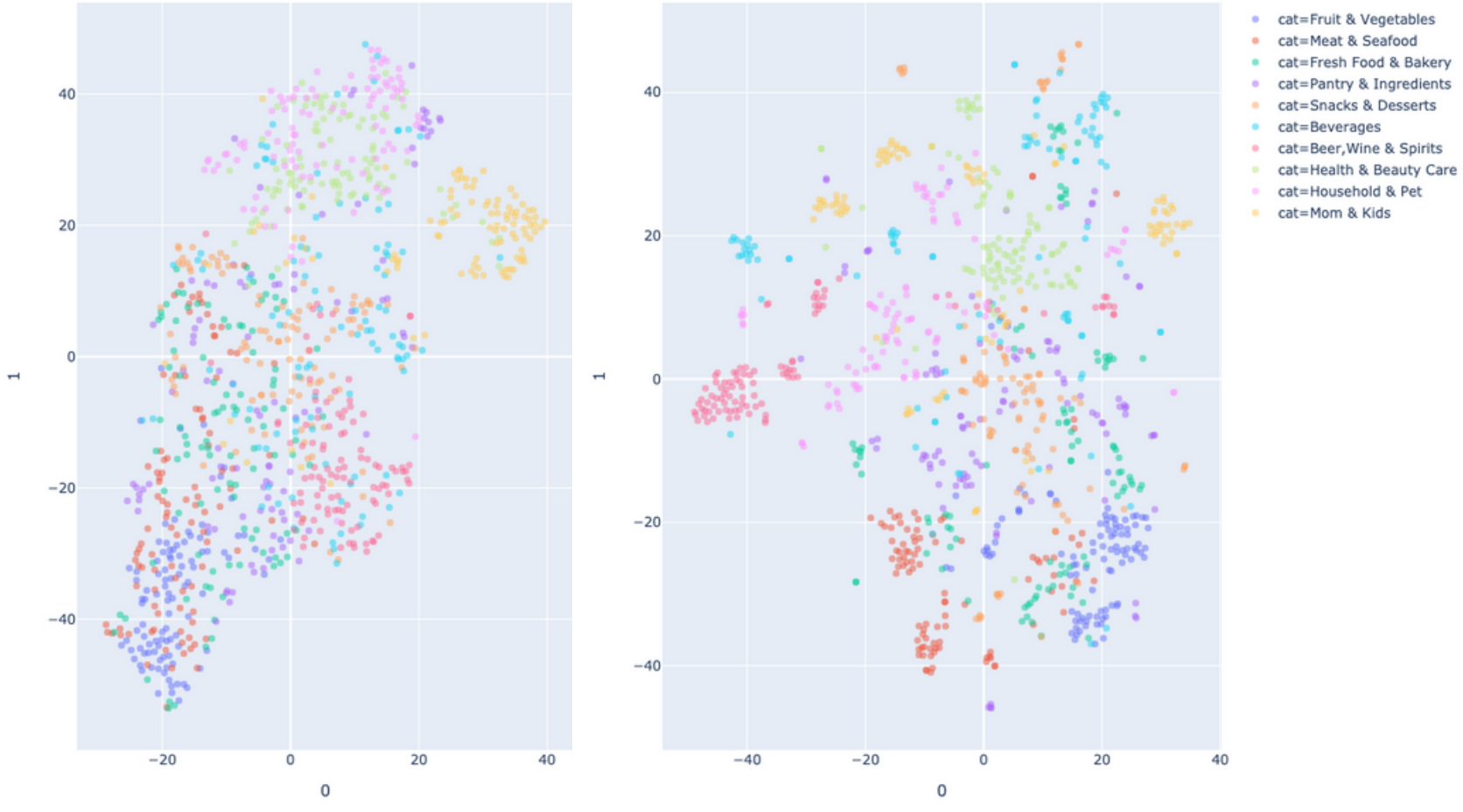
```
[('สิงห์เบียร์ขวดใหญ่_620ซีซี', 0.8387982249259949),
 ('สิงห์เบียร์กระป๋อง_320ซีซี_แพค_6', 0.8317778706550598),
 ('สิงห์เบียร์กระป๋อง_490ซีซี', 0.8223471641540527),
 ('สิงห์เบียร์ขวด_500ซีซี', 0.8207093477249146),
 ('สิงห์โซดาขวดเล็ก_325ซีซี_แพค_24', 0.8183379173278809),
 ('สิงห์เบียร์กระป๋อง_320ซีซี_แพค_24', 0.81
 ('สิงห์เบียร์ไลท์ขวด_620ซีซี', 0.814532935
 ('สิงห์เบียร์ขวดเล็ก_320ซีซี', 0.812136471
 ('สิงห์เบียร์กระป๋อง_490ซีซี_แพค_12', 0.81
 ('สิงห์โซดาขวดเล็ก_325ซีซี', 0.806398391
 ('สิงห์เบียร์ขวดใหญ่_620ซีซี_แพค_12', 0.80
 ('ไทเกอร์เบียร์ขวด_320มล_แพค_24', 0.79
 ('สิงห์เบียร์ไลท์_620ซีซี_แพค_12', 0.79841
 ('ชาวเกาะน้ำมันมะพร้าวบริสุทธิ์_200มล', 0.
 ('สิงห์เบียร์ขวดเล็ก_320ซีซี_แพค_24', 0.79
 ('สิงห์เบียร์แคน_320ซีซี_ไอซ์_แพค_12', 0.7
 ('ชาวเกาะน้ำมันมะพร้าวบริสุทธิ์_400มล', 0.
 ('เสือดำสุราผสม_28ดีกรี_0625ลิตร', 0.791
 ('สิงห์เบียร์ไลท์ขวดเล็ก_320ซีซี_แพค_24', 0
 ('เสือดำสุราผสม_28ดีกรี_033ลิตร', 0.7871
```

```
model.most_similar_cosmul(positive=['สิงห์โซดาขวดเล็ก_325ซีซี','วังขนายน้ำตาลทรายแคลเซียม_500กรัม'],
                          topn=20)
```

```
[('โทมัสเฮนรี่เครื่องดื่มอัดก๊าซกลิ่นดอกเอลเดอร์_200มล', 0.745958149433136),
 ('โทมัสเฮนรี่เครื่องดื่มอัดก๊าซกลิ่นเชอร์รี่บลอสซัม_200มล',
  0.7414625883102417),
 ('ตราเสือยาหม่อง_194กรัม', 0.7382895350456238),
 ('มิตรผลน้ำเชื่อม_180มล', 0.7327777147293091),
 ('เอสโคลา_250มล', 0.7288649678230286),
 ('เอสเครื่องดื่มอัดลมกลิ่นโคลา_1ลิตร', 0.7262457013130188),
 ('ถูกใจน้ำมันปาล์ม_1ลิตร', 0.7257730960845947),
 ('เอสน้ำหวานกลิ่นเลมอนไลน์_16ลิตร', 0.7221303582191467),
 ('มิตรผลน้ำเชื่อม_850มล', 0.7213984131813049),
 ('เอสเพลย์เครื่องดื่มกลิ่นฮาวายเอี้ยนพันช์_16ลิตร', 0.7213659286499023),
 ('เอสน้ำหวานกลิ่นสตรอเบอร์รี่_16ลิตร', 0.7191013693809509),
 ('เฟนติแมนส์จินเจอร์เบียร์เครื่องดื่มกลิ่นน้ำขิงอัดแก๊ส_275_มล',
  0.7143175005912781),
 ('เดอร์กี้แดงกลิ่นรสวานิลลา_473มล', 0.7137290835380554),
 ('เอสโคลา_515มล', 0.7136813998222351),
 ('เฟนติแมนส์จินเจอร์เอลเครื่องดื่มอัดแก๊สกลิ่นน้ำขิง_200มล',
  0.7120649814605713),
 ('ลินท็อปปิ้งกลิ่นคาราเมล_450มล', 0.7119836211204529),
 ('ชเวปส์น้ำโทนิค_330ซีซี_แพค_6', 0.7109414339065552),
 ('ไซโตอินเรียวเครื่องดื่มรามูเนรสพีชอัดก๊าซ_200มล', 0.7101669311523438),
 ('เฟนติแมนส์คิวเรียซิตี้โคลาเครื่องดื่มอัดแก๊ส_275_มล', 0.7098853588104248),
 ('ชีวาดีน้ำหวานดอกมะพร้าว100เปอร์เซ็นต์ออร์แกนิค_420กรัม',
  0.7098439335823059)]
```

Collaborative filtering vs Category prediction

Dense vectors are more nuanced than one-hot encoded features



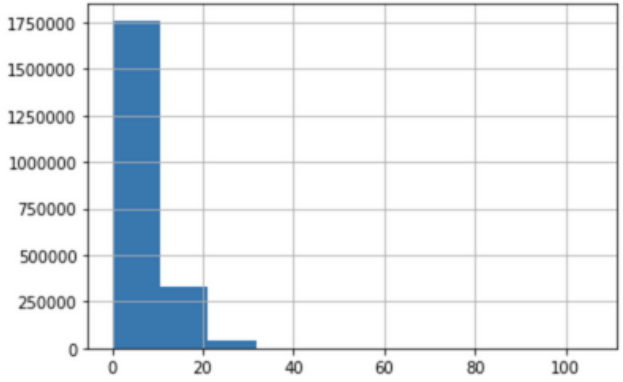
Github: https://github.com/witchapong/product-embedding-viz

Choice of Text-Matching Scores

Ease of implementation vs accuracy compared to ElasticSearch _score

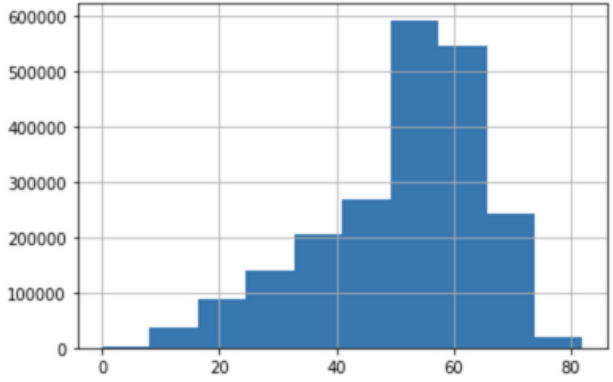
```
event_df.bm25_full_text.hist()
```

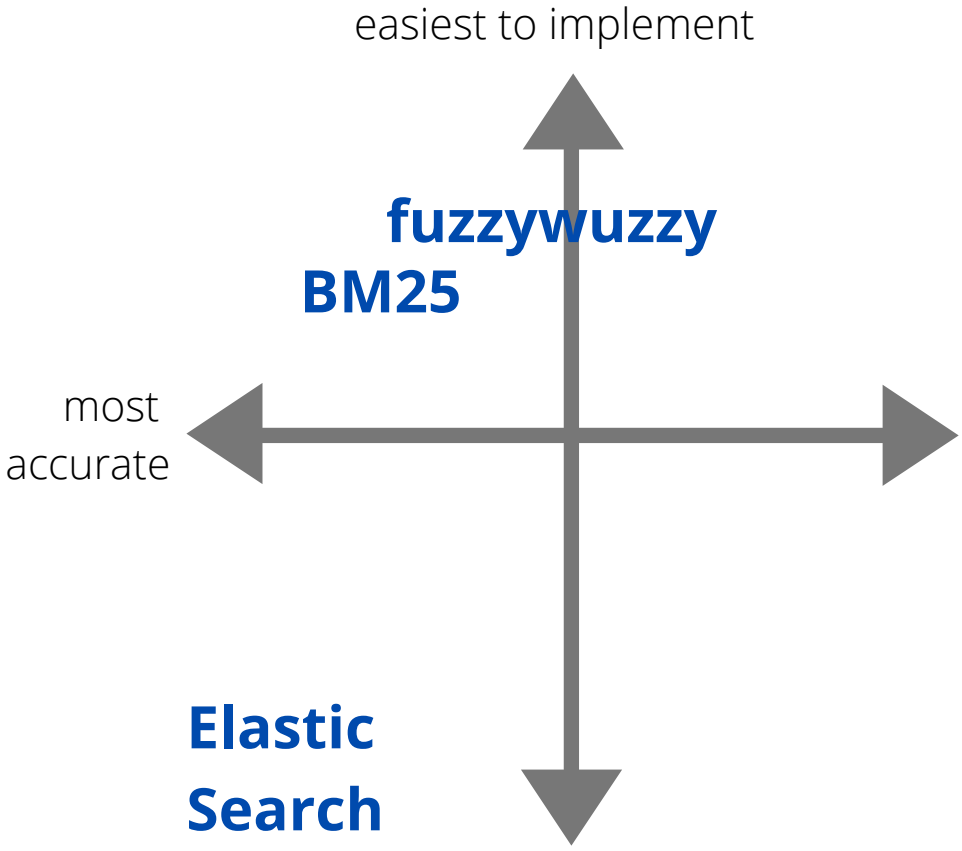
```
<matplotlib.axes._subplots.AxesSubplot at 0x144ecf5c0>
```




```
#This looks more spread out and possibly better as features
event_df.fuzz_full_text.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2c5196ba8>
```





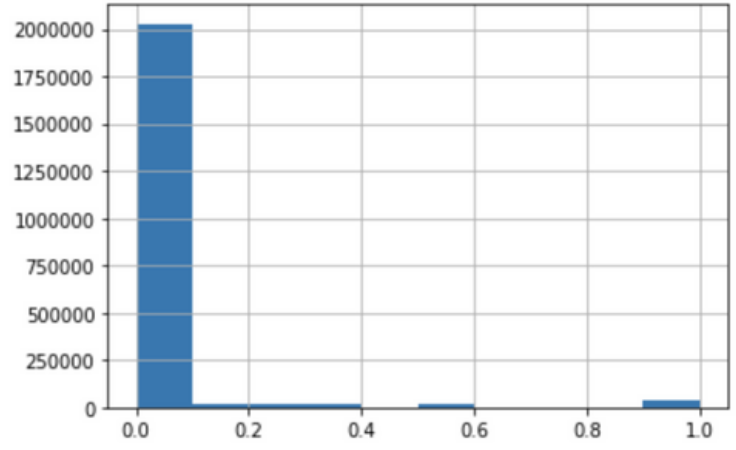
fuzzywuzzy: https://github.com/seatgeek/fuzzywuzzy

To regress or to classify or to rank?

Typically we want to maximize the click-thru rates

```
event_df.click_rate.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x149d279e
```



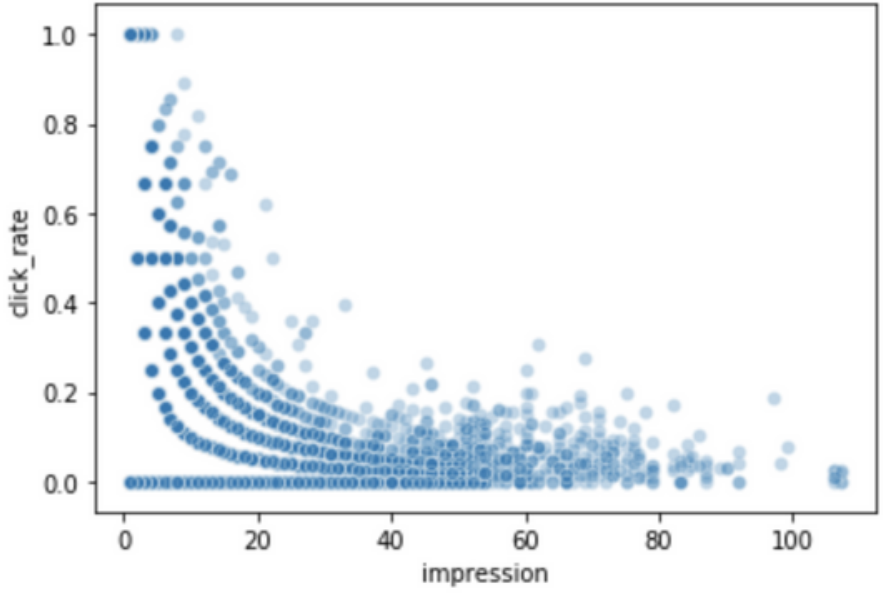

```
event_df.click_rate.describe()
```

```
count     2.132715e+06
mean      2.672722e-02
std       1.383618e-01
min       0.000000e+00
25%       0.000000e+00
50%       0.000000e+00
75%       0.000000e+00
max       1.000000e+00
Name: click_rate, dtype: float64
```

Regress: so many outliers; impossible for low-impression SKUs

Classify: will probably end up as clicked-vs-not-clicked classifier

Rank: Repeatedly compare clicked and not-clicked SKUs to rank them

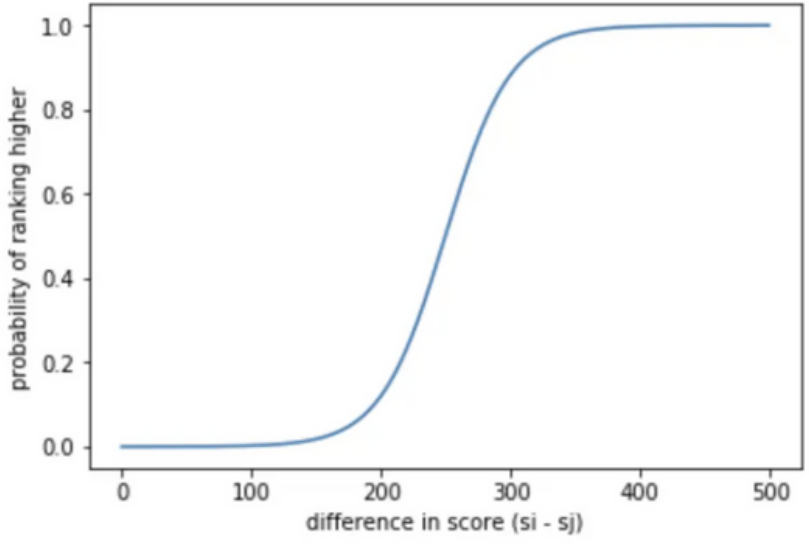


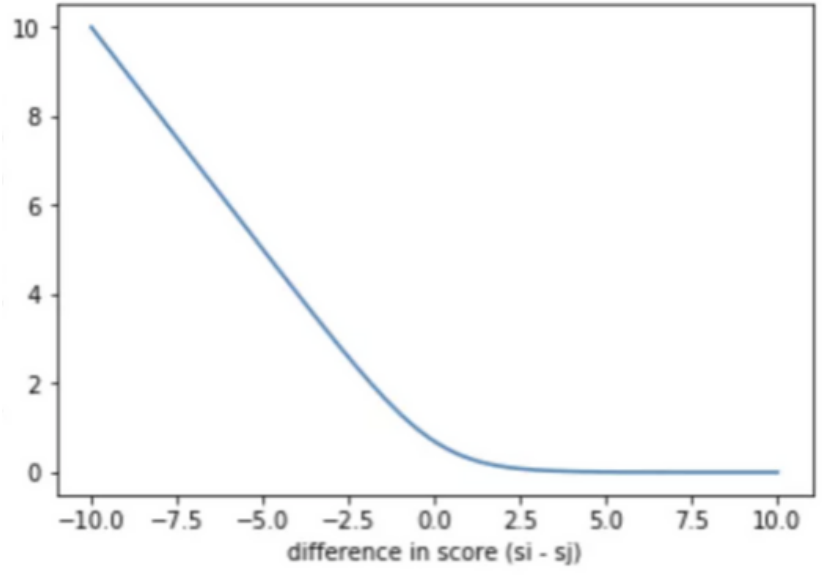
Learning-to-rank with Gradient Boosted Trees

We use binary rank function since very few of our SKUs have clicks

$$P(\text{rank}(i) > \text{rank}(j)) = \frac{1}{1+e^{-(s_i - s_j)}}$$

Negative logloss of P(rank(i) > rank(j))

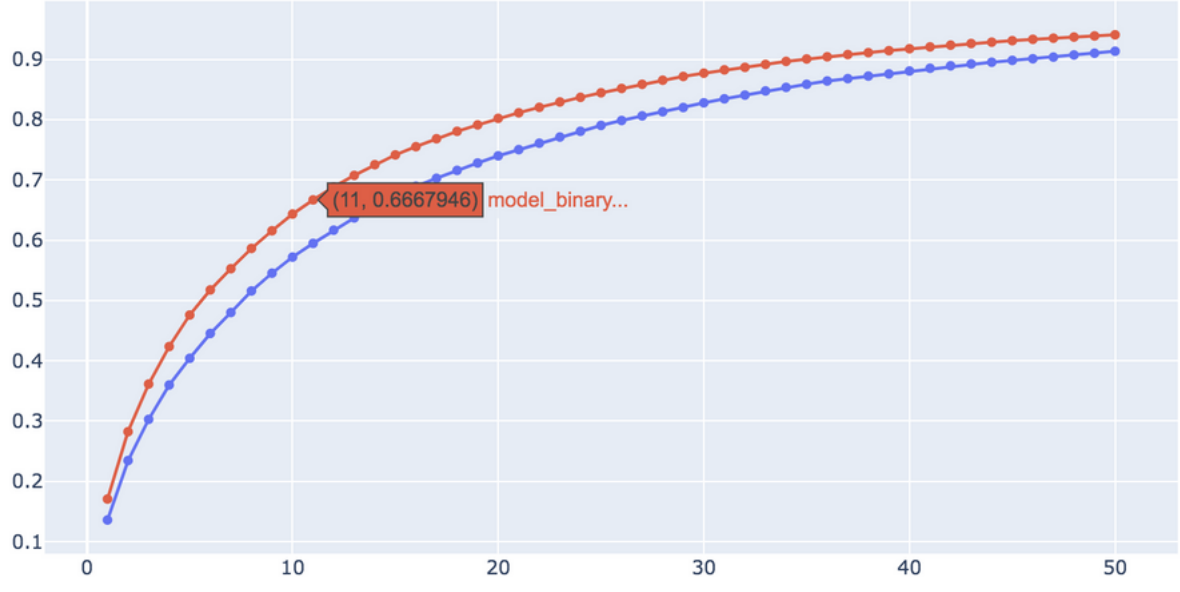




Learning-to-rank Explained:
https://mlexplained.com/2019/05/27/learning-to-rank-explained-with-code/
LightGBM:
https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMRanker.html

Can we beat ranking by past CTR?

Thankfully we can for all metrics



Accuracy@10
aka offline (fake) CTR:
66.7% vs 59.5% (12.1% uplift)

Average position clicked:
16.0 vs 19.8 (23.8% uplift)

Ablation studies

What features are actually most important

Feature set	Accuracy@10	NDCG@10
All	64.22	59.03
Compressed text feature	-.23	-.55
BM25 Score	-.12	-.33
Novelty	-.08	-.03
Aggregated SKU performance of last 28 days	-5.97	-4.63
Contextual feature	-.13	-.13
Categorical feature	-.05	-.11

Feature set	Accuracy@10	NDCG@10
All	66.8100	59.4526
Product text feature	-0.3208	-0.4391
Search term x product text matching feature	-0.6581	-0.7783
Product embedding feature	-0.2976	-0.1140
Aggregated events of last n days	-6.8986	-2.6349
Product novelty	-0.1587	0.1179
Categorical feature	0.0000	0.0000

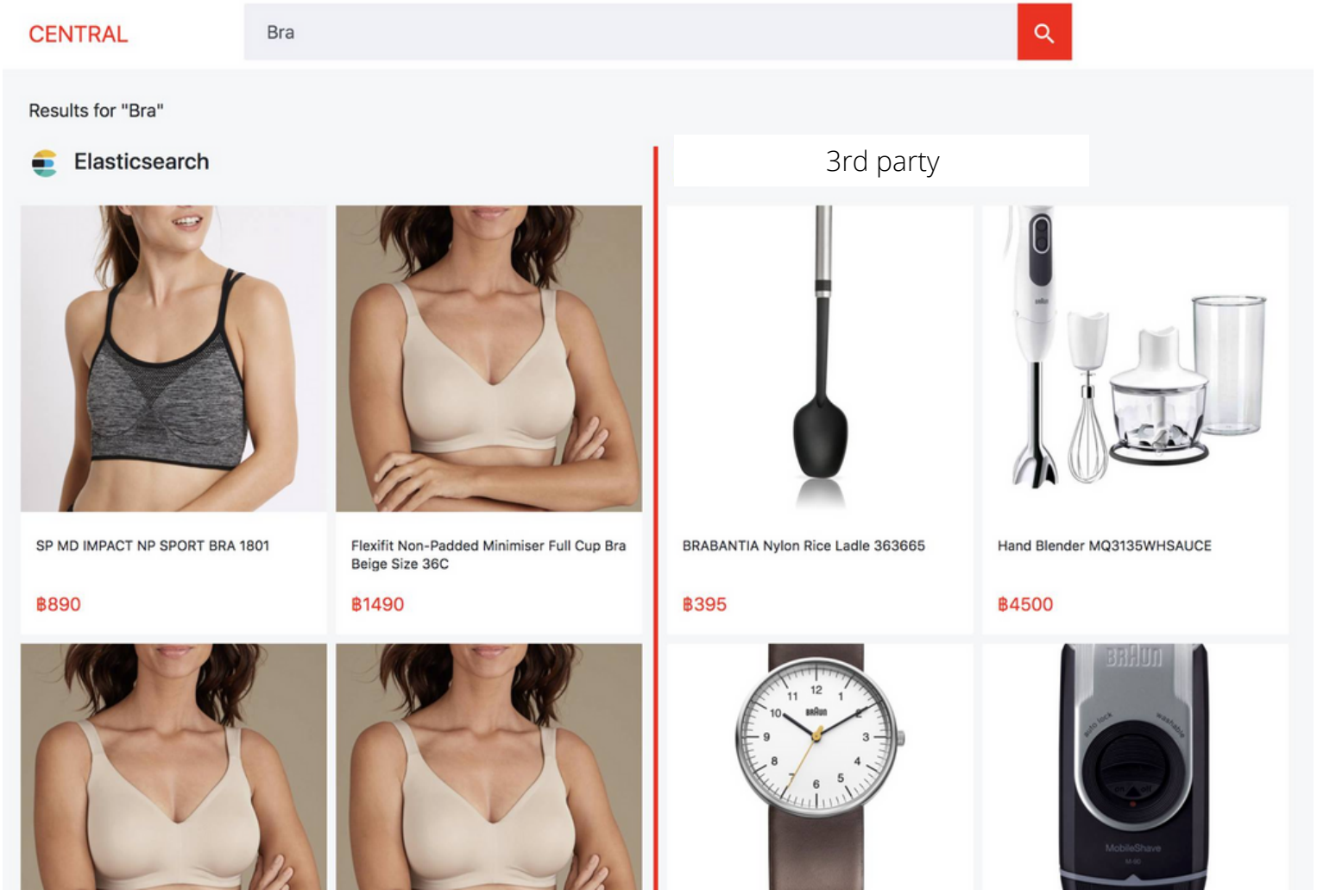
Improvements going forward

We are only at the beginning

- A better way to test full-text search logic than Kibana
- Move from lightGBM to Pytorch to retrain product embeddings; will be fun to see how it might help recommendations
- Implement all this within ElasticSearch and have it play well with multi-armed bandits for new products
- Better way to engineer time series features, which is by far our most important feature
- More accurate and easier way to replicate ElasticSearch text-matching scores
- Add image features
- Add customer embeddings for personalized search; depends on incremental gains

There are some people who prefer *magic*

`bra` on elasticsearch vs third-party vendor that "works out of the box"



People can and will abuse your product search

Marketing team uses search page as landing pages

Our Team

Aka people who actually did the work

June - Full-text search
Github: https://github.com/ben-mj

Tle - Tokenizer and engineering
Github: https://github.com/tlefsad

Mick - Modeling
Github:
https://github.com/witchapong
Page:
https://www.facebook.com/datawizthailand