Towards an open source tool stack for e-commerce search

MICES 2020

Eric Pugh, Johannes Peter, Paul M. Bartusch, René Kriegler

Who we are

Combined 45 years of experience in search

Open Source enthusiasts

ASF member, Committers on: Solr, Querqy, SMUI, Quepid, Contributions: RRE, NiFi

@pbartusch @renekrie @dep4b

In this session...

You will meet Pete - ‘Product Owner E-commerce Search’ (and see Pete struggle through a search project)

Learn about open source components that help speed up search development and search quality work (and make Pete’s life easier)

Learn and discuss about Chorus and Querqy.org - our project to kick-start e-commerce search with open source (and to let Pete do the ‘really cool stuff’)

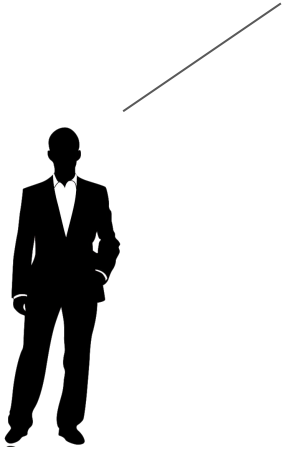
Pete got hired as the Product Owner for search:

> Make our search **better**!

> Build me a **best-in-class** search!

> Great challenge! Thanks!

Pete establishes his perspective on the business challenges with search in his company:



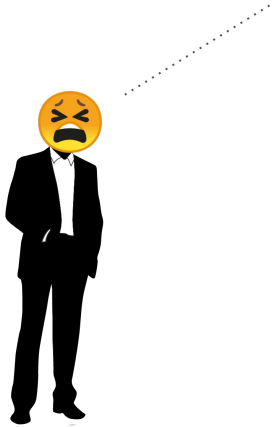
> We are dealing with a lot of **complaints** about our search!

> Search might be tricky for our type of products and given our target customers, but if we got it right, we would definitely **sell more** and have a great **advantage over** our **competitors**!

> We want to go **far beyond text matching** with our innovations!

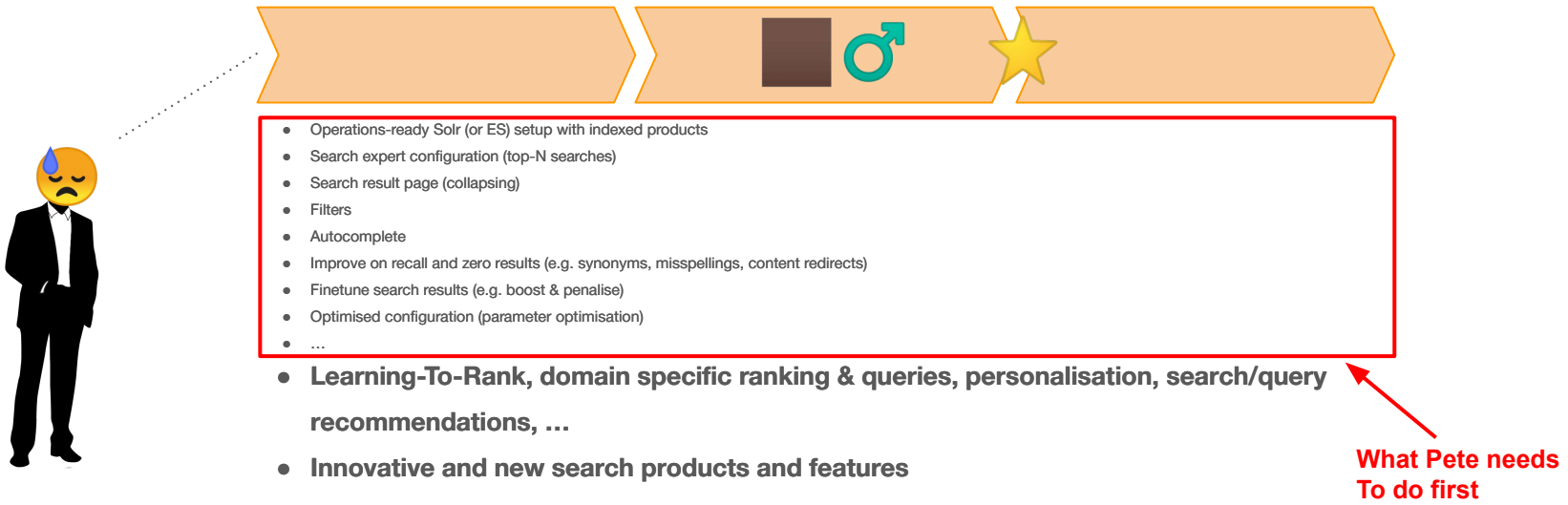
> Our managed search solution is **too expensive** and **doesn't offer enough flexibility**! We want to **improve** on our **search**!

Pete thinks about his approach:



- If we decide, to **use** a **closed source, commercial solution**
- … we might end up with too little flexibility!
- … we will have a hard time to implement our great ideas! Will we ever **achieve that competitive advantage**?
- … we would not **own** our **search**!

> If we decide, to **build** our search **from scratch**, using open source, we need to plan many epics to **get on par with commercial solutions** before we can **beat the market**!



> If we decide, to **build** our search **from scratch**, using open source, we need to plan many epics to **get on par with commercial solutions** before we can **beat the market**!

> **Can we speed this up?**

- Operations-ready Solr (or ES) setup with indexed products
- Search expert configuration (top-N searches)
- Search result page (collapsing)
- Filters
- Autocomplete
- Improve on recall and zero results (e.g. synonyms, misspellings, content redirects)
- Finetune search results (e.g. boost & penalise)
- Optimised configuration (parameter optimisation)
- ...

The hot stuff

- **Learning-To-Rank, domain specific ranking & queries, personalisation, search/query recommendations, ...**
- **Innovative and new search products and features**

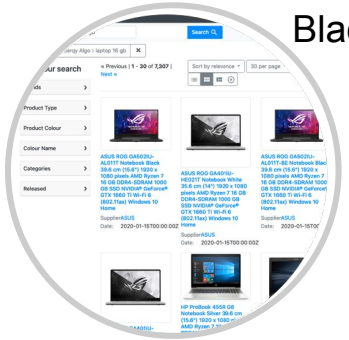
Pete knows:

> Search Quality correlates with money earned at the Chorus Electronics online store!

We got something for you, Pete!

> Open Source Software Components!

List of OSS Components

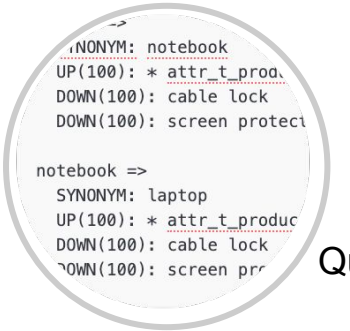


Blacklight

SMUI

Quepid

Chorus



Querqy

RRE

Demo: Assessing Quality in the *“Chorus Electronics Store”*

- OSS Components!

Pete knows, how important active search management is:

> More than 65% of our searches traffic will be optimised

search method vs searches (Google Analytics)	6.149.403
1. spText+rules	3.945.583 (64,16 %)
2. spText	1.004.882 (16,34 %)
[redacted]	[redacted]
[redacted]	[redacted]
5. spTextRelaxed+rules	58.958 (0,96 %)
6. spTextCorrected+rules	50.849 (0,83 %)
7. spContentDirectHit rules	40.110 (0,65 %)
8. spTextRelaxed	37.993 (0,62 %)
9. spTextCorrected	29.047 (0,47 %)
[redacted]	[redacted]

Example based on ca. 6 million searches on a platform of a European e-commerce retailer.

Demo: Querqy + SMUI

- OSS Components!

Demo: Measuring search relevance with RRE and Quepid

- OSS Components!

Break

... Querqy for queries

- OSS Components!

The Querqy library

Library and framework for

- query rewriting
- optimised query building with many parameters to tune search relevance

Plugins available for Solr (2014) and Elasticsearch (2019)

Apache 2 License

github.com/querqy & querqy.org

The Querqy library and SMUI

Developed with e-commerce search in mind* - users & contributors include:

… and others ...

* ... but works well in other domains too

Why would we rewrite a query?

Word-level symbols and semantics

Interprete query intent

Seller interests

Word-level symbols and semantics

Interprete query intent

Seller interests

Synonyms

Word breaks

Orthographic normalisation

Interpret semantic relations Dutch: ‘voer voor honden’ (‘food for dogs’) = ‘hondenvoer’ (dog food); interpret ‘without’

Recognise entities / match with fields

Word-level symbols and semantics

Interprete query intent

Seller interests

Boost ‘laptop’ => Bring laptop computers to the top

Penalise ‘laptop’ => Push ‘sleeve’ to the end of the result list. Make sure tablets are not at the top.

Filter ‘mouse’ => only show computer accessories

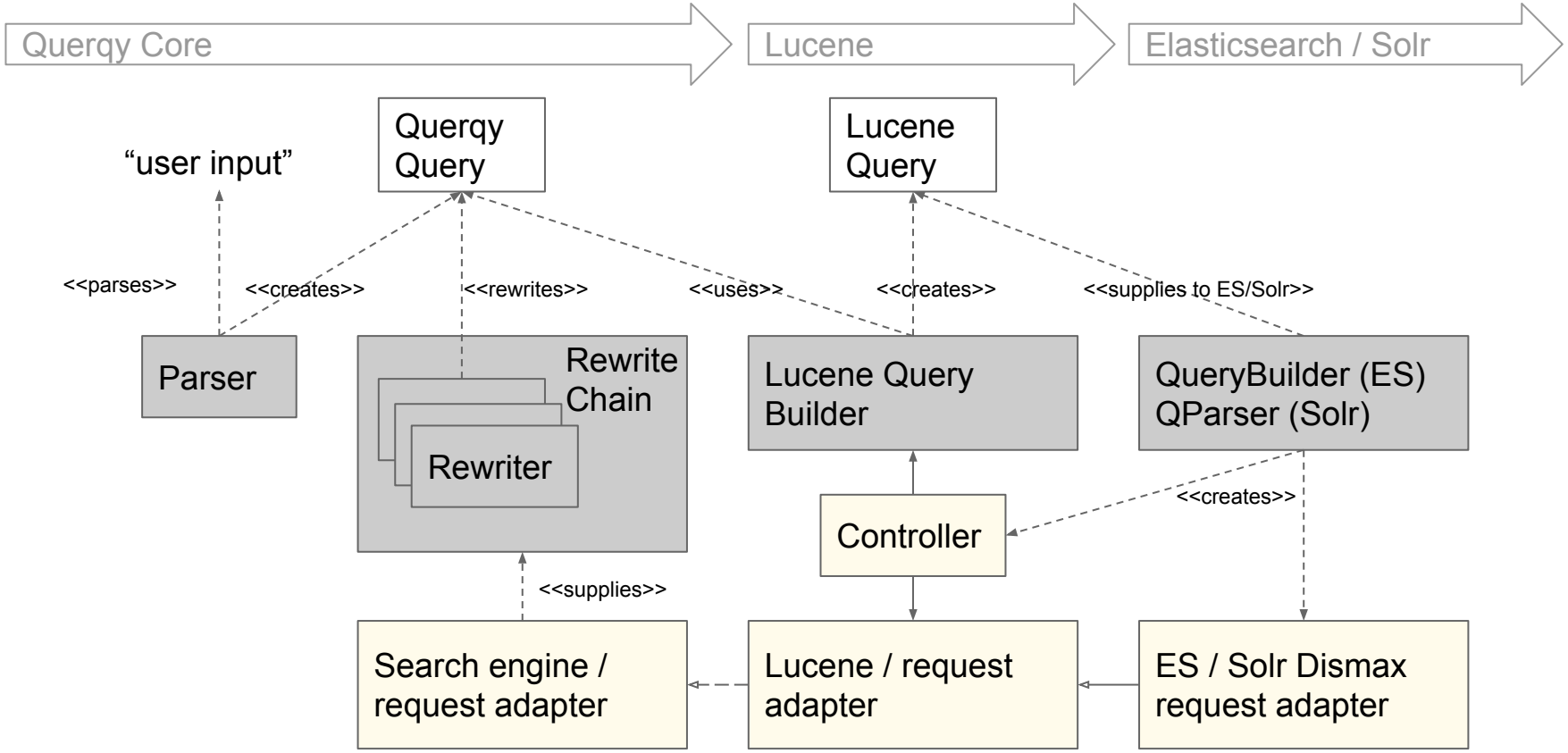
Word-level symbols and semantics

Interprete query intent

Seller interests

Cost/profit optimisation

Brand reputation ‘t-shirt’ => don’t show basic t-shirts at the top, we are a t-shirt design trend setter!



Querqy under the hood

... more query rewriting with Querqy

- OSS Components!

Rewriters in Action

Rewriters that come with Querqy

- Common Rules Rewriter
- Replace Rewriter
- Number-Unit Rewriter
- Shingle Rewriter
- Word Break Rewriter

```
mobile => (mobile OR smartphone)

ombile => mobile

laptop 15” => laptop AND screen_size:[13.5 TO 16.5]

i phone => (i phone OR iphone)

grainfree => (grainfree OR grain free)

voer voor honden => (voer voor honden OR hondenvoer)
```

- Write your own - it’s a framework!

Common Rules Rewriter - Advanced Usage

- RawQuery: Opening the endless world of combined Querqy-Lucene power

```
new =>

    DELETE: new

    UP(1.0): * release_date:[NOW/DAY-4DAYS TO NOW/DAY+1DAY]


special offer =>

    DELETE: special offer

    FILTER: * strike_price:[* TO *]

    UP(1.0): * {!func}if(gte(rint(mul(div(sub(strike_price,price),strike_price),100)),20),20,0)
```

Replace Rewriter - Simple Rules

- Handling term variations (spellings, plural, …)

```
newer; newest; new offers => new

cheapest smartphones => cheap smartphone

/ =>

+ => plus
```

- Common Rules Rewriter (subsequently applied)

```
new =>

    DELETE: new

    UP(1.0): * release_date: …
```

Replace Rewriter - Wildcard Rules

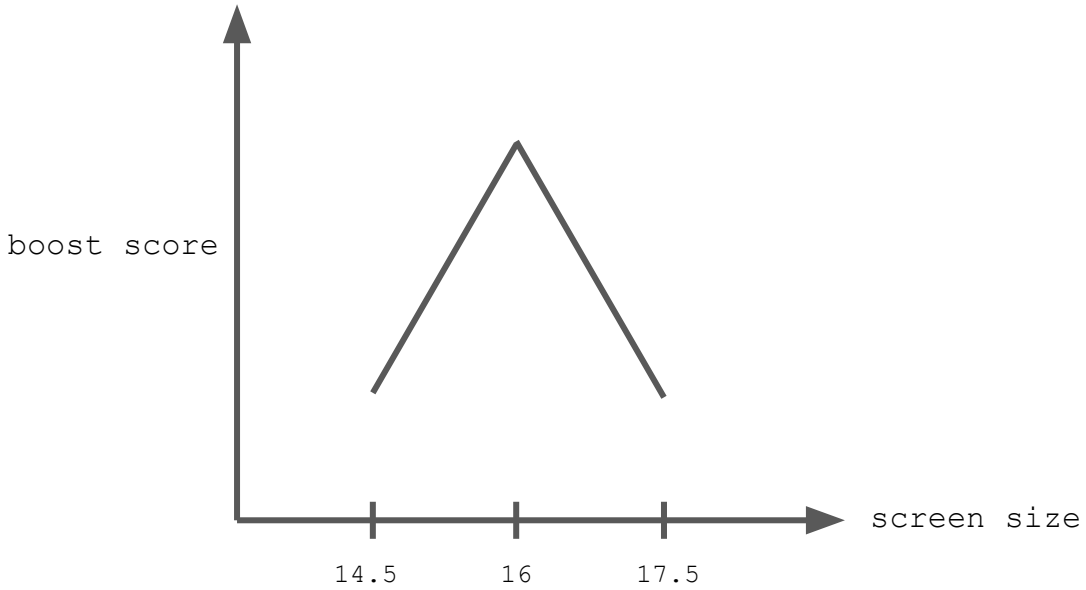
- Handling prefixes and suffixes more generic

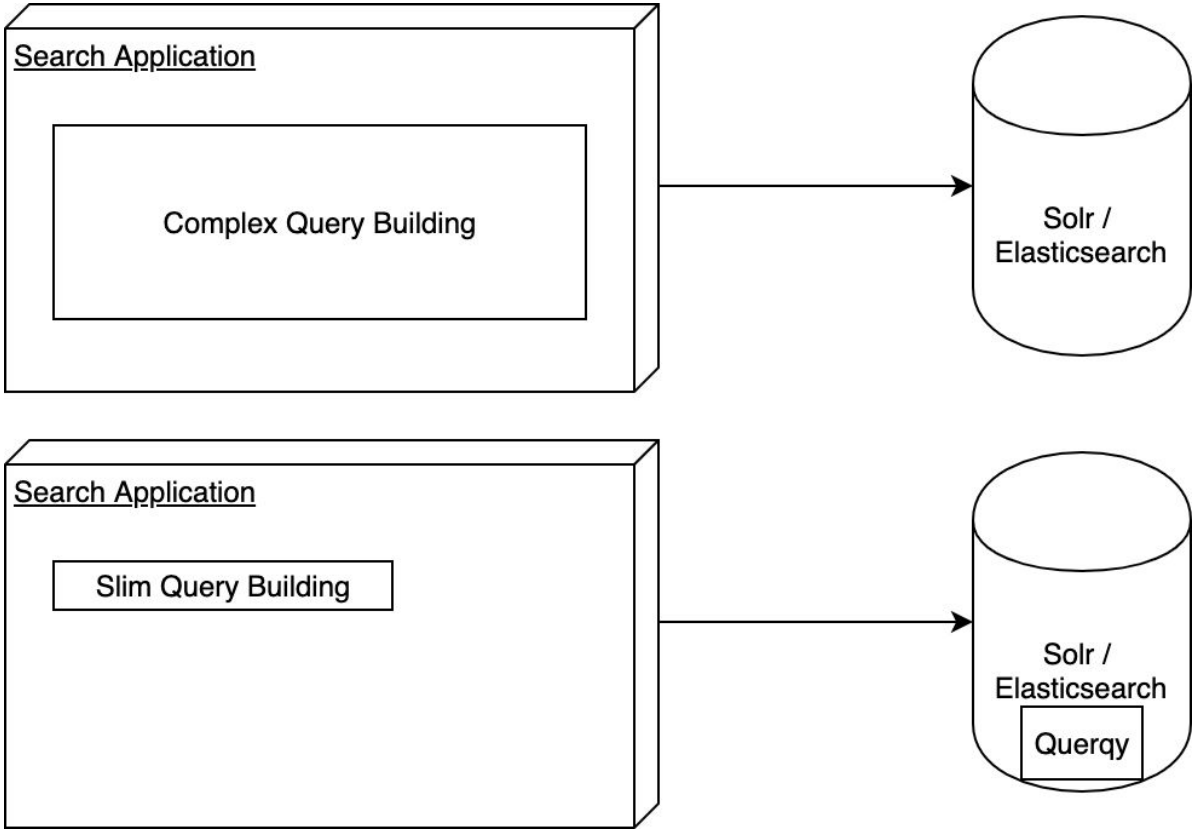
```
new* => new                     # newer; newest; ... => new

msart* => smart$1               # msartphone => smartphone

computer* => computer $1        # computerdesk => computer desk


*phones => $1phone              # smartphones => smartphone

*+ => $1 plus                   # s8+ => s8 plus

*) => $1

(* => $1                        # (2018) => 2018
```

Number-Unit Rewriter

- Rewriting number-unit combinations to filter and boost queries

```
notebook 16 inch  => notebook AND screen_size:[14.5 TO 17.5]
```





Reducing query building complexity with Querqy

Querqy for query building

query=samsung notebooks

What Elasticsearch should produce:

```
AND(
     OR(
          brand:samsung,
          product_type:samsung
     ),
     OR(
          brand:notebooks,
          product_type:notebooks
     )
)
```

Querqy for query building - Elasticsearch

query=samsung notebooks

What Elasticsearch should produce:

```
AND(
    OR(brand:samsung,
        product_type:samsung
    ),
    OR(brand:notebooks,
        product_type:notebook
    )
)
```

What

```
"multi_match": {
    "type": "cross_fields",
    "query": "samsung notebooks",
    "fields": ["brand", "product_type"]
}
```

creates:

```
    OR(
        AND(brand:samsung,
            brand:notebooks),
        AND(product_type:samsung,
            product_type:notebook)
    )
```

Querqy for query building - Elasticsearch

query=samsung notebooks

What Elasticsearch should do:

```
AND(
    OR(brand:samsung,
        product_type:samsung
    ),
    OR(brand:notebooks,
        product_type:notebook
    )
)
```

What Elasticsearch users do:

```
"bool": {
    "must": [
        "multi_match": {
            "type": "cross_fields",
            "query": "samsung",
            "fields": ["brand",
                       "product_type"]},
        "multi_match": {
            "type": "cross_fields",
            "query": "notebooks",
            "fields": ["brand",
                       "product_type"]}
    ]}
```

Querqy for query building - Elasticsearch

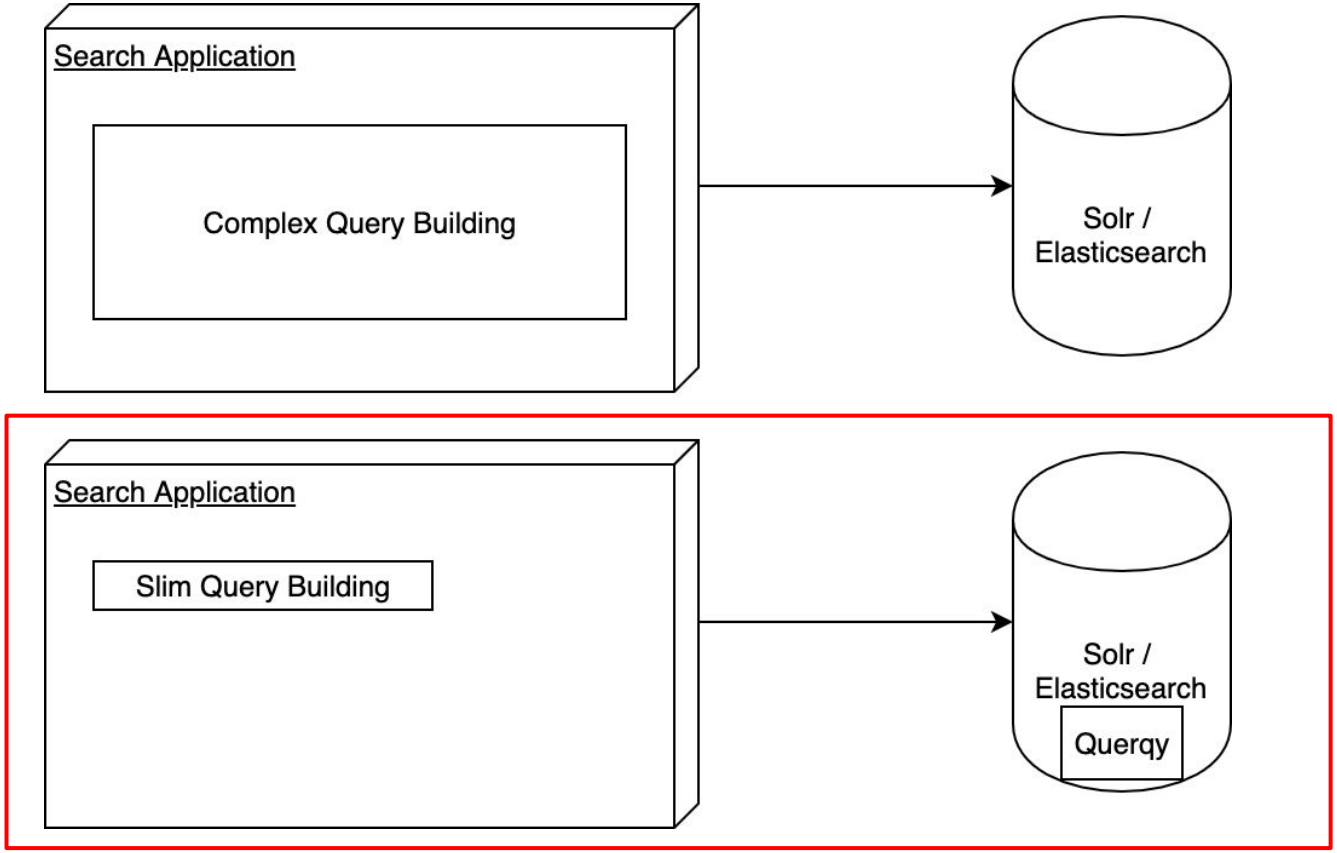
query=samsung notebooks

What Querqy does:

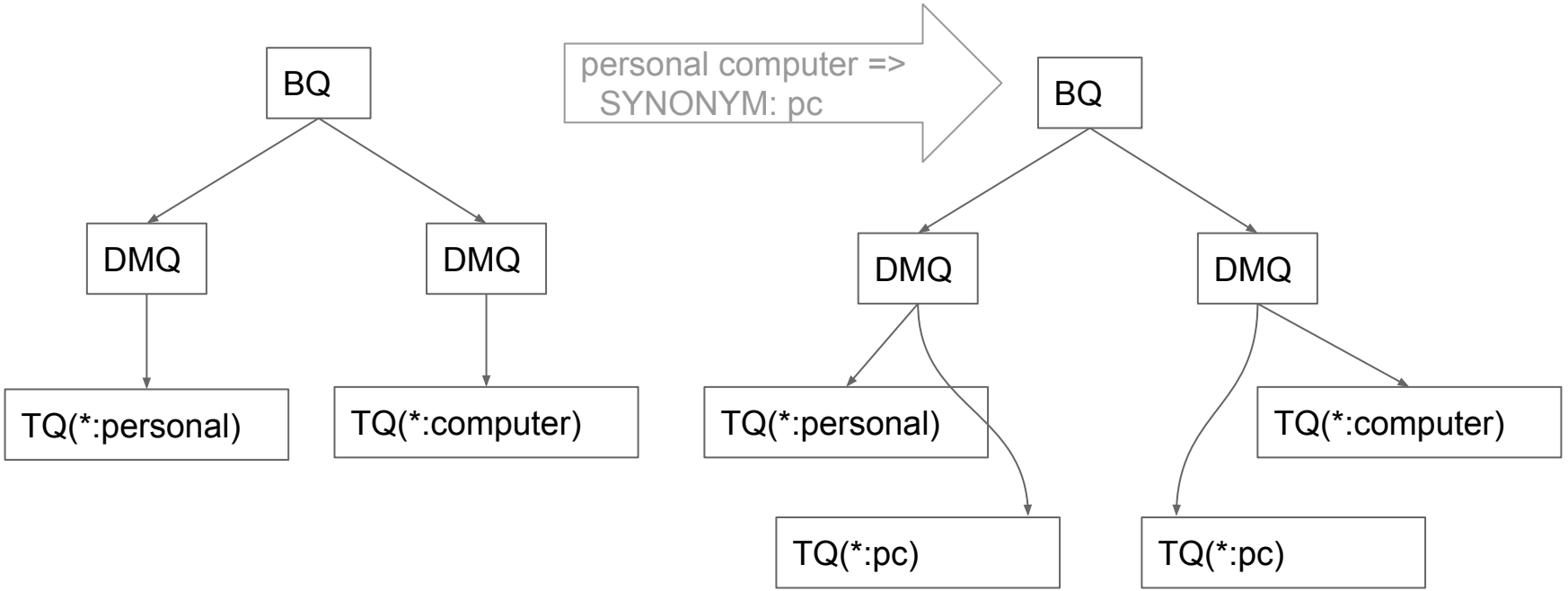
```
AND(
    OR(brand:samsung,
        product_type:samsung
    ),
    OR(brand:notebooks,
        product_type:notebook
    )
)
```

Using the `querqy` query builder for ES:

```
"querqy": {
    "matching_query": {
        "query": "samsung notebook"
    },
    "query_fields": ["brand",
                     "product_type"],
    "rewriters": [...]
}
```



Reducing query building complexity with Querqy



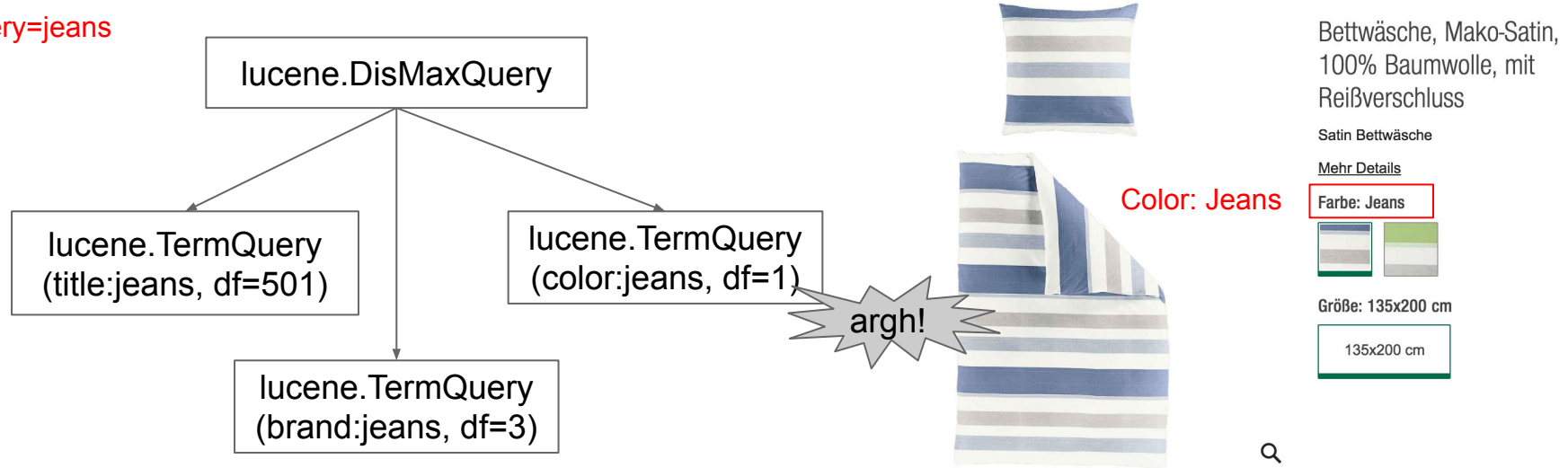
Preserves ‘minimum should match’ / boolean semantics
Works well even for multiple multi-term term synonym input/output
Applied before expanding to fields and before Lucene Analysis chain

Query rewriting: synonyms

BQ: Boolean Query, DMQ: DisMax Query, TQ: Term Query

score ~ (term frequency * inverse document frequency * ...)

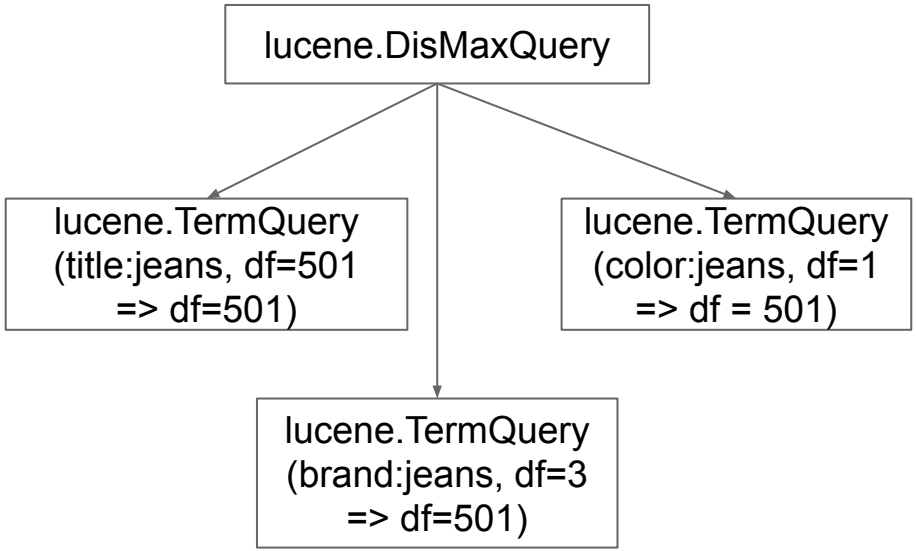
query=jeans



Scoring based on tf*idf vs. fields is a problem

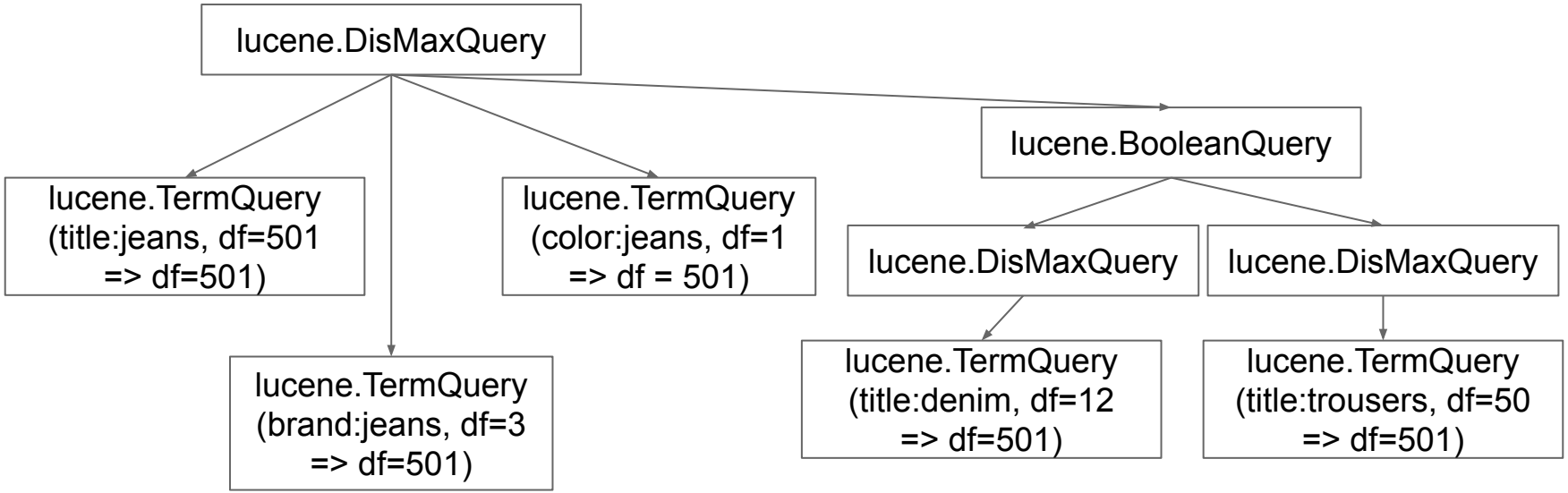
... and e-commerce search makes extensive use of fields!

Lucene Query building: document frequency



Use the max. Document Frequency for all terms a top-level query term is expanded to
Similar to BlendedTermQuery / SynonymQuery in Lucene

Lucene Query building: document frequency in Querqy



Also works for complex query expansions (unlike BlendedTermQuery / SynonymQuery in Lucene)
Enabled by default. Configurable alternatives: standard Lucene scoring, or turn off TF*IDF altogether

Lucene Query building: document frequency in Querqy

- putting the components together

Making Pete’s life easier:

> All these tools - I wish I had known them in the first place!
>
> And how will they work together? Who else uses them?

Why a tool stack for *e-commerce* search?

Strong demand for onsite $€ARCH optimisation

- Needs tools to measure quality
- Sophisticated search management requirements
- Reduce ramp-up time and start optimising for your business earlier

Reduce the gap: open source search engines not built for e-commerce

- Ranking models made for text documents vs highly structured data (fields!) in e-commerce
- How do we deal with variants of a product? (matching, ranking, facets)

Chorus: Open source stack for e-commerce search

- Reduce time to get on par with commercial search engines
  - package as integrated toolset for Elasticsearch and Solr (Chorus bootstrap application)

- Supply solutions for typical e-commerce search:
  - Tools to support merchandiser/search manager (SMUI/Querqy)
  - Easy and extendible query building and rewriting (Querqy)
  - Manual judgment collection (Quepid)
  - Automated search relevance testing (RRE)
  - Parameter/configuration optimisation (??, maybe RRE or Quaerite)
  - Simple search UI (Blacklight)

Vision

Making Pete’s life easier:

> All these tools - I wish I had known them in the first place!
>
> And how will they work together? Who else uses them?

> **Now we can beat our competitors:**

Chorus

Integrated open source stack to bootstrap e-commerce search

Querqy.org

Umbrella project

… What’s next

Querqy.org - What’s next?

Establish a community process

Elasticsearch versions of SMUI and Chorus

Validate Chorus work smoothly for many different organizations

Show a path from bootstrapping with Chorus to production

Integrate your ideas and tools in the Chorus Stack!

Get in touch

hellopete@querqy.org

https://querqy.org

https://github.com/querqy

https://ecom-search.slack.com