How much searchandizing is too much searchandizing?

MICES 2026

MIx-Camp E-commerce Search

Charlie Hull - The Search Juggler - www.thesearchjuggler.com

Who am I?

- Expert search consultant with 25+ years experience
- Clients have included governments, national media companies & billion-dollar e-commerce giants
- Led the Haystack conference series for 5 years
- ex-Managing Consultant & leadership team at OpenSource Connections
- Co-host of the London Search & AI Meetup (with the OpenSearch team)
- Co-author of ‘Searching the Enterprise’
- …and yes, I really do juggle!

www.thesearchjuggler.com
https://x.com/FlaxSearch
https://bsky.app/profile/thesearchjuggler.bsky.social
https://hachyderm.io/@flaxsearch
https://www.linkedin.com/in/charliehullsearch/

www.searchmeetups.com

OpenSearch Software Foundation Ambassador 2026

Vespa Partner

Vespa.ai Live

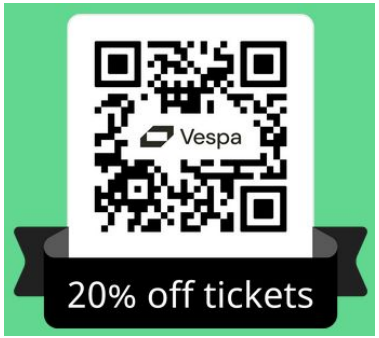
9-10 September 2026 at 📍 Lumiere London

A community event for the builders, architects, researchers, and practitioners shaping the next generation of AI-powered search and retrieval systems.

Speakers include experts from Etsy, Walmart, Ravenpack; Alessandro Benedetti, Atita Arora, Doug Turnbull, Trey Grainger & the Vespa team. Also includes a panel discussion, BarCamp and associated Meetups.

Only £99 a ticket in person / £20 online - Vespa training also available on 9th September

www.vespaai.live



What is searchandizing?

What is searchandizing?

It's modifying search results & their ranking for **business reasons**.

- Standard search algorithms match based on keywords or vectors (or both)
- Ranking is based on scores generated during matching, often with some other factors (e.g. overall product popularity)
- Users may also sort/filter by price, availability etc.
- The order of search results is *neutral* from a business perspective

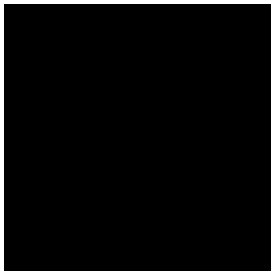
search + merchandizing = searchandizing

What are some ‘business reasons’?

- Users don’t use the same terms as we do to describe products
- Our product data quality isn’t always great so we need to restrict or promote some results we know are good matches
- An agreement with a supplier that their products should appear first
- Domain knowledge tells us that X is a better result than Y for this query
- We have a promotion or sale running and want to push some products higher up the list
- We don’t carry a particular brand, but we have an alternative

What are some ‘business reasons’?

- Users don’t use the same terms as we do to describe products
- Our product data quality isn’t always great so we need to restrict or promote some results we know are good matches
- An agreement with a supplier that their products should appear first
- Domain knowledge tells us that X is a better result than Y for this query
- We have a promotion or sale running and want to push some products higher up the list
- We don’t carry a particular brand, but we have an alternative
- **An executive has found a broken query - fix it now!!**



Examples

- The ‘accessories problem’
  - Fix by boosting results matching a category (e.g. actual laptops not laptop PSUs and sleeves)
- No results
  - Add synonyms to translate user queries to product language or to provide alternative brands
- Functional queries (“sale”, “office location”)
  - Add redirects to fixed pages
- Promotions & favoured suppliers
  - Boost results matching certain field values (“sale=TRUE” or “supplier=MyCo”)

Advantages of Searchandizing

- Rules are only triggered by one query (or a pattern)
  - Low risk of impact across other queries
- Most systems let you apply rules to production
  - Fix issues in minutes, no need for re-indexing or a new search algorithm
- Reduces load on the search team
  - Push the effort to the edge, closer to the business
- Focuses on the specific
  - Your data and users are unique - no OOTB algorithm or model will fit

How do we searchandize?

How does searchandizing work?

- We create *business rules* to modify a query
- Each rule triggers on a particular query (or pattern)
  - E.g. not a global fix - unlike index-time synonyms or a query-time field boost
- We can:
  - Add boosts (positive or negative: 'bury')
  - Remove or replace words
  - Split or combine words
  - Add filters
  - Redirect to a fixed page

```
# if the input contains 'personal computer', add two synonyms, 'pc' and
# 'desktop computer', and rank down by factor 50 documents that
# match 'software':

personal computer =>
  SYNONYM: pc
    SYNONYM: desktop computer
  DOWN(50): software
  @_id: "ID1"
  @enabled: true
  @{
    priority: 100,
    tenant: ["t1",   "t2", "t3"],
  }@
```

Let’s searchandize!

Let’s go!

- Our search
- We’ll write c
- Anyone can
- Since each n need for offline testin
- We can stor
- Rules are ec ntation
- There will or

….5 years later

- Thousands of rules exist
- We don’t always know who created them or why
- Some rules conflict with each other
- People who wrote rules have left the company
- The spr
- We can
- We don’t know how rules interact with everything else (index-time synonyms, automatic filtering, vector search…)
- Our rules framework is old, unmaintained and doesn’t work very well with the latest version of our search engine

Too much searchandizing!

How we can do better

Good practice

- Use an existing framework, don't build your own
  - Querqy & SMUI (open source, used by Otto, Walmart and many others, works with Solr, Elasticsearch, OpenSearch and others)
  - Elastic's Governed Search Patterns (only available to consulting customers) or Vespa's Rules
  - Commercial e-commerce engines (e.g. Algolia) usually have similar features
- Create a proper process
  - Ask the questions: is this a real user query? who can create a rule? What is the likely impact? Should the rule be permanent? Is there a better solution?
  - Document every rule properly: what does it fix? Who created it and why? How was it tested?
  - Housekeeping: review old rules periodically and remove those that are unused/irrelevant
- Logging and explainability
  - You should be able to see when a rule is triggered
  - Create dashboards to show overall statistics
  - Use debug PLPs to show when a result is there because of a rule

Testing

- Test the rule against known user queries
- If possible run offline tests, calculate metrics etc.
- BUT…Testing should not be a burden - the power of searchandizing is that it can fix issues on production very quickly
- A/B tests should not be necessary

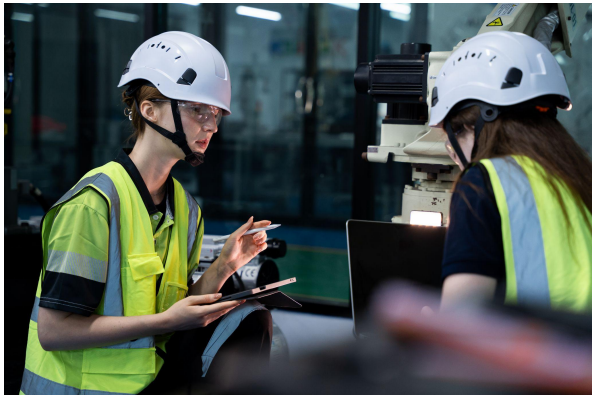
Who should do it?

- Search Product Managers
  - Sit between commercial requirements and the search technical team
  - Have access to analytics to see what needs fixing
  - BUT…they may not have domain knowledge
- Category Managers
  - They know their customers and their products
  - BUT…they may need training to understand how search works with rules
  - Process is key here
- Search engineers
  - Should be working on the core algorithm instead
  - May already be very busy - and may not have domain knowledge

Who should do it?

- Search Product Managers
  - Sit between commercial requirements and the search technical team
  - Have access to analytics to see what needs fixing
  - BUT…they may not have domain knowledge
- Category Managers
  - They know their customers and their products
  - BUT…they may need training to understand how search works with rules
  - Process is key here
- Search engineers
  - Should be working on the core algorithm instead
  - May already be very busy - and may not have domain knowledge

Examples from the real world



RUBIX

Thankyou Shaun, Birgita & Charline for explaining how you do it!

- Over 5 years in production
- Rules inspired by:
  - Promotions created by category managers
  - Trading reports showing underperforming queries
  - Zero result reports
  - LLM-as-a-judge showing low relevance
- Checklist is used to make decisions about rules
  - Will this affect one or all customers?
  - Is it actually a problem searchandizing should/can fix? (e.g. not just ‘no stock’)
  - Is the query something real users type?
  - Balance around fine-tuning and ongoing maintenance
- Sometimes searchandizing is the quickest short-term fix

RUBIX

- Internal guidance written based on training given by OSC and knowledge of Rubix' Solr setup
- SMUI/Querqy provides rule interface and rewriting facility - www.querqy.org
  - Around 700-1000 rules per territory
  - Rules can be pushed straight to production - fix issues in minutes
  - Tags are used for seasonal rules (e.g. "summer", "winter")
- Known issues
  - Some e-commerce engines can set results at fixed positions - we can't do this with boosting
  - Ideally we would test more deeply (Quepid, NDCG changes etc.) but usually no time
  - Sometimes queries with multiple meanings cause issues
  - There can be hard to predict cross-category impacts
  - With the best intentions, you can break things
  - No process for expiring rules (although seasonal tags help)
  - No full-time searchandizer roles so time can be tight
  - Some customers have their own catalogue but rules work the same for everyone - rules per customer would be a great thing to have

RUBIX

- What we have learned
  - We can only do what we can - not force users to buy things - there's no guarantee of commercial success!
  - Relevance is subjective and each stakeholder may have a different view
  - Successful Rules have helped with internal adoption of searchandizing
  - One bad result can be interpreted as 'our website is broken!' - we can fix this
  - Searchandizing brings people together for honest conversations about how to improve search - and to discuss any necessary compromises

Check out OSC's case study for more background
https://opensourceconnections.com/case-studies/rubix/

Takeaways

- Searchandizing is a powerful way to fix bad queries in minutes
- Without a solid process it can create more problems
- Use this process to develop solid, data-driven arguments for when to use (and not use) searchandizing
- Make sure you know when searchandizing has been used
- Use an established platform rather than building your own
- Use searchandizing to open up honest conversations about how to fix search - it should be a collaborative process

References & Attributions

1. Querqy.org - Querqy & SMUI, open source query rewriting
2. https://www.elastic.co/search-labs/blog/ecommerce-search-governance-improve-retrieval - good series of blogs on Elastic's tools
3. https://www.youtube.com/watch?v=dtzSt6RTpmI Shaun Brazendale from Rubix on a panel talking about how they adopted Querqy
4. https://www.youtube.com/watch?v=8iyNppg1UvI Optimising Search Relevance with Querqy
5. https://www.algolia.com/doc/guides/managing-results/rules/rules-overview Algolia's rules feature
6. https://docs.vespa.ai/en/reference/querying/semantic-rules.html Vespa rules (note these require a redeploy, but the commercial version of Vespa can do this live on production)

Images from Vecteezy, created by member Angga Afin and Factory Workers Stock photos

Thank you for listening!

Any questions?

www.thesearchjuggler.com

See you at Vespa AI Live!