# Hybrid Search Lessons Learned

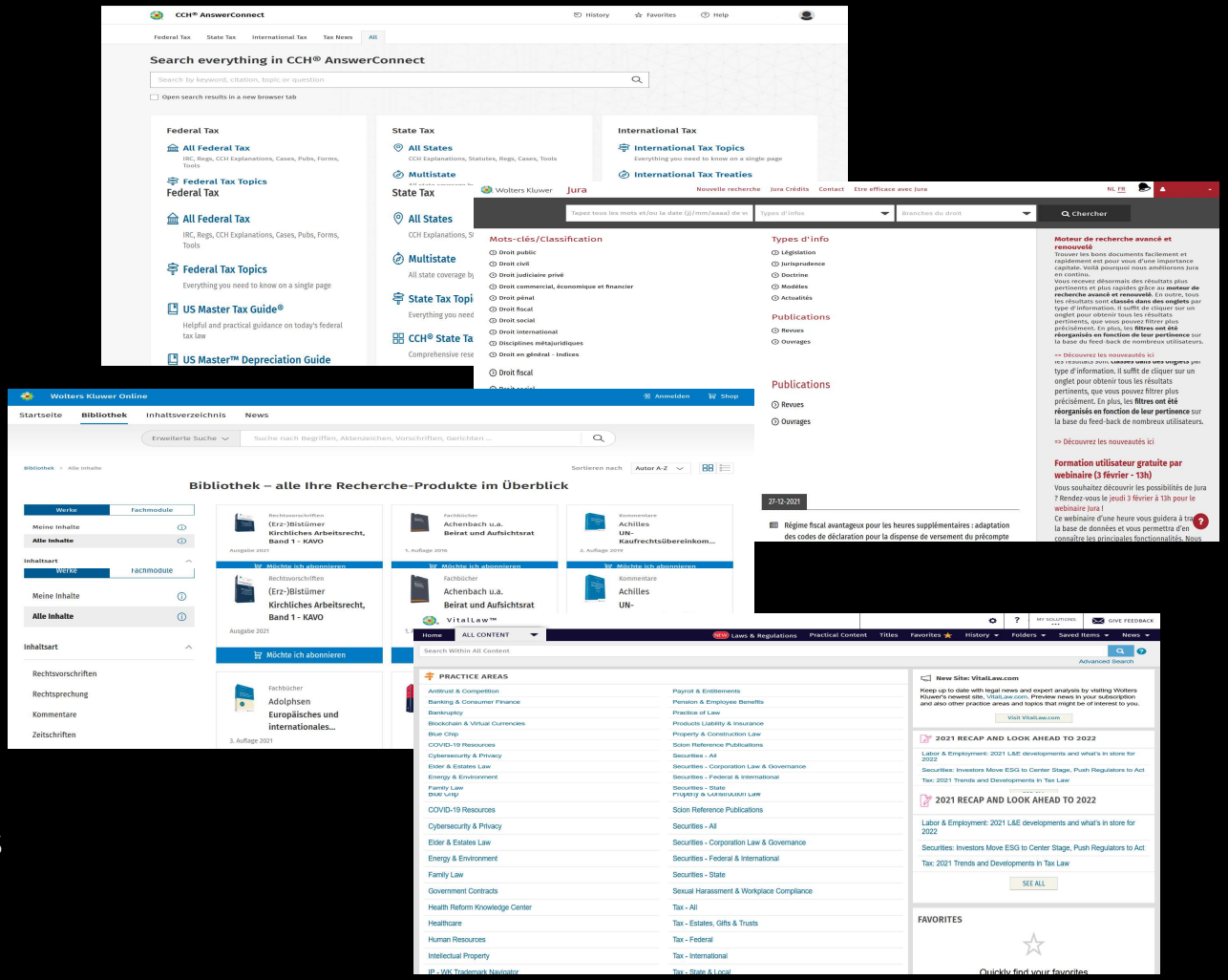
Mohit Sidana

Tom Burgmans

Haystack EU 2025

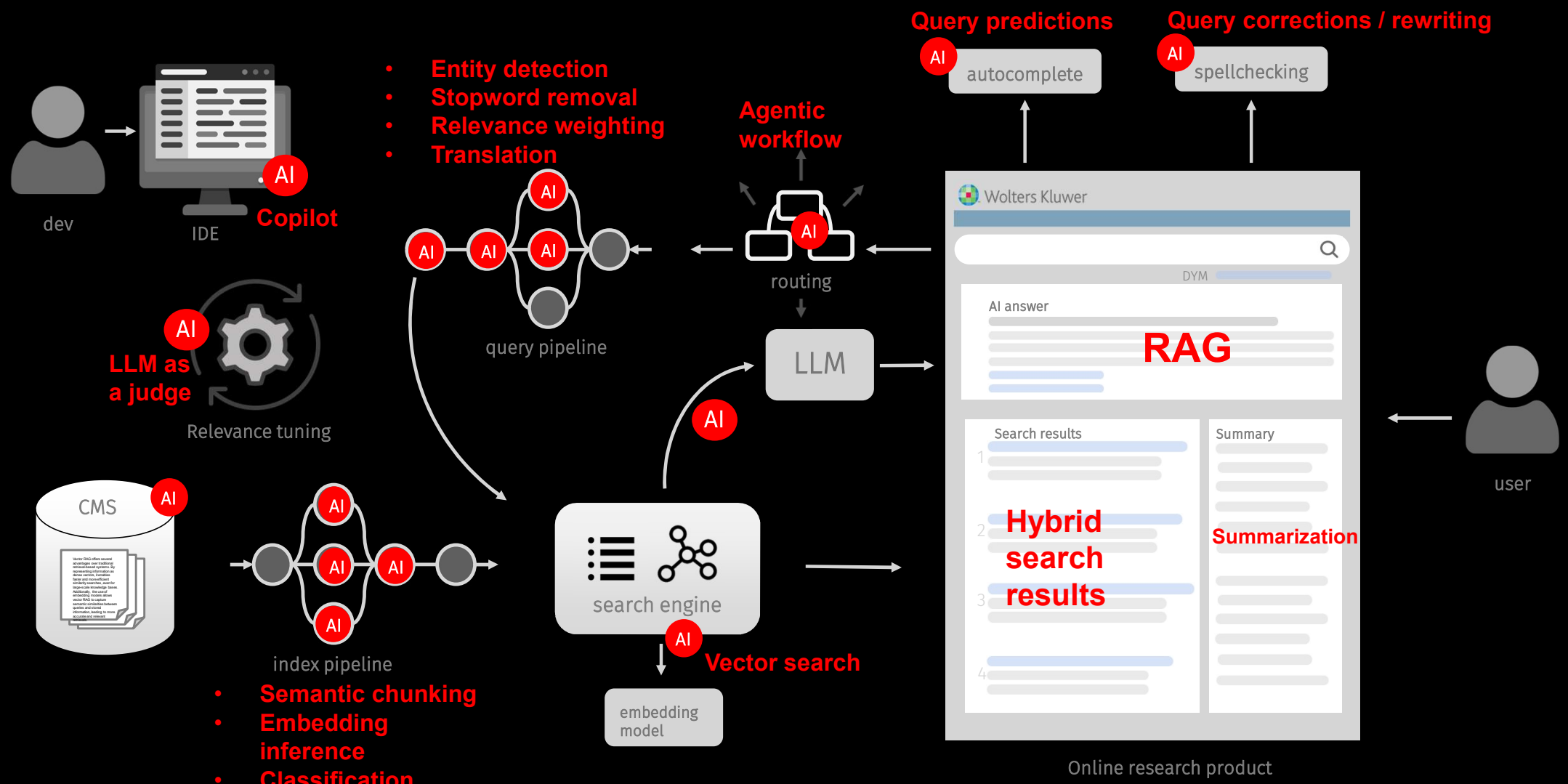
# Wolters Kluwer *“When you have to be right”*

Information services provider for experts in legal, business, tax, accounting, finance, audit, risk, compliance, and healthcare


- 21600 Employees
- Active in over 180 countries
- Mostly subscription based online products

# AI-ification of the Search Domain



# Researching query latency of vector search in Solr at scale

## Index Segments

Minimizing the number of index segments reduces overhead, enhancing the query performance.

## K-Value

High k-values directly increase query time; select only those essential for relevance.

## Vector Dimensions & Quantization

Dimensionality reduction with Matryoshka embeddings preserves key data while saving space. Scalar quantization reduces the vector precision to speed up the queries and reduce memory usage.

## Filters

Search filters can significantly increase query latencies due to processing overhead.

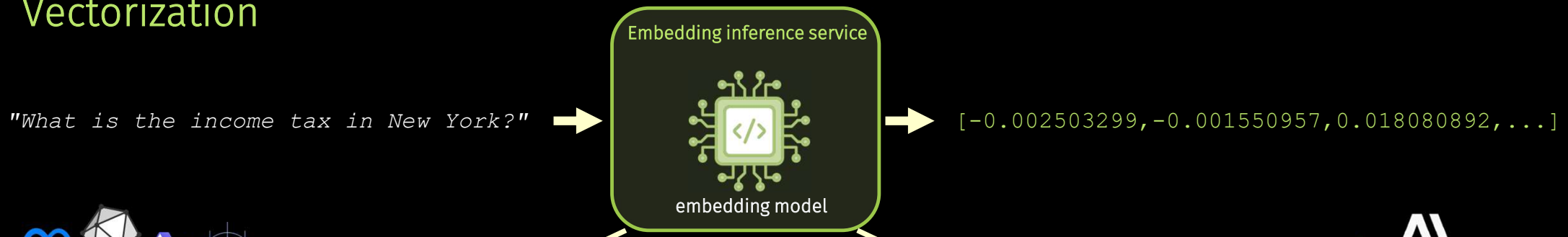
## Vectorizing content

Another bottleneck might be the external vectorizer.

## Dot Product Similarity

For efficient scoring calculations, use dot-product similarity. Don't forget to normalize vectors to unit length 1

# Vectorization




Self-managed

- Use **GPU** over CPU
- Have enough **VRAM** to hold the complete model + 10-20% extra
- **Smaller** models are faster
- **Low dimensional** models are faster
- Models could be **quantized** too. Quantized models are faster
- Convert models to **ONNX**
- **Short input texts** are faster
- Embedding inference service should be **closely hosted** to other connected services to limit network latency

- **Costs**: hosting infrastructure and operations
  - Efficient for **high volume** tasks

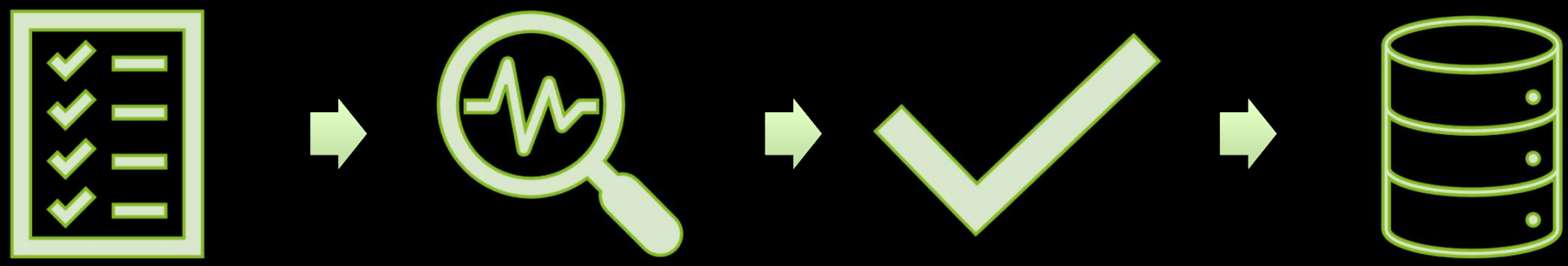
From a 3rd party

- Test the **throughput**! Understand what throttles it
- For more throughput capacity, **multiple deployments** in different regions (+ load balancer) may be needed
- Models from same provider could widely differ in **quality**
- Quantization and truncation could be done at **service side**
- Choose a **region close** to where connected services are hosted

- **Costs**: fee per M tokens
  - Efficient for **low volume** tasks

# Optimizing Indexing : When to Re-use Vectors

- Vectorization is computationally intensive
- Re-use vector embeddings to save resources
- Reduce latency where system need to handle frequent document metadata updates in the content
- Chunked text remains unmodified during metadata updates



Identify metadata updates to determine vector reuse opportunities.

Verify text changes swiftly using checksums or hashes.

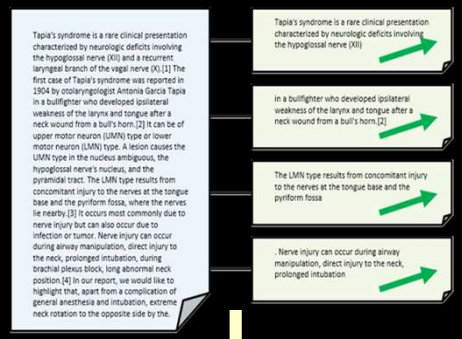
Decide whether to reuse existing vectors or Recompute.

Update vector store with reused or recomputed vectors accordingly.

# Setting up a multi-purpose Hybrid Search Index

## Preparing the content set

Nested document hierarchy with parent and child documents




```
<doc>
    <field name="id">doc-1</field>
    <field name="type_s">parent</field>
    <field name="title"><![CDATA[Tapia's syndrome title field]]></field>
    <field name="body_en"><![CDATA[Tapia's syndrome is a rare clinical presentation characterized by neurologic deficits involving
    <field name="asset_status_s_lower"><![CDATA[active]]></field>
    <field name="epubdate_dts"><![CDATA[2022-04-01T00:00:00Z]]></field>
    <field name="publicationtype_ss_lower"><![CDATA[Letter]]></field>
    <doc>
    <doc>
    <doc>
        <field name="id">doc-1.3</field>
        <field name="parentDoc_s">doc-1</field>
        <field name="type_s">child</field>
        <field name="chunkoffsets_s"><![CDATA[8130-12130]]></field>
        <field name="vector_medium"><![CDATA[-0.01133619]]></field>
        <field name="vector_medium"><![CDATA[-0.02574570]]></field>
        <field name="vector_medium"><![CDATA[-0.03118439]]></field>
        <field name="vector_medium"><![CDATA[-0.03804982]]></field>
        <field name="vector_medium"><![CDATA[-0.01053053]]></field>
        <field name="vector_medium"><![CDATA[-0.04007610]]></field>
        <field name="vector_medium"><![CDATA[-0.04348175]]></field>
        <field name="vector_medium"><![CDATA[0.04725934]]></field>
        <field name="vector_medium"><![CDATA[-0.01782340]]></field>
        <field name="vector_medium"><![CDATA[-0.00262971]]></field>
```

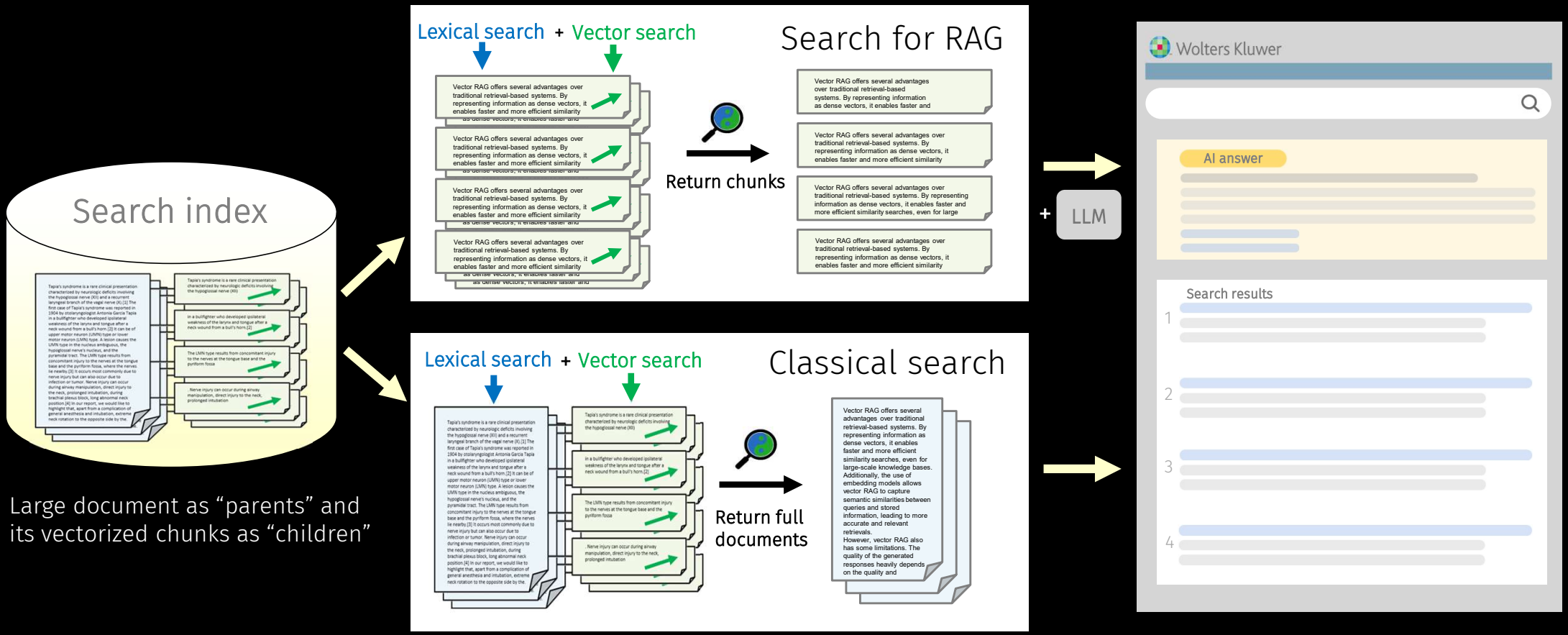
Parent document

Keyword search fields

Child documents, one for each chunk

Vector of the chunk

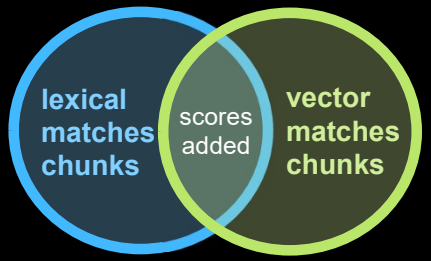
# A nested index could serve multiple hybrid search use cases



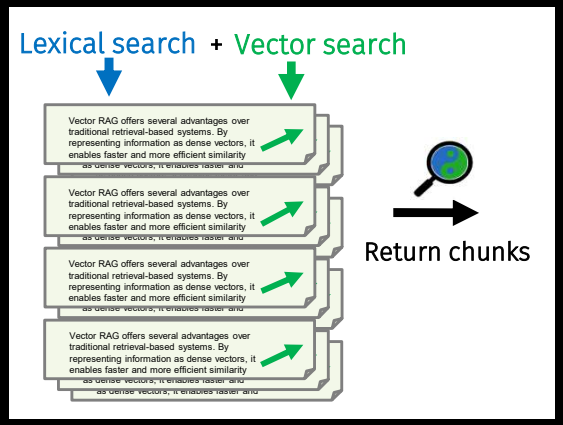

Large document as “parents” and its vectorized chunks as “children”

Online research product

# Hybrid search Solr syntax



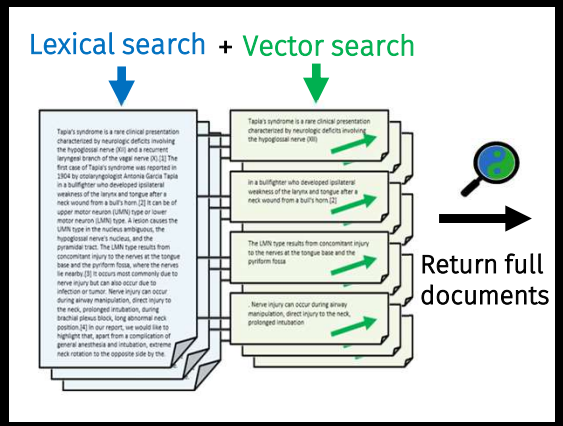

## Search for RAG

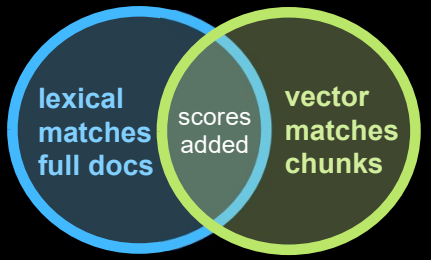



```
params:{
  uf:"* _query_",
  q:"{!bool filter=$hybridlogic must=$hybridscore}",
  hybridlogic:"{!bool should=$kwq should=$vectorq}",
  hybridscore:"{!func}sum(product($kwweight,$kwq),product($vectorweight,query($vectorq)))",
  kwq:"{!type=edismax qf=\"chunk_body\" v=$qq}",
  qq:"What is the income tax in New York?",
  vectorq:"{!knn f=vector_field topK=10}[-0.002503299,-0.001550957,0.018080892,...]",
  kwweight:1,
  vectorweight:4
}
```

- allow embedded Solr queries
- OR-relationship between lexical search & vector search
- sum the scores
- typical lexical search
- balance the impact of keyword matches vs vector matches
- vector representation of the query (externally vectorized)

## Classical search




```
params:{
  uf:"* _query_",
  q:"{!bool filter=$hybridlogic must=$hybridscore}",
  hybridlogic:"{!bool should=$kwq should=$vectorq}",
  hybridscore:"{!func}sum(product($kwweight,$kwq),product($vectorweight,query($vectorq)))",
  kwq:"{!type=edismax qf=\"full_doc_body full_doc_title^3\" v=$qq}",
  qq:"What is the income tax in New York?",
  vectorq:"{!parent which=\"type_s:parent\" score=max v=$childq}",
  childq:"{!knn f=vector_field topK=10}[-0.0034859276,-0.028224038,0.0024048693,...]",
  kwweight:1,
  vectorweight:4
}
```

- block join: vector search children, return parents

# Relevancy tuning

Blending lexical results with vector results is like mixing water and oil

| | Lexical search | Vector search |
|---|---|---|
| Scoring algorithm | TF-IDF (BM25) | Vector distance (i.e. cosine angle) |
| Debug score EXPLAIN | Yes | No |
| Effect of boosting | On the entire result set | Only the top *k* results |
| Number of results | Between 0 - all documents | Between 0 – *k* (in most cases exactly *k*) |

# Relevancy tuning

Tuning relevance for LLMs (RAG) is not equal to tuning for humans

| | For humans | For LLMs |
|---|---|---|
| Unit of Retrieval | Full documents | Document chunks |
| Consumption Mode | Scroll, KWIC, Facets, Paginates | Top N chunks |
| Result list importance | Ordering (*precision* of the top results) | Coverage (*recall* in the top N) |
| Tuning goals | Engagement, diversity | Informativeness, factuality |

Balancing weights in Hybrid search is dependent on the nature of the query

Skip vector search with:

- "*" search or wildcarded queries
- Boolean constructions
- Explicit phrase search
- Fielded searches
- Proximity queries

## Hybrid search

Lexical match score

Vector match score

Citation

Case nickname

Multi-lingual

Typos

Case summary

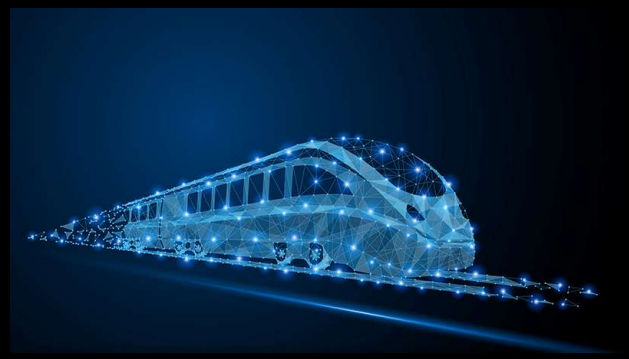
# Where we'll expect a lot of 'tuning' energy will be spent

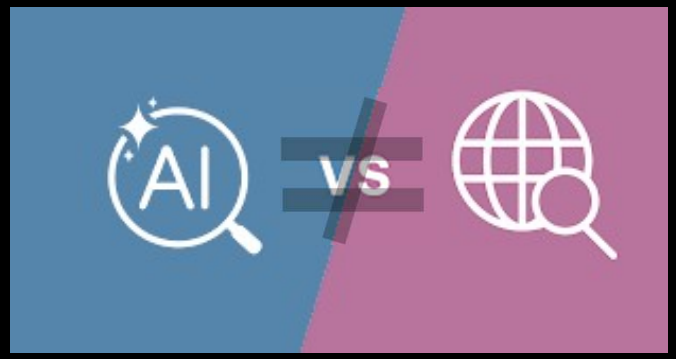
User's query
+ history
+ context

The Fabulous Agentic Intelligent Query Interpreting and Semantically Smart Omni-Dimensional Hyper-Adaptive Cognitive-Neural Orchestrator of Dynamic Search Syntax Generation (or in short: FAIQISSODH-AC-NODSSG)

- (Hybrid) search syntax
- Boosting weights
- Auto filters
- Rewriting
- Expansions
- LLM prompt

# Enrolling hybrid search in a large organization: challenges







*"Will AI replace search?"*

*"Search is not developing anymore"*

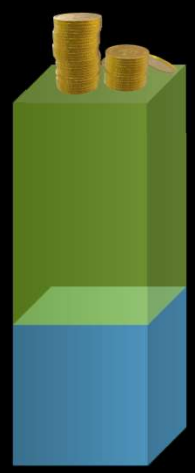
*"Why not just train a model?"*

*"Future AI systems won't need search for RAG"*

# Enrolling hybrid search in a large organization

1. Create interest & awareness
   - Emphasis on the benefits
   - Short videos
2. Prototype
   - Real use cases
   - Shareable
3. Education & Transparency
   - Deep(er) dive in the pro's & con's
   - Cost impact
4. PoC
   - Integrated in product
   - Go/No-Go moment
5. Productionization
   - Start small
   - A/B
6. Adoption & Enablement
   - Measure usage
   - Keep evolving

# Cost factors lexical -> hybrid

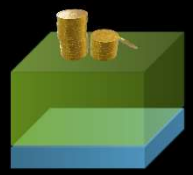


Infrastructure
- Memory
- Disc

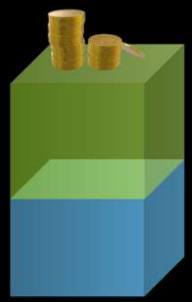
Vectorization

Self-hosted:
- infrastructure
- operations

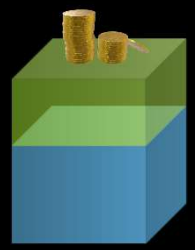
Commercial:
- Fee per token



Storage

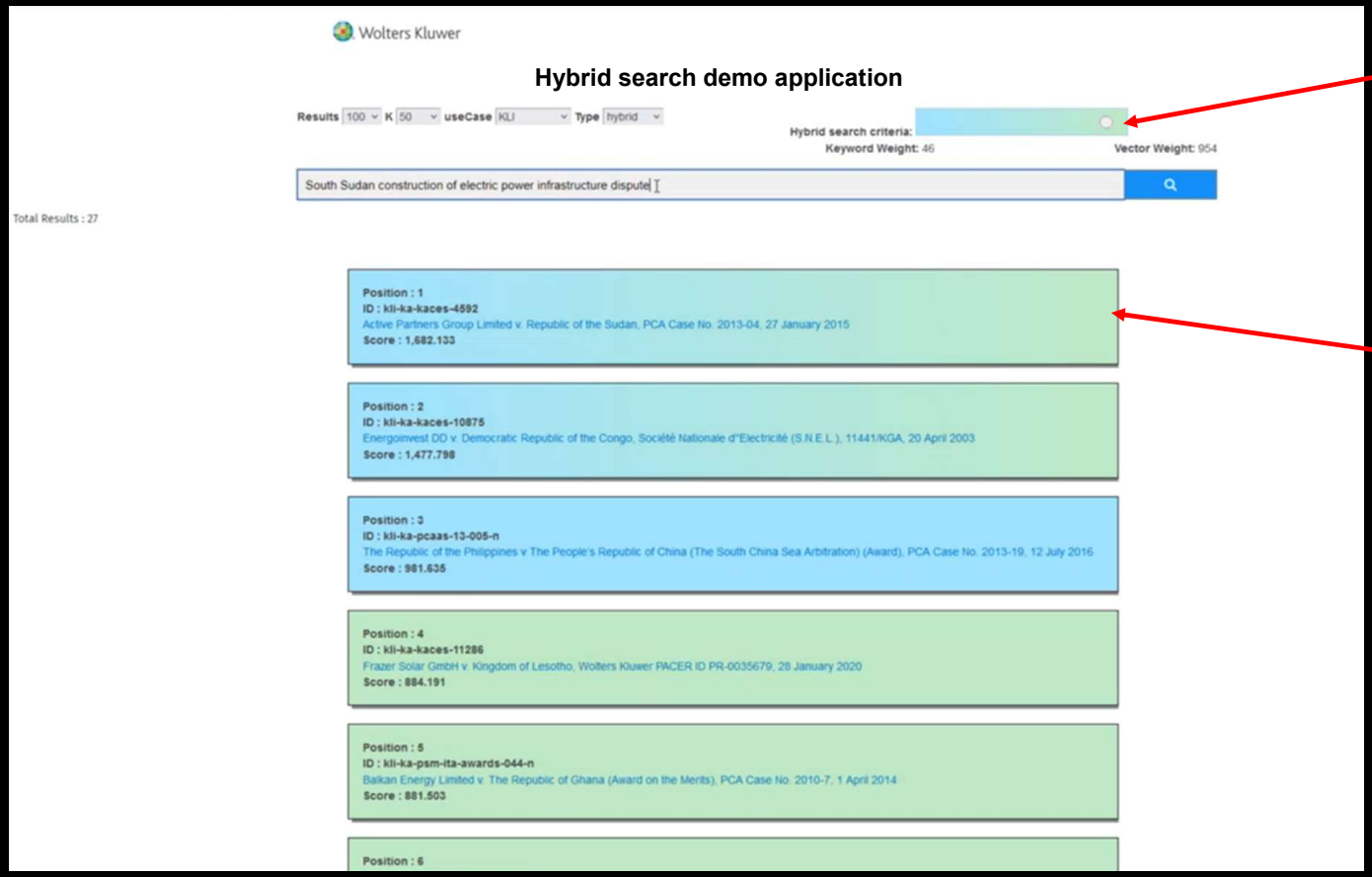


Development



Tuning

# Hybrid search prototype




Manual slider to balance the impact of lexical search vs vector search

Background colors indicate:

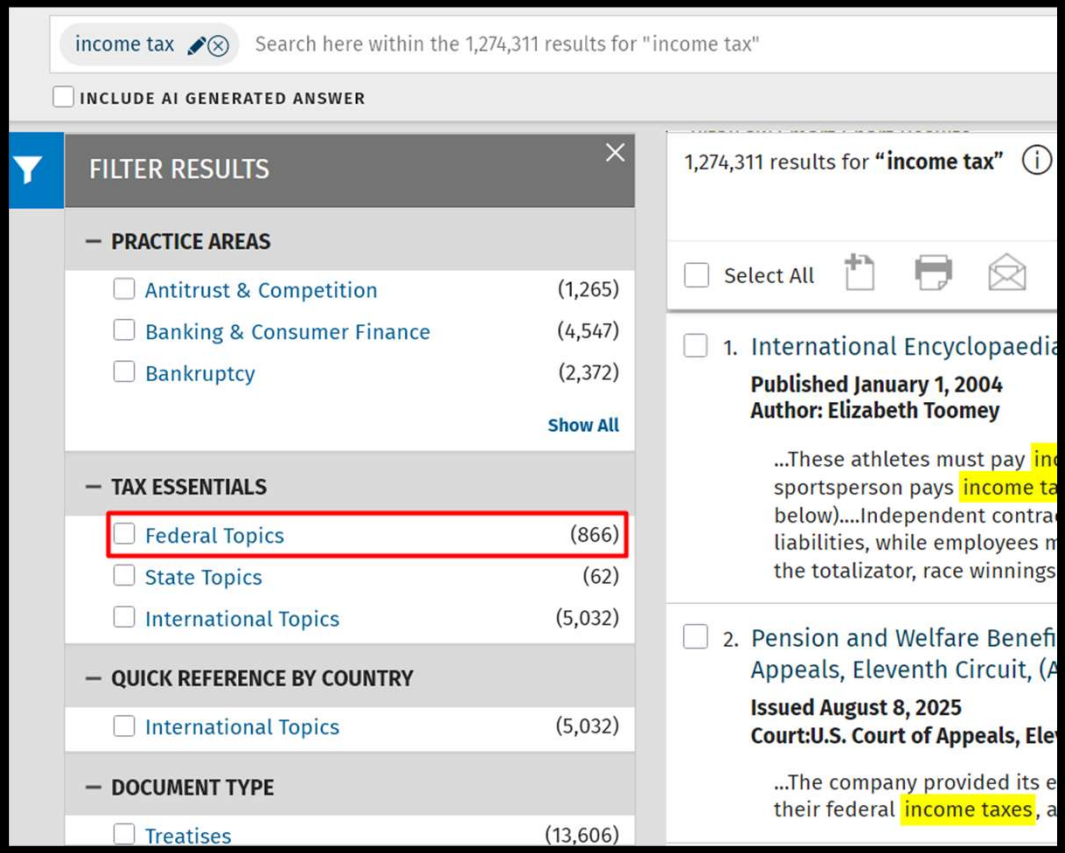
- Lexical match
- Vector match
- Hybrid match

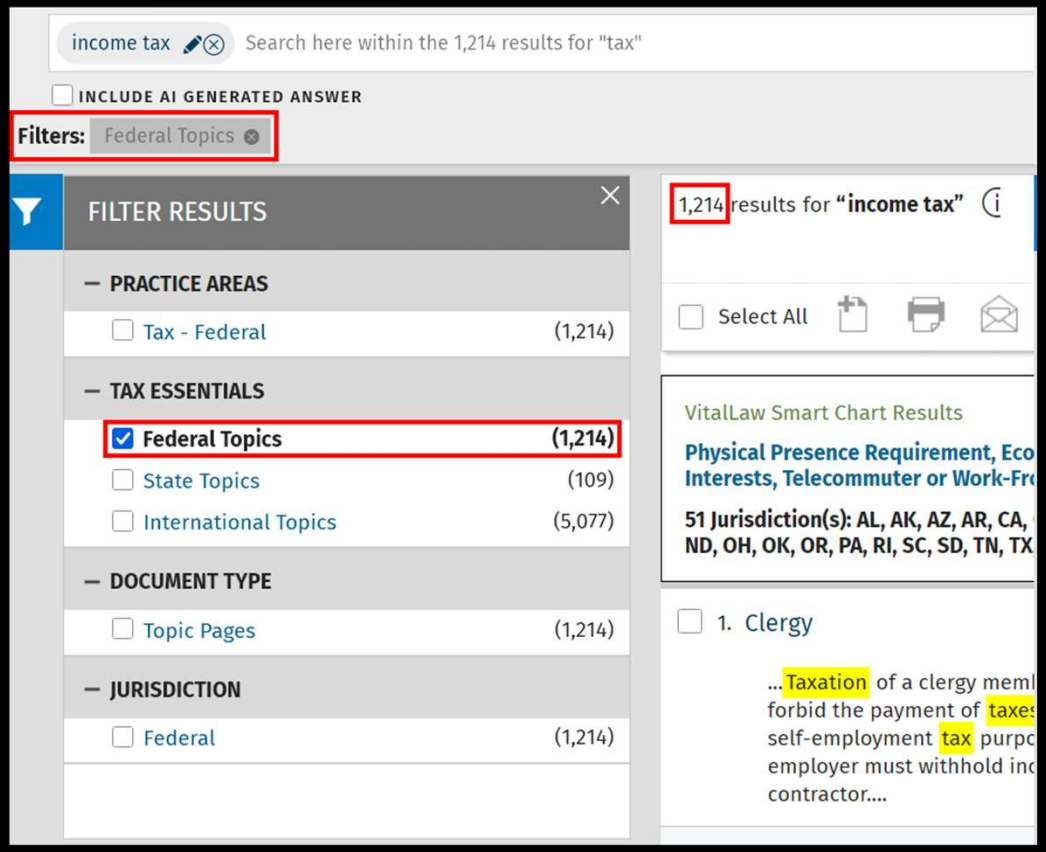
Tip: how to let Solr tell with what search technique it found its results (see syntax slide 9 ):

```
fl:…,keywordmatch:exists($kwq),vectormatch:exists($vectorq),…
```

# Hybrid search requires explaining

Warning: counts for facets/total results may behave unintuitive



Wolters Kluwer

# Thank you!

# Questions?

Or contact us later:


Mohit Sidana
Search Architect
Wolters Kluwer

Tom Burgmans
Principle Product Software Engineer
Wolters Kluwer