

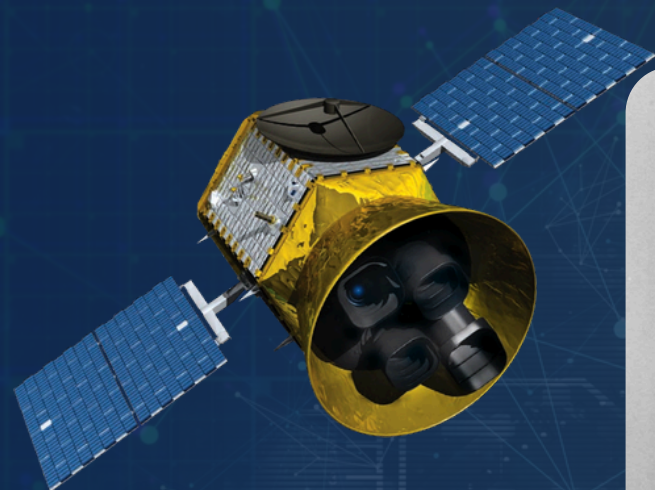


Arcturus Labs

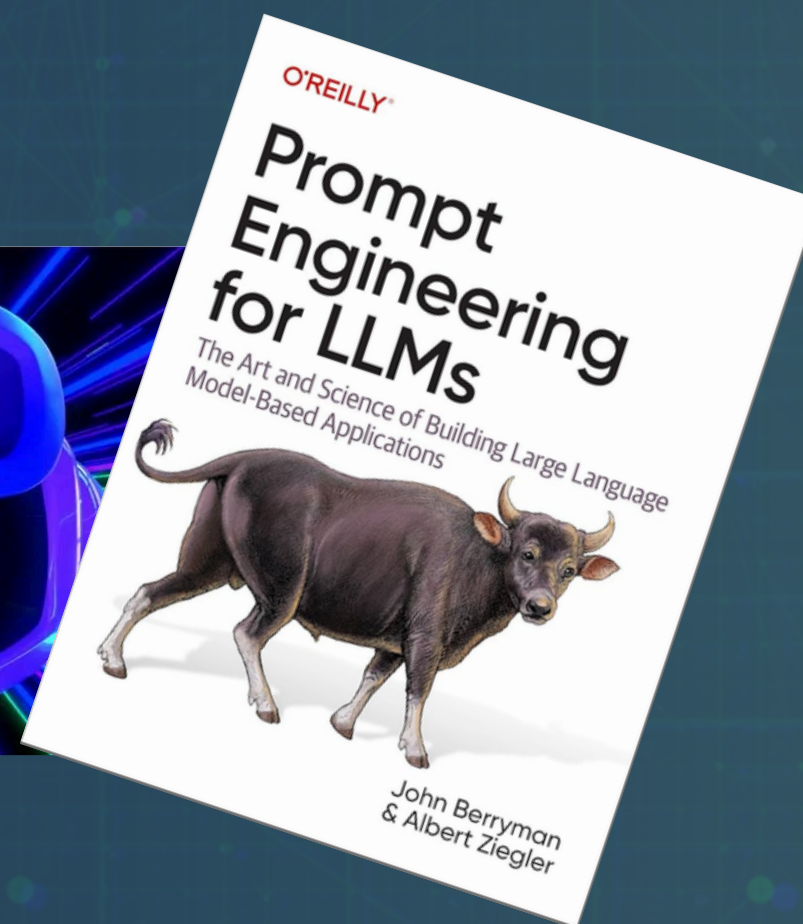
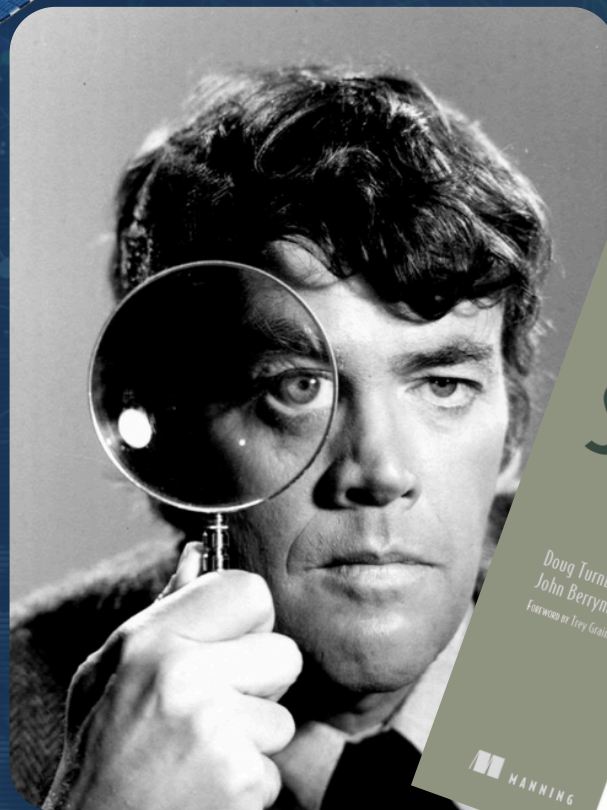
# **Lexical Love: Rediscovering the Power of Lexical Search in RAG**







# Hi, I'm John Berryman



Arcturus Labs

[www.arcturus-labs.com](http://www.arcturus-labs.com)







**Caveat:**

**Lexical search is my  
hammer, and the world  
is my nail.**







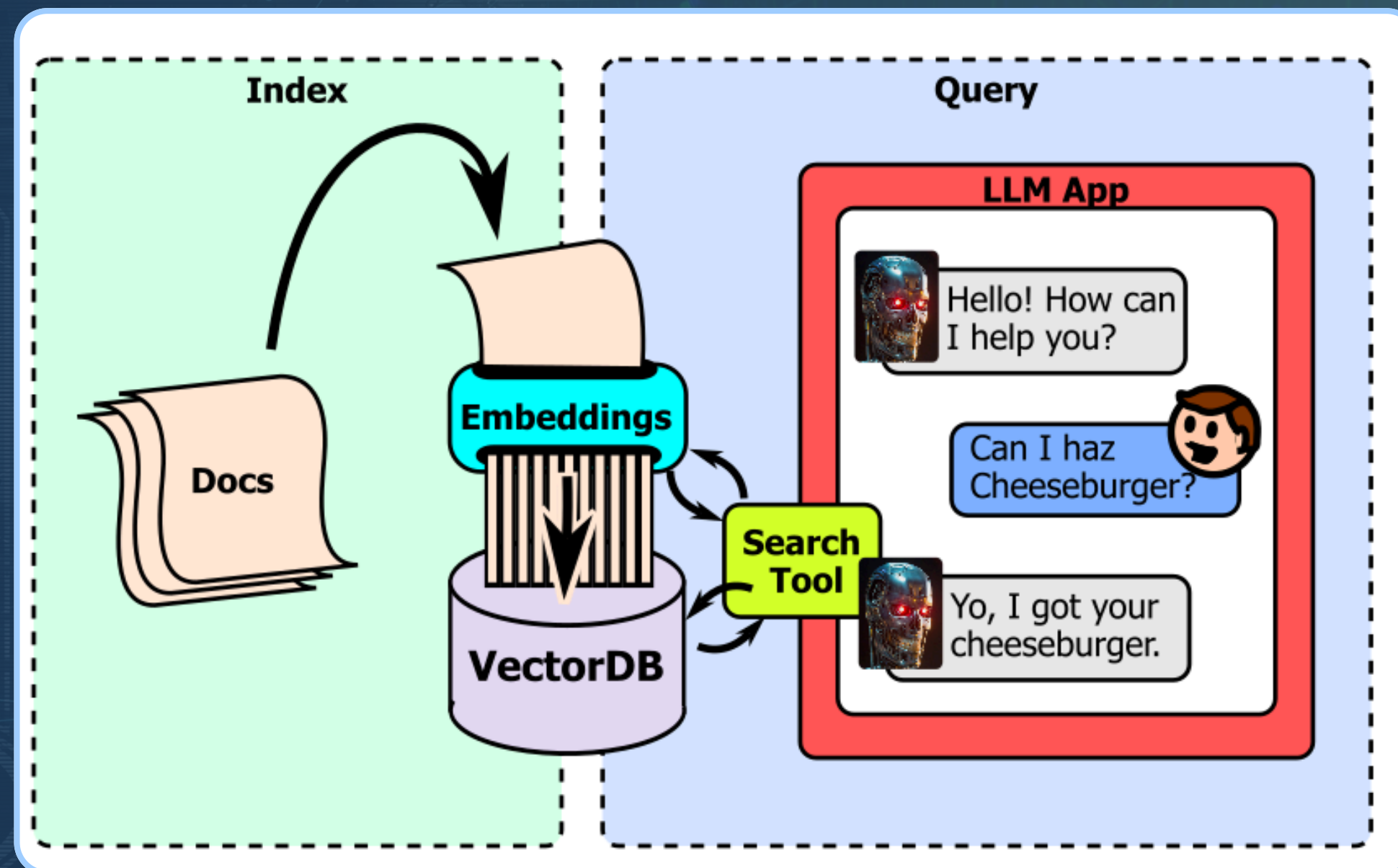
# Semantic Search – “It’s how you do RAG”

## “It’s easy”

- Chunk the documents
- Use BERT-like model to embed as vectors
- Store in a vector store
- At query time you embed the query and retrieve the nearest docs

## “It’s cool”

Unlike “old fashioned” lexical search which uses exact token matching, Semantic Search matches based on *meaning*!



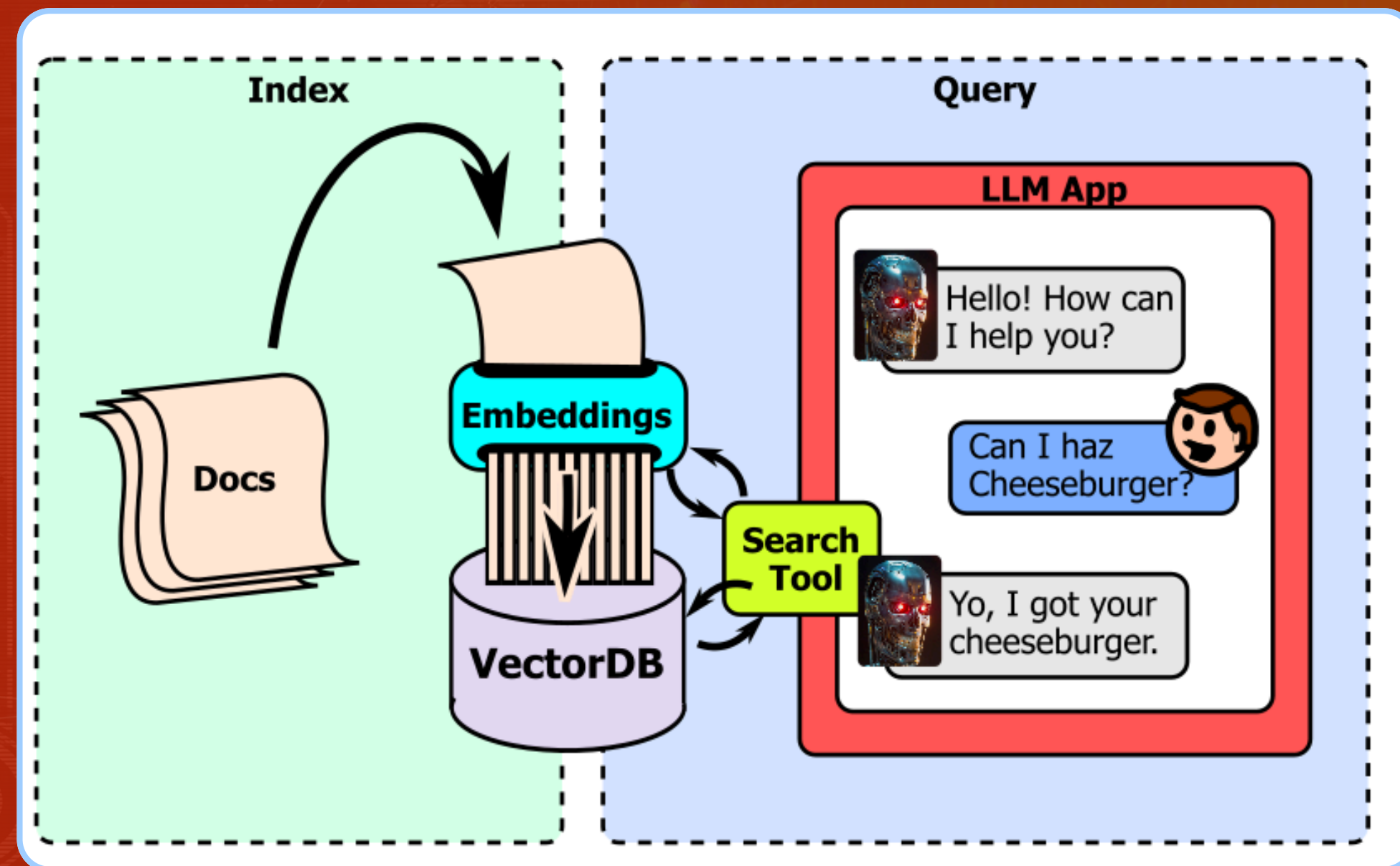




# Semantic Search - “It’s how you do RAG”

## BUT *there are challenges*

- Impossible to find
  - Exact term matches (e.g. ids, people names)
  - Phrase matches
  - New jargon introduced since training
- Fixing relevance problems is VERY involved
- Filtering is clunky - hard to “slice and dice” data set

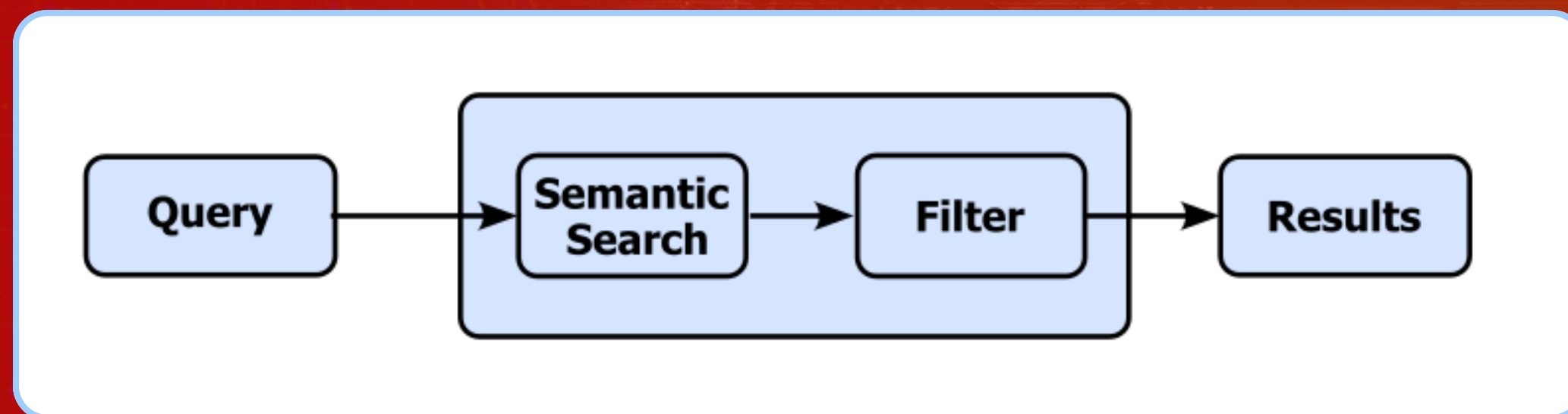






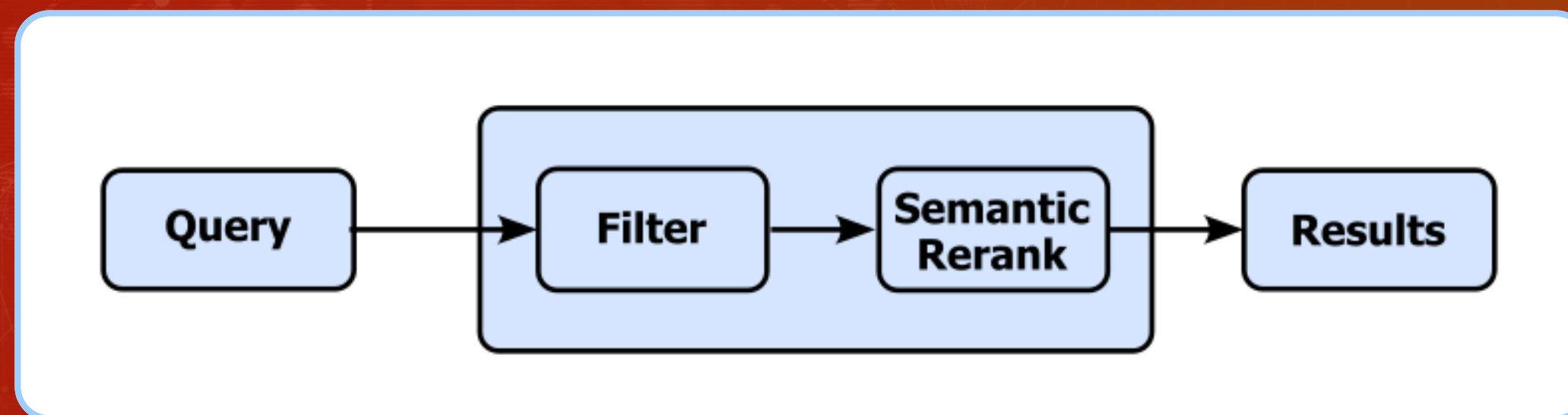
# Semantic Search - "It's how you do RAG"

How do you filter?



your filter might remove all the matches

reranking is brute force







# IN THIS TALK:

- “Body slam” introduction to lexical search
- Example with RAG underscoring benefits
- Discuss the problems
- Point toward a hybrid search







*(really friggin' fast)*

# Let's Learn Lexical Search!

- Indexing
- Searching
- Results







# Lexical Indexing

## Data Types

### Numbers:

- bool
- int
- float
- date/time

### Strings:

- keywords - indivisible strings
- text - will be processed into array of tokens

### Etc:

- Geo points
- Geo shapes
- Intervals of time
- compound types

← the star of the show

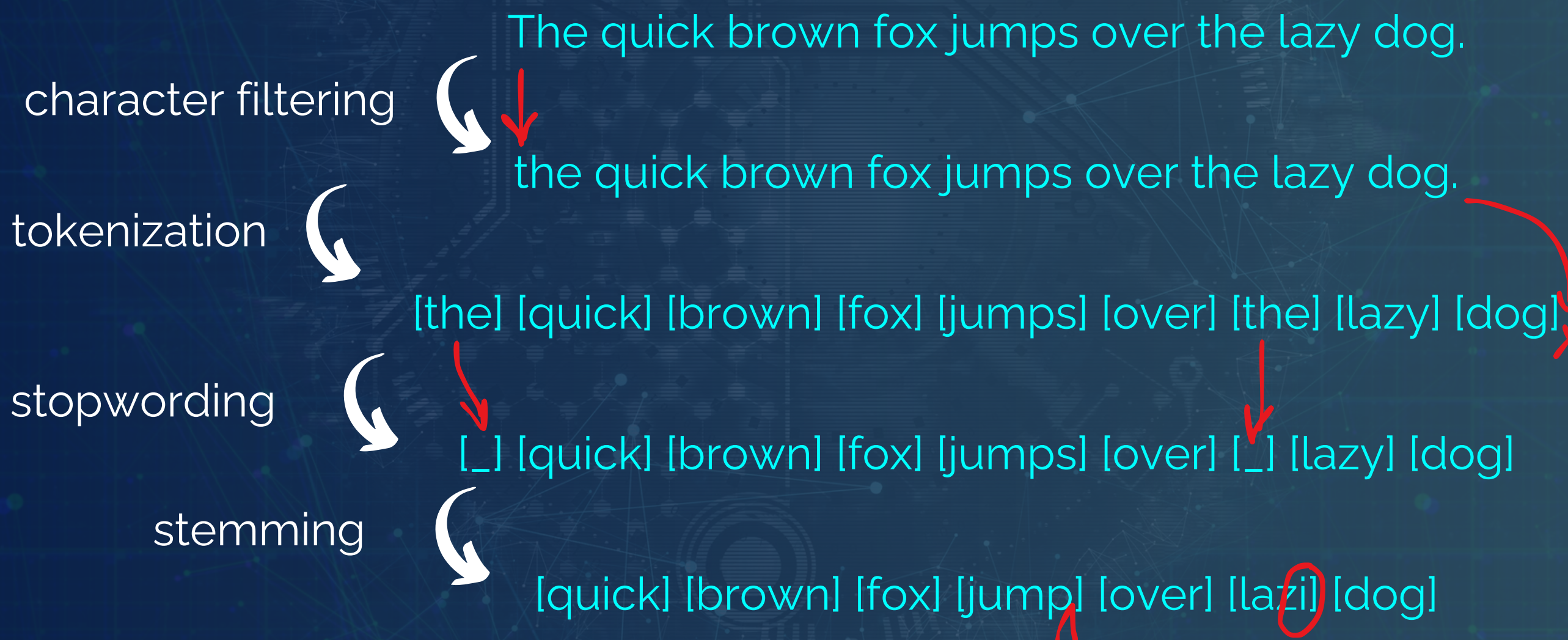






# Lexical Indexing

**Analysis** – converts a string into an array of normalized tokens







# Lexical Indexing

## Inverted Index

**Document 12:** [he] [was] [quick]

**Document 43:** [quick] [brown] [fox] [jump] [over] [lazy] [dog]

**Document 88:** [slow] [red] [fox] [ran] [under] [porch]







# Lexical Indexing

## Inverted Index

**Document 12:** [he] [was] [quick]

**Document 43:** [quick] [brown] [fox] [jump] [over] [lazi] [dog]

**Document 88:** [slow] [red] [fox] [ran] [under] [porch]

### the Inverted Index

[quick] → [3, 12, **43**, 67, 81, 92]  
[brown] → [7, 15, **43**, 56, 78, 84, 97]  
[fox] → [2, 18, 29, **43**, 88]  
[jump] → [5, 21, 34, **43**, 59, 90]  
[over] → [9, 22, 35, **43**, 51, 66, 73, 95]  
[lazi] → [1, 19, 27, **43**, 99]  
[dog] → [4, 20, 31, **43**, 50, 61, 72, 87, 100]







# Lexical Indexing

## Differences

Semantic	Lexical
docs are pre-chunked	no chunking
index is often larger than text	index is often smaller than text
data is opaque (vectors)	data is transparent (tokens)







# Lexical Search

## Fast Searching

### the Inverted Index

Find all "fox"  
documents

quick	→	[3, 12, 43, 67, 81, 92]
brown	→	[7, 15, 43, 56, 78, 84, 97]
fox	→	[2, 18, 29, 43, 88]
jump	→	[5, 21, 34, 43, 59, 90]
over	→	[9, 22, 35, 43, 51, 66, 73, 95]
lazi	→	[1, 19, 27, 43, 99]
dog	→	[4, 20, 31, 43, 50, 61, 72, 87, 100]

"brown" and "fox"  
documents

"fox" or "dog"  
documents







# Lexical Search

## Relevant Searching

### the Inverted Index

quick → [3, 12, 43, 67, 81, 92]  
brown → [7, 15, 43, 56, 78, 84, 97]  
fox → [2, 18, 29, 43, 88]  
jump → [5, 21, 34, 43, 59, 90]  
over → [9, 22, 35, 43, 51, 66, 73, 95]  
lazi → [1, 19, 27, 43, 99]  
dog → [4, 20, 31, 43, 50, 61, 72, 87, 100]

} "brown" and "fox"  
documents

- Matching docs are scored based on ***TF\*IDF***\*

- It means:

$$\frac{\text{Term Frequency}}{\text{Doc Frequency}}$$

- Which *really* means

$$\frac{\text{num times the word appears in the doc}}{\text{num docs where the word appears}}$$

- With multi-field search, you can apply boosts and filtering

\*more accurately it's BM25







# Lexical Searching

## Differences

Semantic	Lexical
Searches by approx. nearest neighbor	Searches by matching tokens
Search and filter in different steps	Search by all fields simultaneously
Relevancy is distance (and that's it)	Can debug and apply boosts w/o reindexing





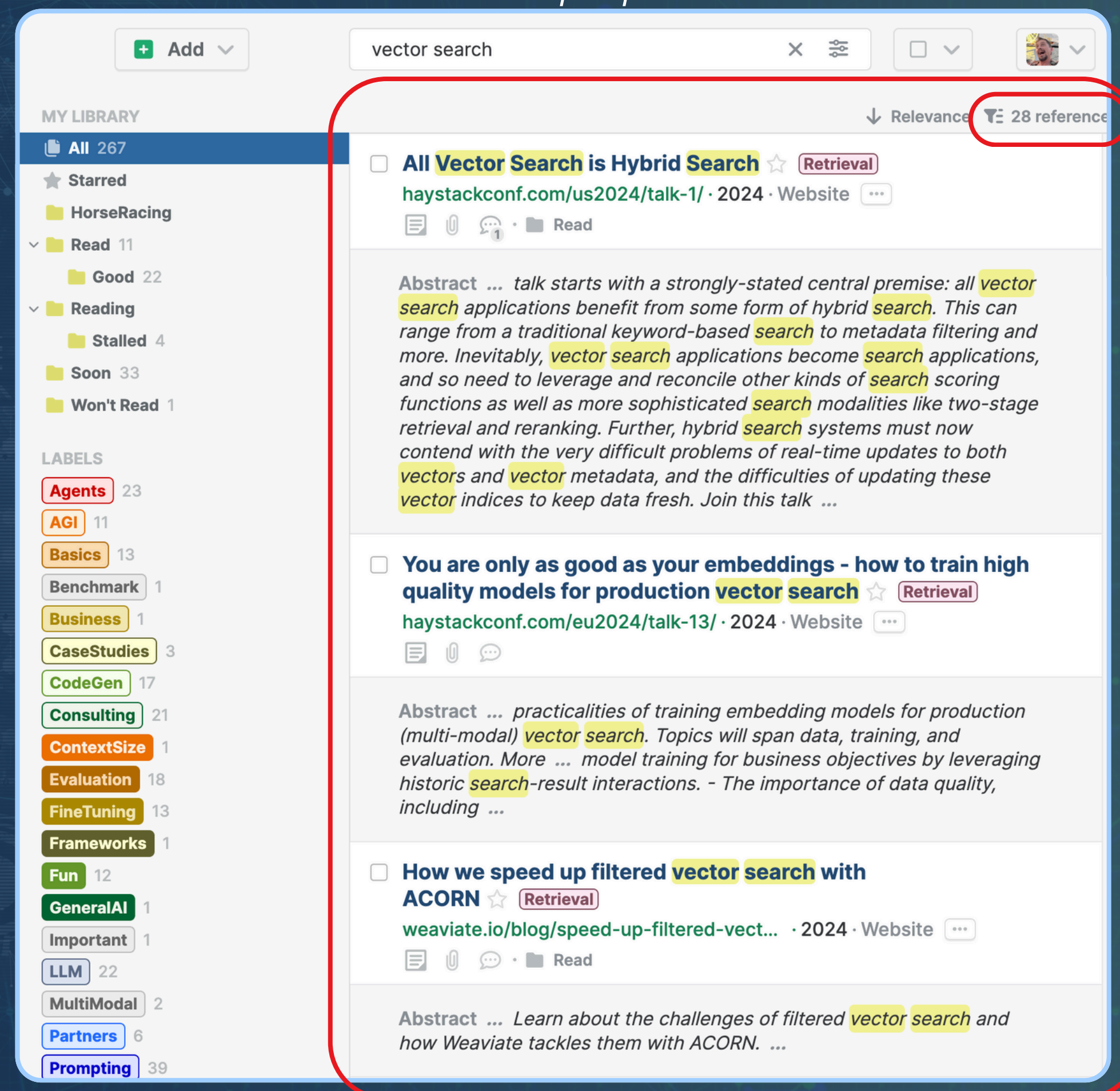


Arcturus Labs

# Lexical Results

# Result Set

- How many matching documents are in the set
- A list of the top N documents





# Lexical Results

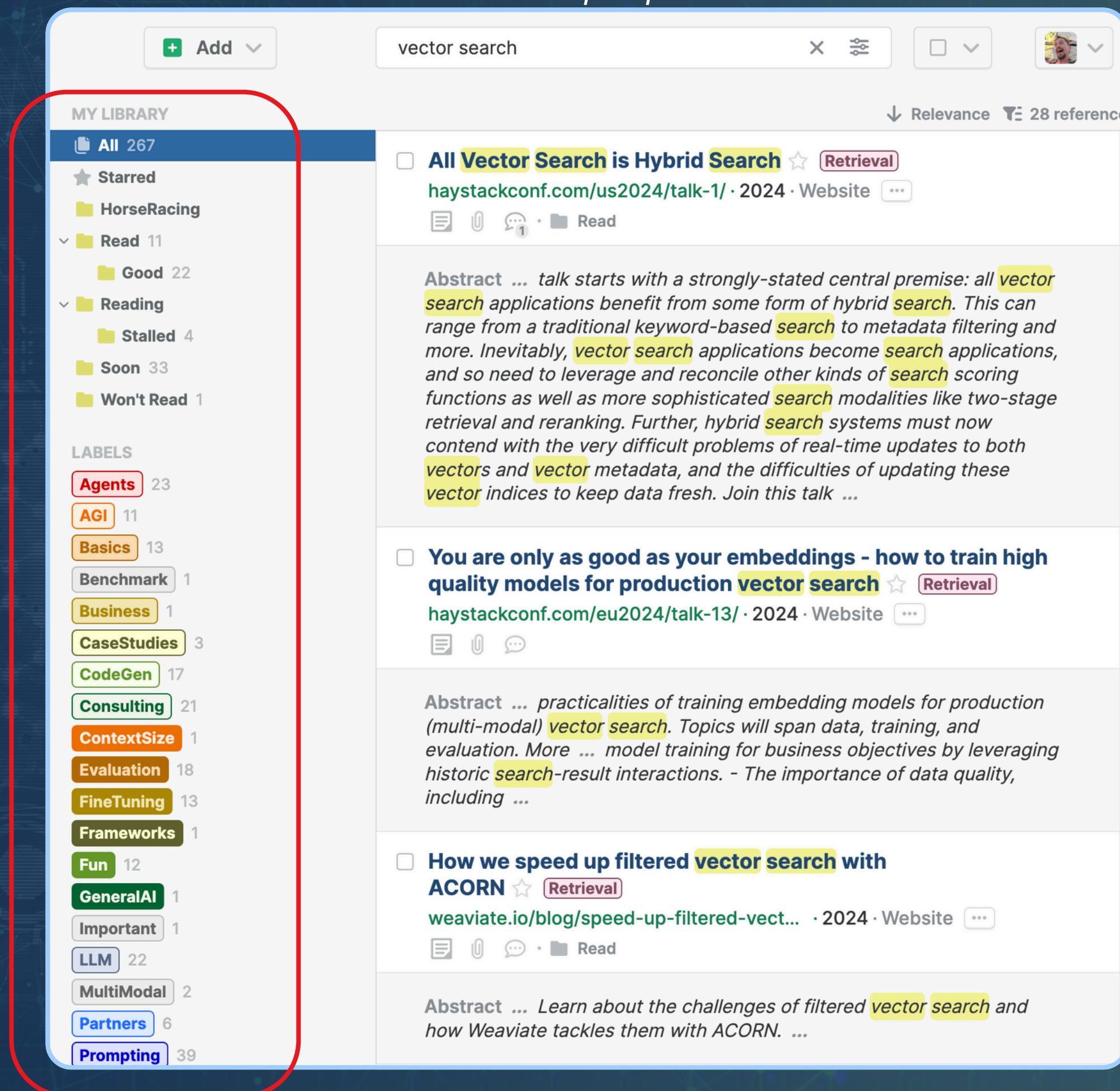
## Result Set

- How many matching documents are in the set
- A list of the top N documents

## Aggregate Analytics

Can summarize fields of all matching docs (not just the top N)

- average, max, min
- histograms, cardinality
- facet counts

The screenshot shows the Paperpile application interface. At the top, there's a search bar with the text 'vector search'. Below the search bar, the 'MY LIBRARY' sidebar is visible, containing a list of folders and labels. The main content area displays search results for 'vector search', showing three document entries with their titles, sources, and abstracts. The first result is 'All Vector Search is Hybrid Search' from haystackconf.com. The second is 'You are only as good as your embeddings - how to train high quality models for production vector search' from haystackconf.com. The third is 'How we speed up filtered vector search with ACORN' from weaviate.io. The interface includes various UI elements like a '+ Add' button, a 'Relevance' dropdown, and a '28 reference' indicator.

**MY LIBRARY**

- All 267
- Starred
- HorseRacing
- Read 11
  - Good 22
- Reading
  - Stalled 4
- Soon 33
- Won't Read 1

**LABELS**

- Agents 23
- AGI 11
- Basics 13
- Benchmark 1
- Business 1
- CaseStudies 3
- CodeGen 17
- Consulting 21
- ContextSize 1
- Evaluation 18
- FineTuning 13
- Frameworks 1
- Fun 12
- GeneralAI 1
- Important 1
- LLM 22
- MultiModal 2
- Partners 6
- Prompting 39

**vector search** [x] [filter icon] [dropdown icon] [user icon]

↓ Relevance 28 reference

- ☐ **All Vector Search is Hybrid Search** ☆ Retrieval  
haystackconf.com/us2024/talk-1/ · 2024 · Website ...  
[document icon] [link icon] [comment icon] [folder icon] Read
- Abstract ...** talk starts with a strongly-stated central premise: all vector search applications benefit from some form of hybrid search. This can range from a traditional keyword-based search to metadata filtering and more. Inevitably, vector search applications become search applications, and so need to leverage and reconcile other kinds of search scoring functions as well as more sophisticated search modalities like two-stage retrieval and reranking. Further, hybrid search systems must now contend with the very difficult problems of real-time updates to both vectors and vector metadata, and the difficulties of updating these vector indices to keep data fresh. Join this talk ...
- ☐ **You are only as good as your embeddings - how to train high quality models for production vector search** ☆ Retrieval  
haystackconf.com/eu2024/talk-13/ · 2024 · Website ...  
[document icon] [link icon] [comment icon]
- Abstract ...** practicalities of training embedding models for production (multi-modal) vector search. Topics will span data, training, and evaluation. More ... model training for business objectives by leveraging historic search-result interactions. - The importance of data quality, including ...
- ☐ **How we speed up filtered vector search with ACORN** ☆ Retrieval  
weaviate.io/blog/speed-up-filtered-vect... · 2024 · Website ...  
[document icon] [link icon] [comment icon] [folder icon] Read
- Abstract ...** Learn about the challenges of filtered vector search and how Weaviate tackles them with ACORN. ...



# Lexical Results

## Snippets

- Segments of text that contain the interesting phrases in context.
- It's like search-time chunking.



+

Add

▼

vector search

×

⚙

□

▼

▼

MY LIBRARY

📁

All

267

★

Starred

📁

HorseRacing

▼

📁

Read

11

📁

Good

22

▼

📁

Reading

📁

Stalled

4

📁

Soon

33

📁

Won't Read

1

LABELS

Agents

23

AGI

11

Basics

13

Benchmark

1

Business

1

CaseStudies

3

CodeGen

17

Consulting

21

ContextSize

1

Evaluation

18

FineTuning

13

Frameworks

1

Fun

12

GeneralAI

1

Important

1

LLM

22

MultiModal

2

Partners

6

Prompting

39

Relevance

28 references

□

All Vector Search is Hybrid Search

☆

Retrieval

haystackconf.com/us2024/talk-1/

· 2024 · Website

⋮

📄

📎

💬

📁

Read

Abstract ... talk starts with a strongly-stated central premise: all vector search applications benefit from some form of hybrid search. This can range from a traditional keyword-based search to metadata filtering and more. Inevitably, vector search applications become search applications, and so need to leverage and reconcile other kinds of search scoring functions as well as more sophisticated search modalities like two-stage retrieval and reranking. Further, hybrid search systems must now contend with the very difficult problems of real-time updates to both vectors and vector metadata, and the difficulties of updating these vector indices to keep data fresh. Join this talk ...

□

You are only as good as your embeddings - how to train high quality models for production vector search

☆

Retrieval

haystackconf.com/eu2024/talk-13/

· 2024 · Website

⋮

📄

📎

💬

Abstract ... practicalities of training embedding models for production (multi-modal) vector search. Topics will span data, training, and evaluation. More ... model training for business objectives by leveraging historic search-result interactions. - The importance of data quality, including ...

□

How we speed up filtered vector search with ACORN

☆

Retrieval

weaviate.io/blog/speed-up-filtered-vect...

· 2024 · Website

⋮

📄

📎

💬

📁

Read

Abstract ... Learn about the challenges of filtered vector search and how Weaviate tackles them with ACORN. ...



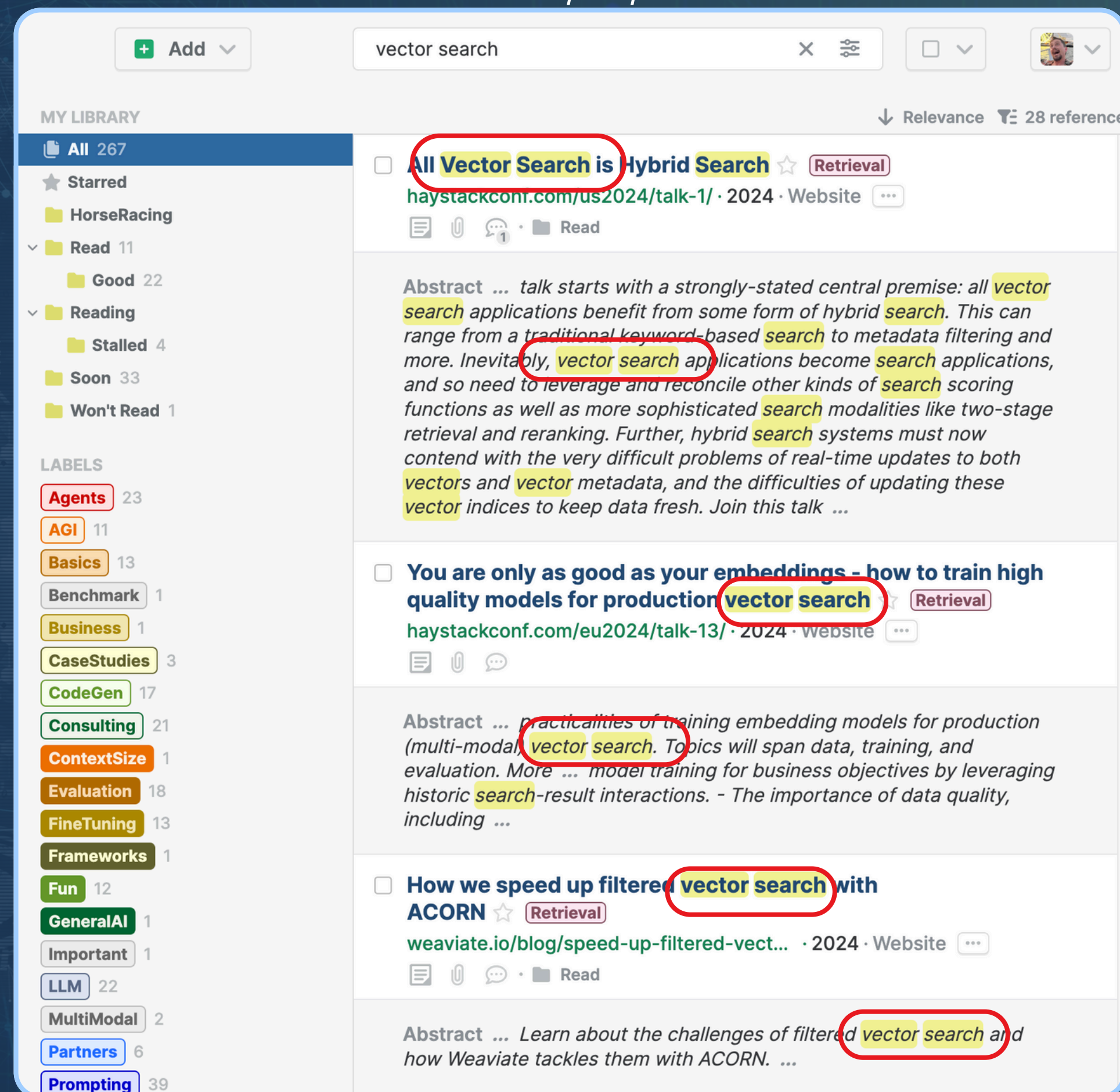
# Lexical Results

## Snippets

- Segments of text that contain the interesting phrases in context.
- It's like search-time chunking.

## Highlights

- Search text highlighted in results
- Mostly for human consumption.

The screenshot shows the Paperpile interface with a search bar at the top containing the text "vector search". Below the search bar, there's a "MY LIBRARY" section on the left with a sidebar containing various categories like "All 267", "Starred", "HorseRacing", "Read 11", "Good 22", "Reading", "Stalled 4", "Soon 33", and "Won't Read 1". Below these are "LABELS" such as "Agents 23", "AGI 11", "Basics 13", "Benchmark 1", "Business 1", "CaseStudies 3", "CodeGen 17", "Consulting 21", "ContextSize 1", "Evaluation 18", "FineTuning 13", "Frameworks 1", "Fun 12", "GeneralAI 1", "Important 1", "LLM 22", "MultiModal 2", "Partners 6", and "Prompting 39".

The main content area displays search results for "vector search". The first result is titled "All Vector Search is Hybrid Search" from haystackconf.com/us2024/talk-1/. The abstract snippet for this result is: "Abstract ... talk starts with a strongly-stated central premise: all vector search applications benefit from some form of hybrid search. This can range from a traditional keyword-based search to metadata filtering and more. Inevitably, vector search applications become search applications, and so need to leverage and reconcile other kinds of search scoring functions as well as more sophisticated search modalities like two-stage retrieval and reranking. Further, hybrid search systems must now contend with the very difficult problems of real-time updates to both vectors and vector metadata, and the difficulties of updating these vector indices to keep data fresh. Join this talk ...".

The second result is titled "You are only as good as your embeddings - how to train high quality models for production vector search" from haystackconf.com/eu2024/talk-13/. The abstract snippet for this result is: "Abstract ... practicalities of training embedding models for production (multi-modal) vector search. Topics will span data, training, and evaluation. More ... model training for business objectives by leveraging historic search-result interactions. - The importance of data quality, including ...".

The third result is titled "How we speed up filtered vector search with ACORN" from weaviate.io/blog/speed-up-filtered-vect... The abstract snippet for this result is: "Abstract ... Learn about the challenges of filtered vector search and how Weaviate tackles them with ACORN. ...".





# Lexical Results

## Differences

Semantic	Lexical
The chunks and their metadata	The full doc (all fields)
	Snippets
No Facets	Facets Aggregate Data

Maybe good for RAG

Watch us apply this to RAG!







# Supercharging RAG with Lexical Search







# Indexing Docs

## WANDS (Wayfair ANnotation Dataset)

- E-commerce product dataset
  - Home items
  - Furniture
  - Appliances
- Used to benchmark search relevance algorithms.
- 42,994 items.
- Fields:
  - product\_name
  - product\_class
  - product\_description
  - rating\_count

I added "availability"  
which lists the  
states where the  
product is available

## Elasticsearch Mapping

```
17 mapping = {
18     "mappings": {
19         "properties": {
20             "product_id": {"type": "keyword"},
21             "product_name": {
22                 "type": "text",
23                 "analyzer": "english",
24                 "fields": {
25                     "exact": {
26                         "type": "text",
27                         "analyzer": "standard"
28                     }
29                 }
30             },
31             "product_class": {"type": "keyword"},
32             "product_description": {
33                 "type": "text",
34                 "analyzer": "english",
35                 "fields": {
36                     "exact": {
37                         "type": "text",
38                         "analyzer": "standard"
39                     }
40                 }
41             },
42             "rating_count": {"type": "integer"},
43             "average_rating": {"type": "float"},
44             "availability": {"type": "keyword"},
45         }
46     }
47 }
```







Arcturus Labs



# Searching

## Search query

```
def high
  if availability:
    search_query["query"]["bool"]["filter"].append(
      {
        "term": {
          "availability": availability
        }
      }
    )
  if product_class:
    search_query["query"]["bool"]["filter"].append(
      {
        "term": {
          "product_class": product_class
        }
      }
    )
  if min_average_rating:
    search_query["query"]["bool"]["filter"].append(
      {
        "range": {
          "average_rating": {"gte": min_average_rating}
        }
      }
    )

  search_query["size"] = num_results
  results = es.search(index=index_name, body=search_query)
],
```

## Search results as text

for laptops or tablets . the space-saving design lets you put  
Average Rating: 5.0

---

Product ID: 37239

Product Name: anti-fatigue comfort floor mat kitchen mat

Product Class: ['Kitchen Mats']

Product Description: prop a foot up , take a wide stance , ma  
your feet ! this contoured , not flat anti fatigue mat provid  
standing desk mat is engineered from the ground up to be the  
anti fatigue mat ? the patented active standing mat has a con  
excellent cushioning comfort that you can feel when standing  
Average Rating: 4.0

---

Product ID: 9458

Product Name: standing desk converter 100 % natural bamboo ad



[www.arcturus-labs.com](http://www.arcturus-labs.com)







# Formatted Response

## Facets definition

```
"aggs": {  
  "product_class": {  
    "terms": {  
      "field": "product_class",  
      "size": 10  
    }  
  }  
}
```

## Facet counts as text

Facet Counts:

product\_class:

Desks: 553  
TV Stands & Entertainment Centers: 273  
Kitchen Mats: 128  
Plant & Telephone Tables: 120  
Office Chairs: 108  
Area Rugs: 90  
Bathroom Storage: 79  
Toilet Paper Holders: 59  
Patio Umbrella Stands & Bases: 54







# LLM Search Tool

## Definition

```
tools = [{  
  "type": "function",  
  "function": {  
    "name": "search_catalog",  
    "description": "Search for products in the catalog using various filters. Sometimes the results will be an imperfect match for the query. If you feel that the results can be improved, you should refine the query by adding a product_class filter or by modifying the query string to use different search terms.",  
    "parameters": {  
      "type": "object",  
      "properties": {  
        "query_string": {  
          "type": "string",  
          "description": "The search query to match against product names and descriptions"  
        },  
        "product_class": {  
          "type": "string",  
          "description": "Filter results by product class. It is important to use exact string matches from the product_class list, so only use this after making a preliminary query_string-only search and reviewing the product_class facet.",  
          "optional": True  
        },  
        "min_average_rating": {  
          "type": "number",  
          "description": "Filter results by minimum average rating - this should be a number between 0 and 5",  
          "optional": True  
        }  
      }  
    }  
  }  
}]
```

## Implementation

```
from functools import partial  
  
tool_lookup = {  
    "search_catalog": partial(  
        high_level_search,  
        availability=user.state,  
    )  
}
```







# Application Assembled

system = ""

You are a helpful assistant that can the user find products from the catalog.

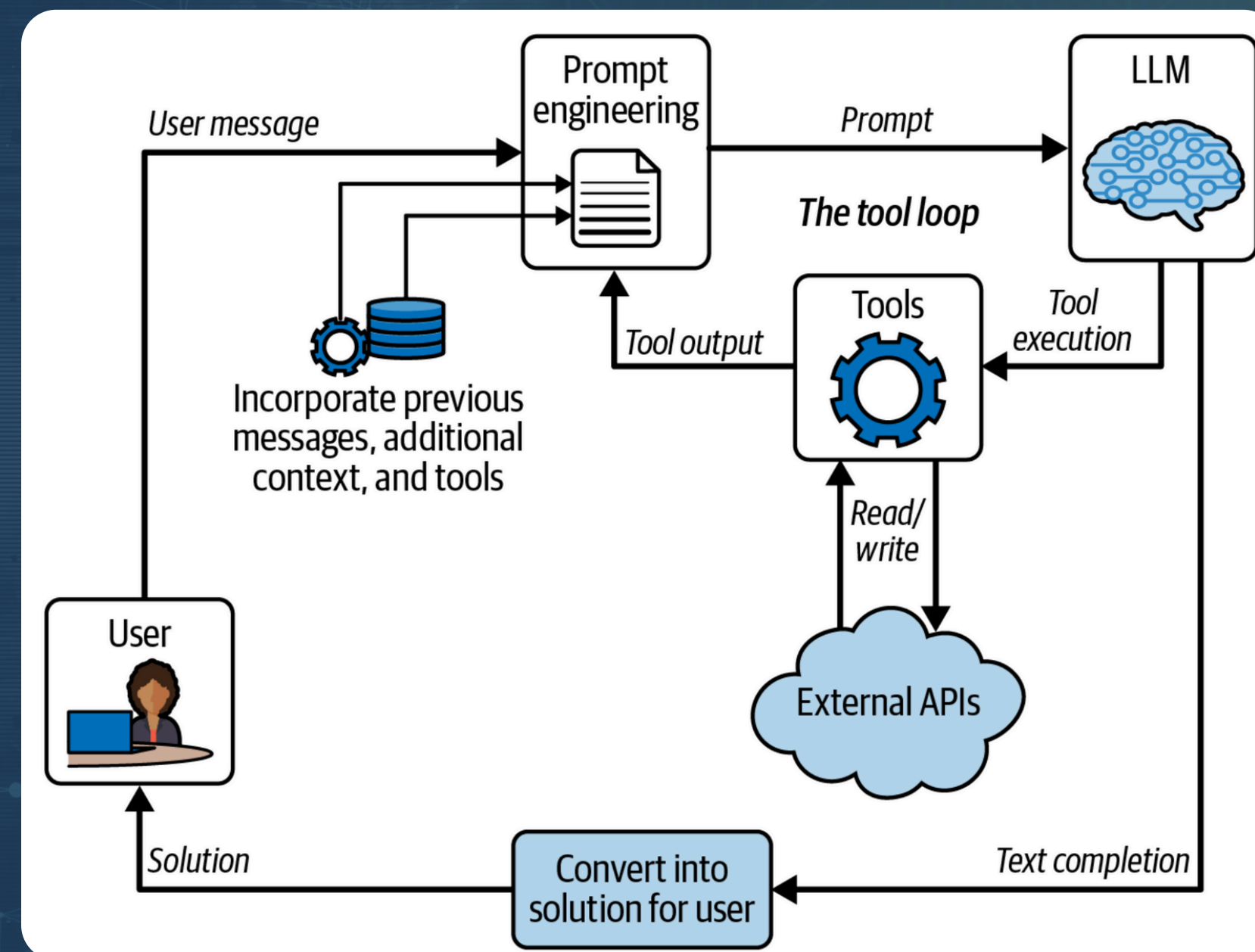
The user will discuss what they are looking for and it is your job to research the catalog and find the best matches. Research follows these steps:

1. Make a preliminary search based on whatever the user says they want.
2. Review the results in order to get a sense of what is available. Pay special attention to the product\_classes and counts that are available.
3. Prior to answering the user, make additional refined searches based on what you learned from the results of the preliminary search. If the results contain irrelevant items mixed in, then consider adding a product\_class filter to narrow the scope.
4. Finally, report back to the user about all that you've discovered.

When reporting the results follow these steps:

1. Start with a quick summary of the relevant results (across all searches) that addresses how they will help the user based upon the context of the conversation.
2. If it makes sense, describe the natural grouping of the results. Then you should present the top most relevant results sorted by relevance. Make sure to manually filter out results that you deem irrelevant.
3. At the end, make recommendations for further research that you can do to help the user find what they are looking for.

""





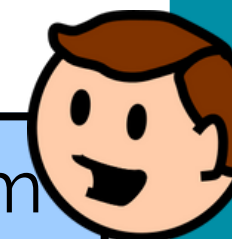
# Lexical RAG in Action

## Cool things:

- Search is automatically pre-filtered to items available in this state.



Hello! How can I help you?



My back hurts from sitting here in this chair all day.

...



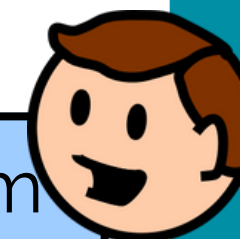
# Lexical RAG in Action

## Cool things:

- Search is automatically pre-filtered to items available in this state.



Hello! How can I help you?



My back hurts from sitting here in this chair all day.

```
search(  
  query="ergonomic chair"  
)  
search(  
  query="standing desk"  
)
```



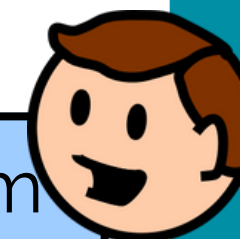
# Lexical RAG in Action

## Cool things:

- Search is automatically pre-filtered to items available in this state.



Hello! How can I help you?



My back hurts from sitting here in this chair all day.

```
search(  
  query="standing desk"  
)
```

```
search(  
  query="adjustable standing desk"  
)  
search(  
  query="standing desk converter"  
)
```



# Lexical RAG in Action

## Cool things:

- Search is automatically pre-filtered to items available in this state.
- The LLM can see the facets.

product\_class:  
Desks: 553  
TV Stands & Entertainment Center  
Kitchen Mats: 128  
Plant & Telephone Tables: 120  
Office Chairs: 108  
Area Rugs: 90  
Bathroom Storage: 79  
Toilet Paper Holders: 59  
Patio Umbrella Stands & Bases:



Hello! How  
can I help you?



My back hurts from  
sitting here in this  
chair all day.

```
search(  
  query="standing desk"  
)
```





# Lexical RAG in Action

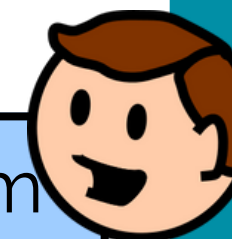
## Cool things:

- Search is automatically pre-filtered to items available in this state.
- The LLM can see the facets.
- The LLM can use what it learned to filter search.

product\_class:  
Desks: 553  
TV Stands & Entertainment Center  
Kitchen Mats: 128  
Plant & Telephone Tables: 120  
Office Chairs: 108  
Area Rugs: 90  
Bathroom Storage: 79  
Toilet Paper Holders: 59  
Patio Umbrella Stands & Bases:



Hello! How can I help you?



My back hurts from sitting here in this chair all day.

```
search(  
  query="standing desk"  
)
```

```
search(  
  query="standing desk",  
  prod_type="Desk",  
)
```



May I recommend the Stantastic adjustable standing desk?



# Lexical RAG in Action

## Cool things:

- Search is automatically pre-filtered to items available in this state.
- The LLM can see the facets.
- The LLM can use what it learned to filter search.
- The LLM can also add filters based on the user's interactions. Filters can be anything!

In principle you can even modify the relevance score.

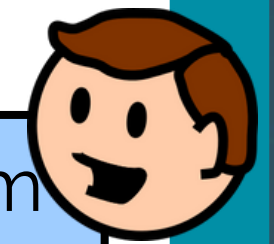


*I prefer lower-priced desks.*



can I help you?

My back hurts from sitting here in this chair all day.



```
search(  
  query="standing desk"  
)
```

```
search(  
  query="standing desk",  
  prod_type="Desk",  
)
```



May I recommend the Stantastic adjustable standing desk?

I only want the best rated desks.



```
search(  
  query="standing desk",  
  prod_type="Desk",  
  min_rating=4.5,  
)
```





# Limitations of Lexical Search

## and Hybrid Search to the Rescue?







# Limitations of Lexical Search

I know



You know



We know







# Limitations of Lexical Search

This is what Lexical Search thinks



Lexical Search

- Uses Bag of Words + Phrase Matching
- Doesn't get synonyms
- Doesn't get homonyms
- Is clueless about negation
- Misses context clues that modify intent

... the very things that  
Semantic Search  
excels at.



[www.arcturus-labs.com](http://www.arcturus-labs.com)







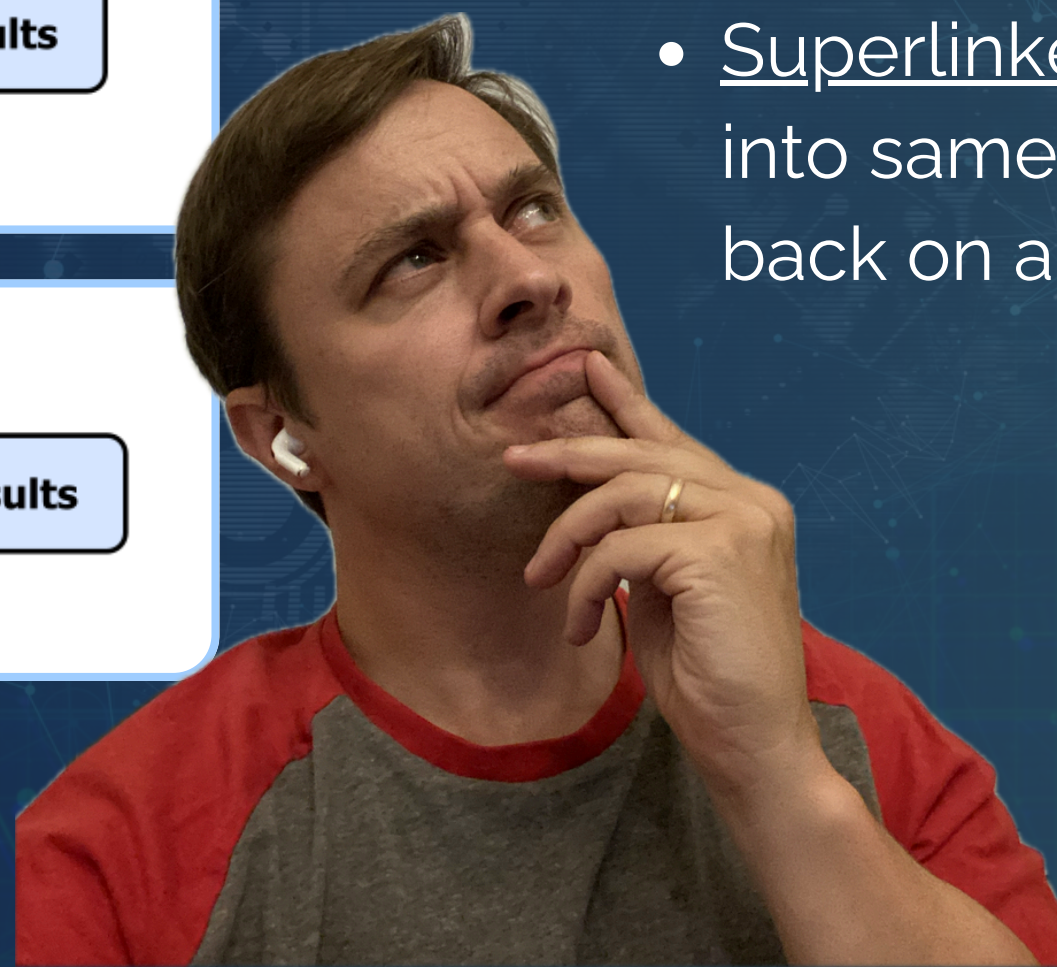
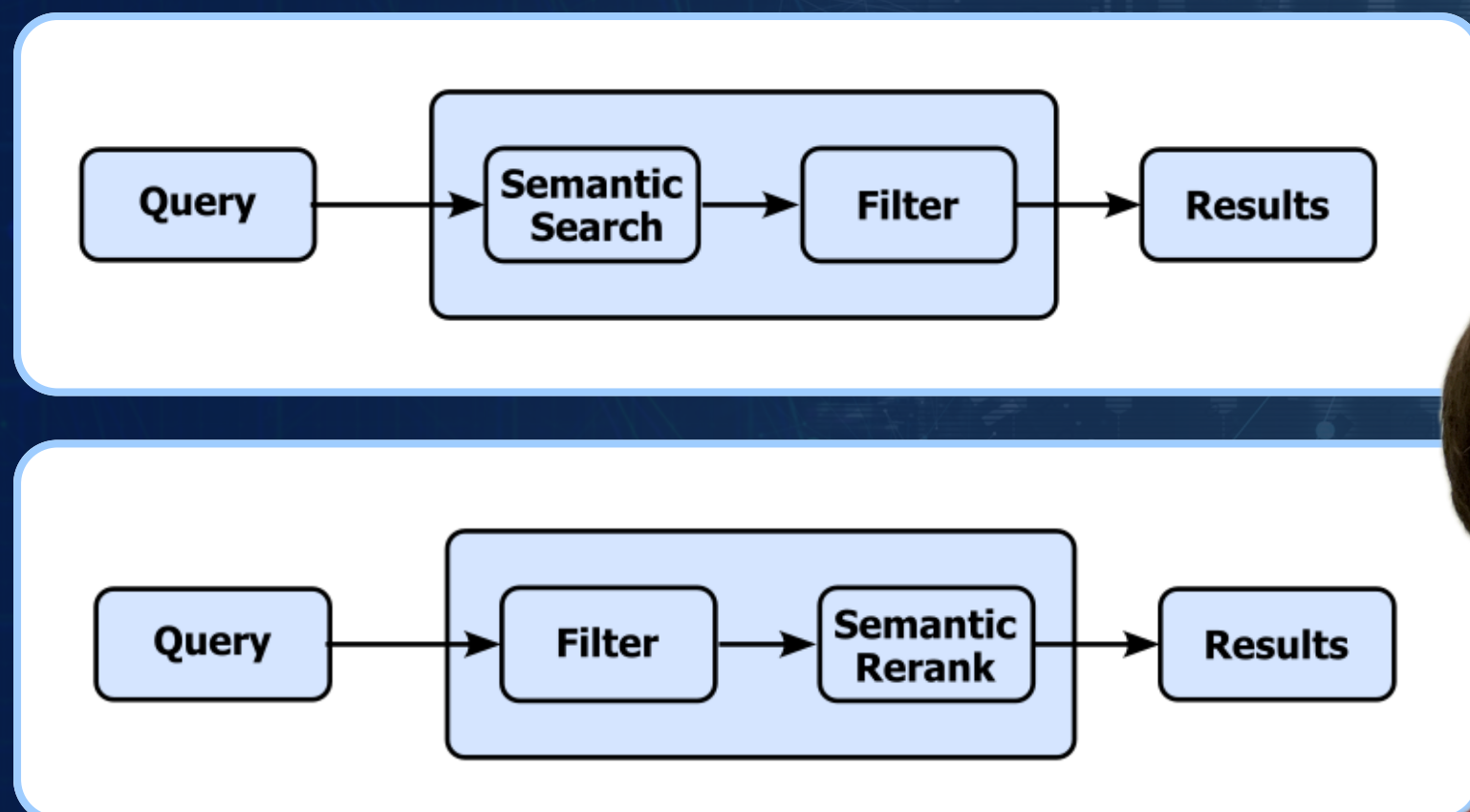
# Hybrid Search?

I want filters, facets and boosting from Lexical Search.  
I want search-by-meaning from Semantic Search.

## Are we really stuck here?

## Maybe. But we're working on it.

- Lexical search + reranking
- SPLADE - use embedding model to generate synthetic synonyms - shove that into lexical search.
- ACORN (now in Weaviate) - filters items *while* traversing HNSW datastructure
- Superlinked - embeds multiple datatypes into same vector(?) and can then piggy-back on any vector store







Arcturus Labs



# Thank you!



[www.arcturus-labs.com](http://www.arcturus-labs.com)



[www.arcturus-labs.com](http://www.arcturus-labs.com)

