Advanced RAG Optimization

Build & Ship Production-ready RAG

KRUX AI

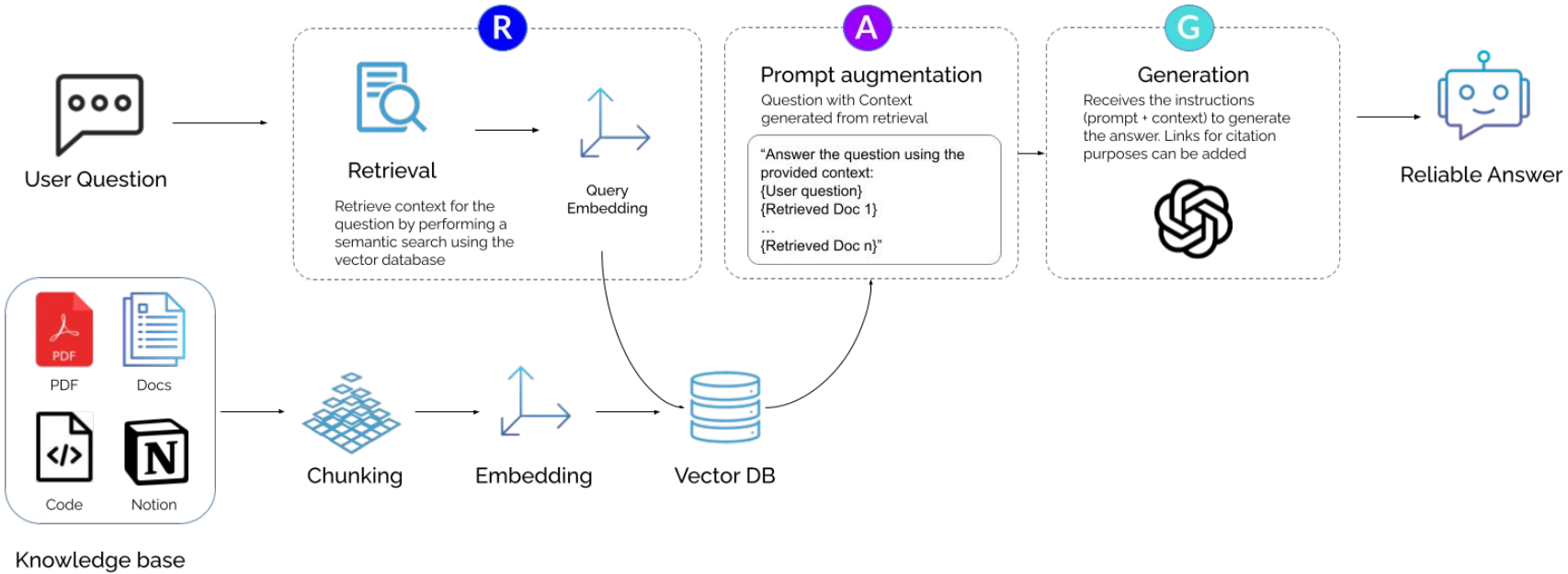
Aravind Parameswaran

Co-Founder

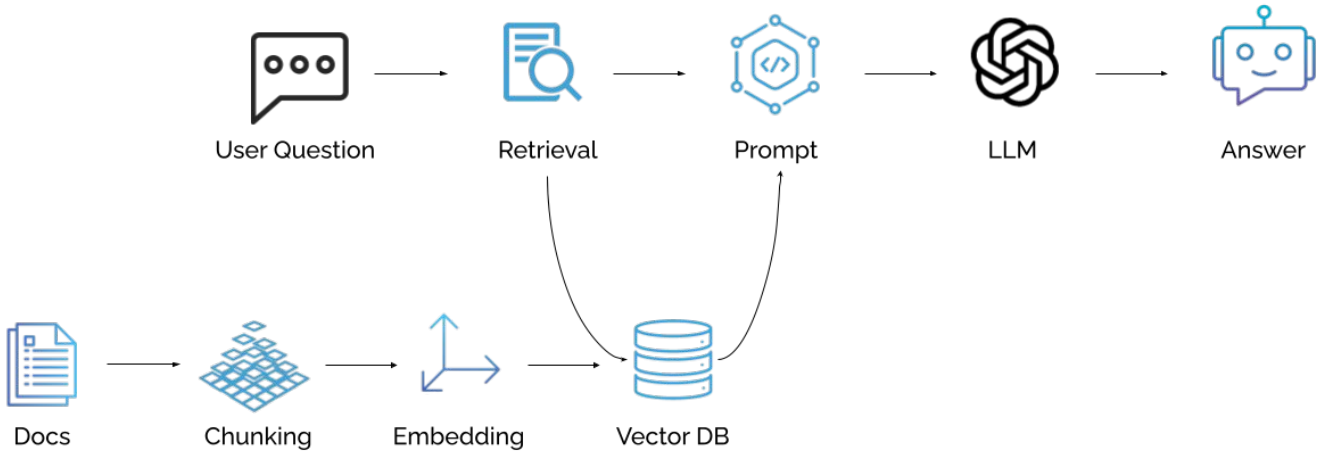
KRUX AI

aravind@krux.ai

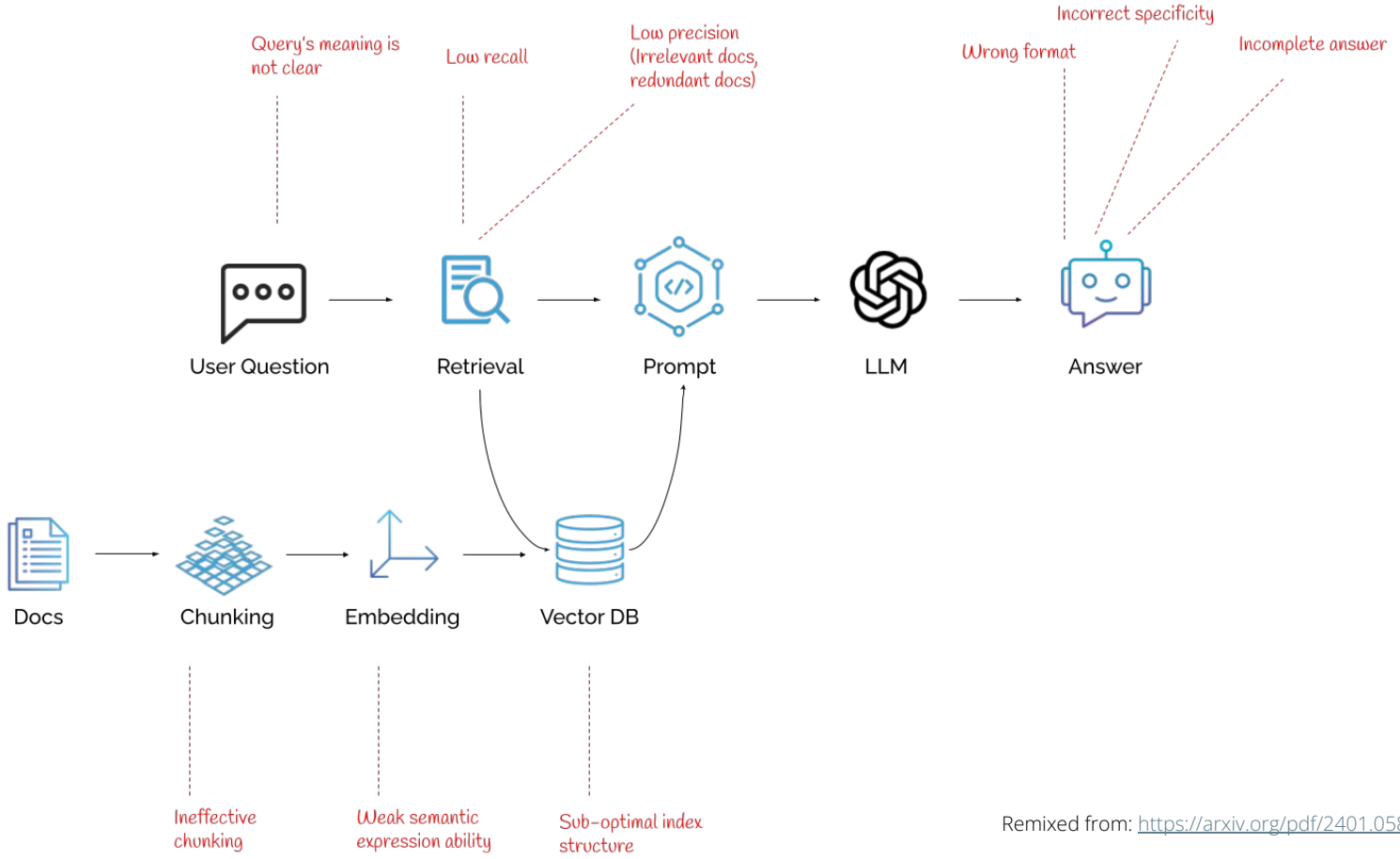
RETRIEVAL AUGMENTED GENERATION



NAIVE RAG


User Question
Retrieval
Prompt
LLM
Answer
Docs
Chunking
Embedding
Vector DB

FAILURE POINTS IN RAG



Remixed from: https://arxiv.org/pdf/2401.05856

RAG OPTIMIZATION TECHNIQUES

Pre-Retrieval

Improving the quality and retrievability of the information

TECHNIQUES

- Information density
- Optimize chunking
- Hierarchical index
- Retrieval symmetry
- Deduplication
- Context enrichment
- Query Transformation

Retrieval

Improving search performance & retrieval results

TECHNIQUES

- Query Routing
- Metadata Filtering
- Recursive Retrieval
- Ensemble / Fusion retrieval
- Late interaction models
- Combine graph + vector search

Post-Retrieval

Improving the relevance & information density of the retrieved results

TECHNIQUES

- Re-ranking
- Contextual compression
- Corrective RAG

Generation

Improving the the LLM call(s) that generate the final user response

TECHNIQUES

- Optimize Top “k” chunks to user in prompt
- Prompt optimization (CoT, CoN, etc.)
- Self-RAG
- Fine-tuning

PRE-RETRIEVAL OPTIMIZATION

Improving Information Density

Issues to address:

- Low information density
- Irrelevant information and/or noise
- Information duplication

Low Information density

More chunks in LLM context

Higher probability of incorrect response

Higher token usage & cost

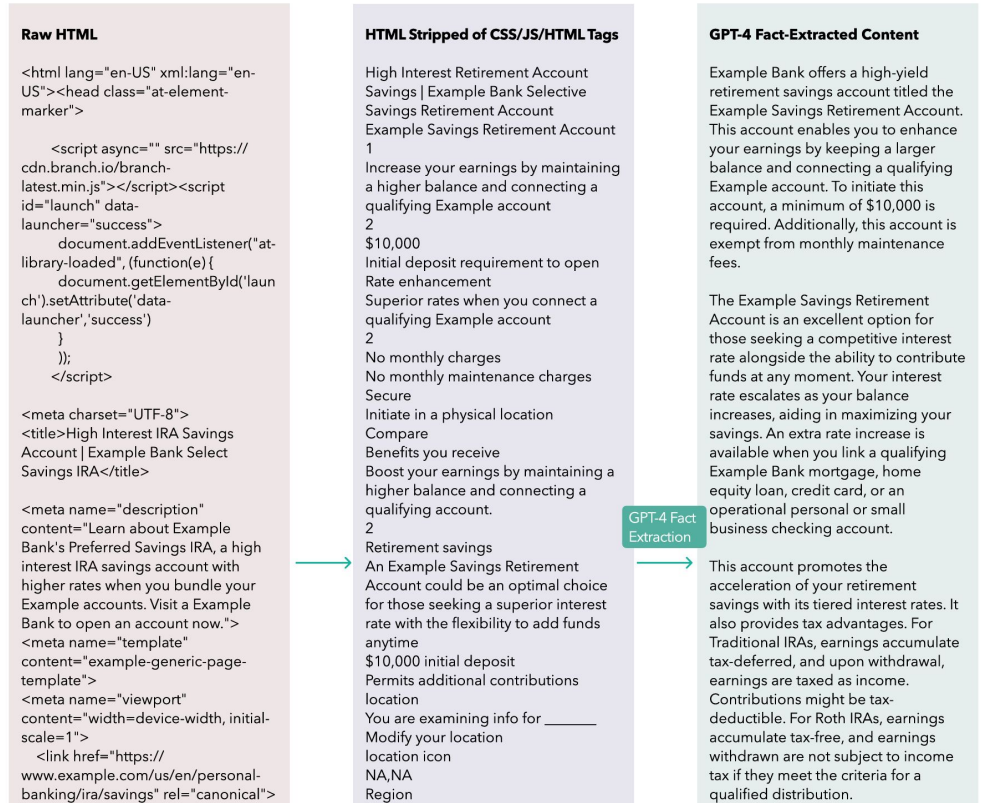
PRE-RETRIEVAL OPTIMIZATION

Improving Information Density

- Raw HTML: **~55,000** tokens
- Stripped HTML: **1,500** tokens
- GPT-4 processed HTML: **330** tokens

~4x

reduction in token costs



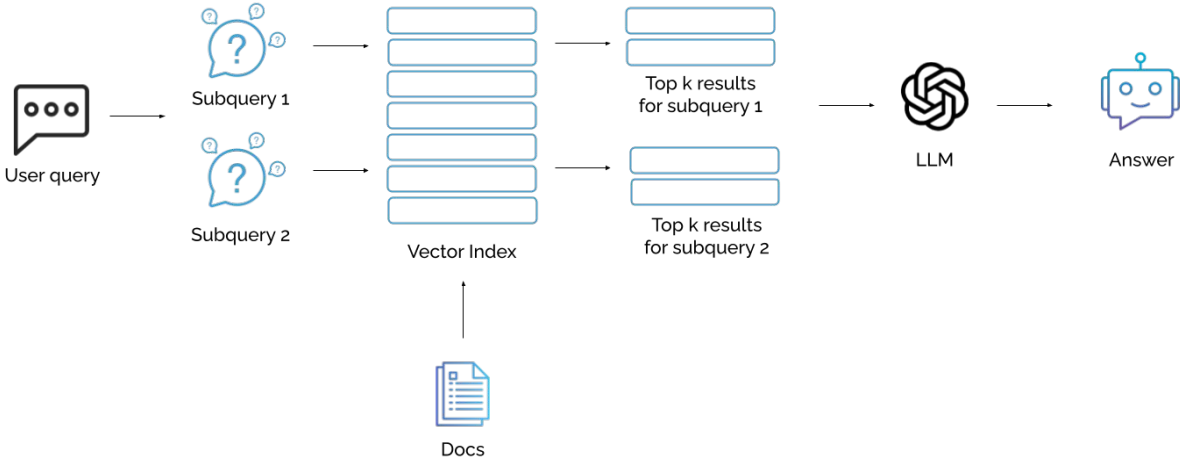
⚠️ Caveat: Risk of Information Loss

PRE-RETRIEVAL OPTIMIZATION

Query Transformation - Query Expansion, Query Reformulation

Issues to address:

- Complex queries
- Ambiguous or poorly worded queries



PRE-RETRIEVAL OPTIMIZATION

Query Transformation - Optimize search query with conversation history as input

```
Customer: “What are the interest rates for your CDs?”
Assistant: “Our interest rates are XYZ.”
Customer: “Which credit card is good for travel?”
Assistant: “The XYZ credit card is good for travel for ABC reasons”
Customer: “Tell me more about the interest rate for that”
```

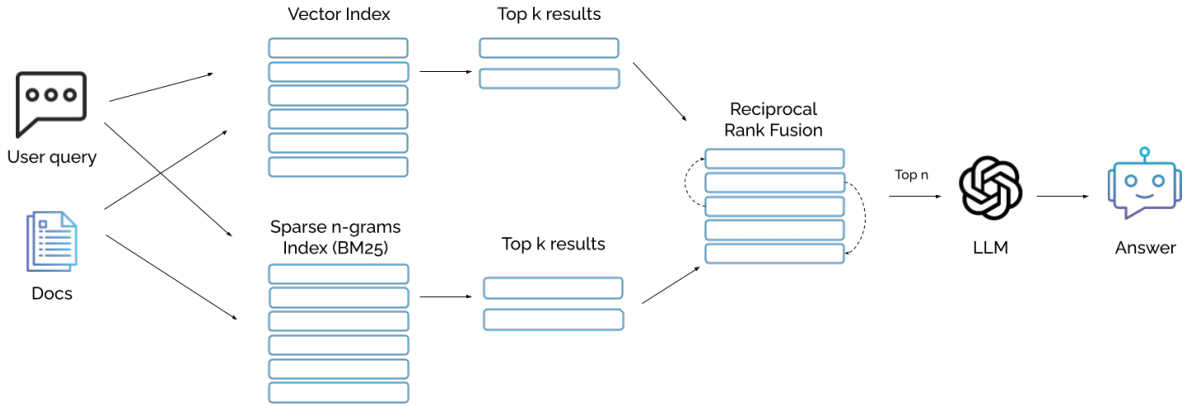
```
Prompt = ‘You are examining a conversation between a customer of Example bank
and an Example bank chatbot. A documentation lookup of Example bank’s
policies, products, or services is necessary for the chatbot to respond to the
customer. Please construct a search query that will be used to retrieve the
relevant documentation that can be used to respond to the user.’
```

RETRIEVAL OPTIMIZATION

Ensemble / Fusion Retrieval

- Combine multiple retrievers
- For eg: keyword-based search (BM25) + vector search
- Use Reciprocal Rank Fusion to merge results

$$\text{RRF Score} = \sum_{i=1}^{N} \left(\frac{\text{weight}_i}{\text{rank}_i + c} \right)$$

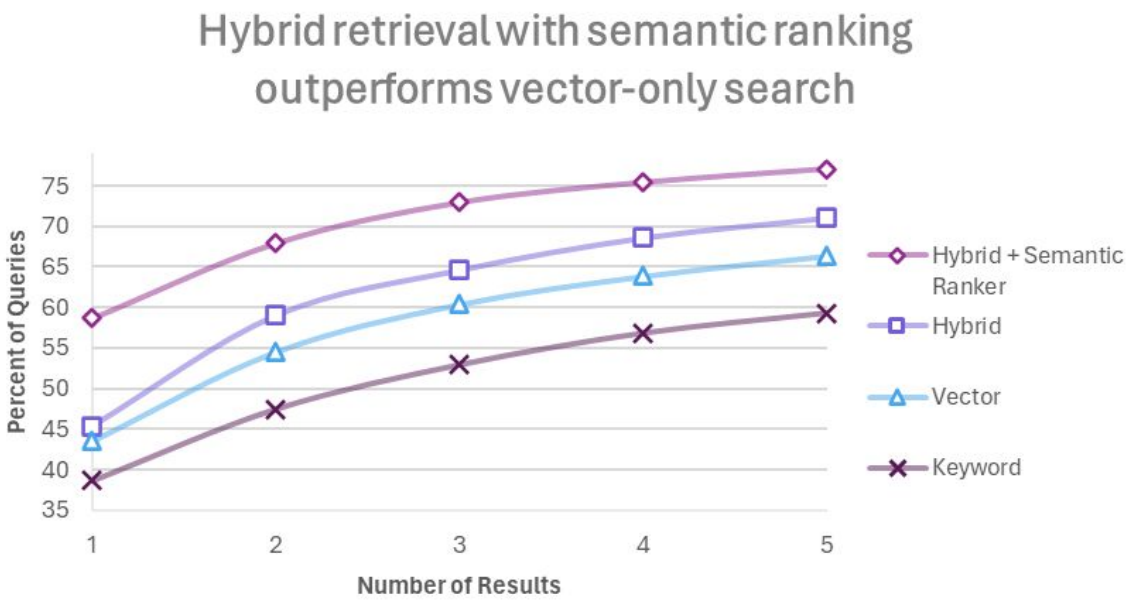


RETRIEVAL OPTIMIZATION

Ensemble / Fusion Retrieval

- Combine multiple retrievers
- For eg: keyword-based search (BM25) + vector search
- Use Reciprocal Rank Fusion to merge results

$$\text{RRF Score} = \sum_{i=1}^{N} \left(\frac{\text{weight}_i}{\text{rank}_i + c} \right)$$

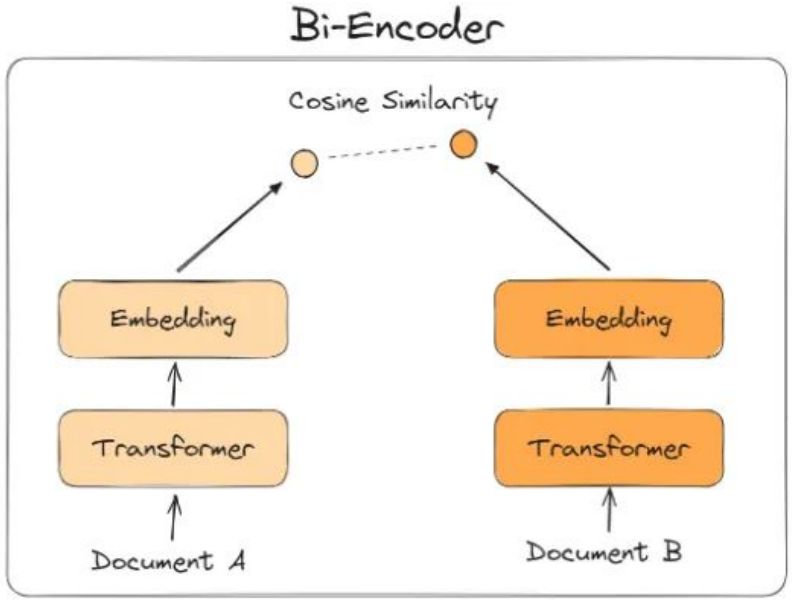


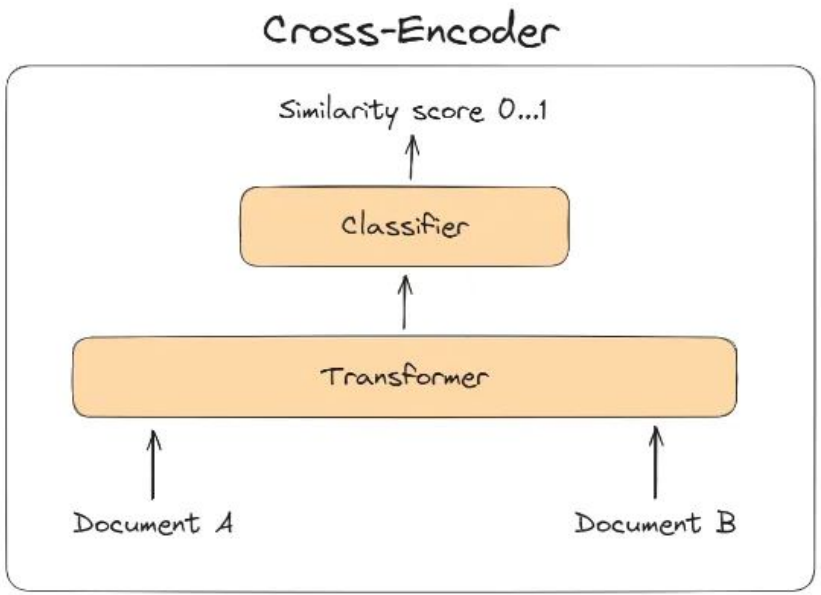
Percentage of queries where high-quality chunks are found in the top 1 to 5 results

Source: https://techcommunity.microsoft.com/t5/ai-azure-ai-services-blog/azure-ai-search-outperforming-vector-search-with-hybrid/ba-p/3929167

POST-RETRIEVAL OPTIMIZATION

Cross-encoder Re-ranking





POST-RETRIEVAL OPTIMIZATION

Cross-encoder Re-ranking

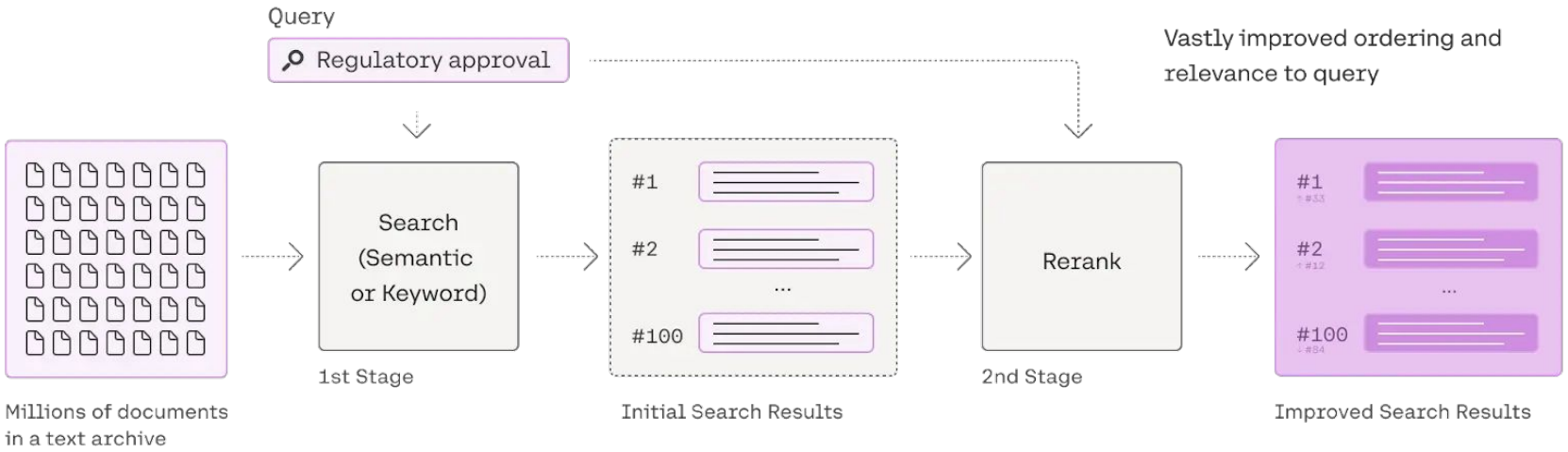
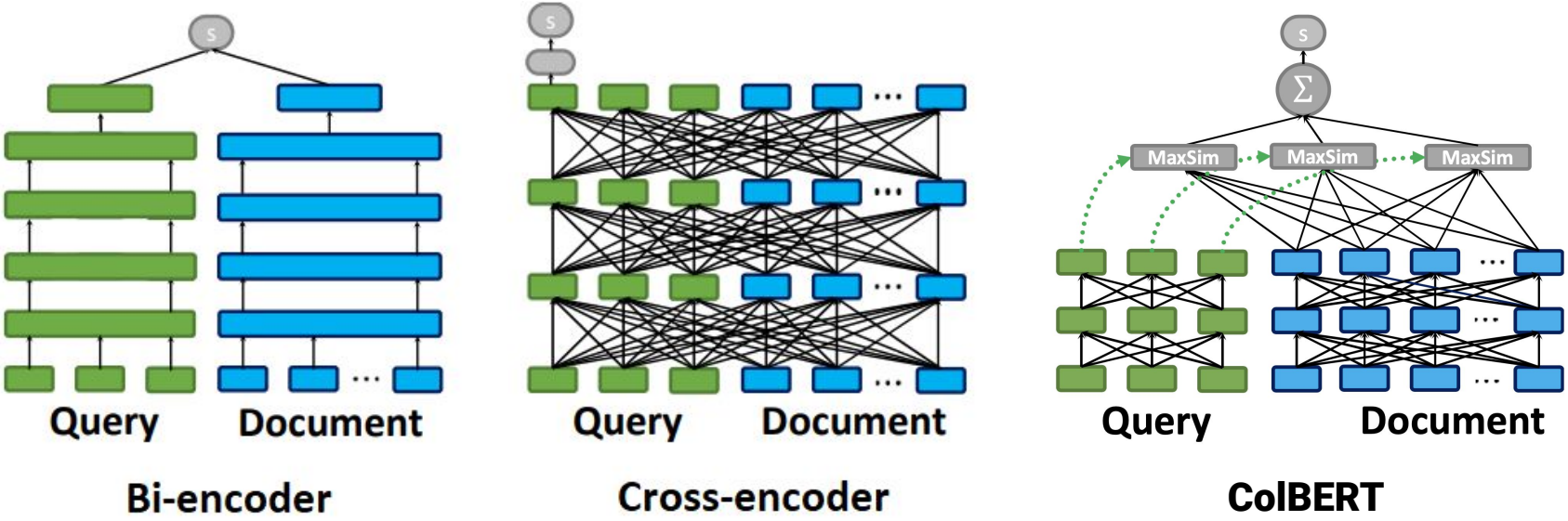


Image credits: https://txt.cohere.com/rerank/

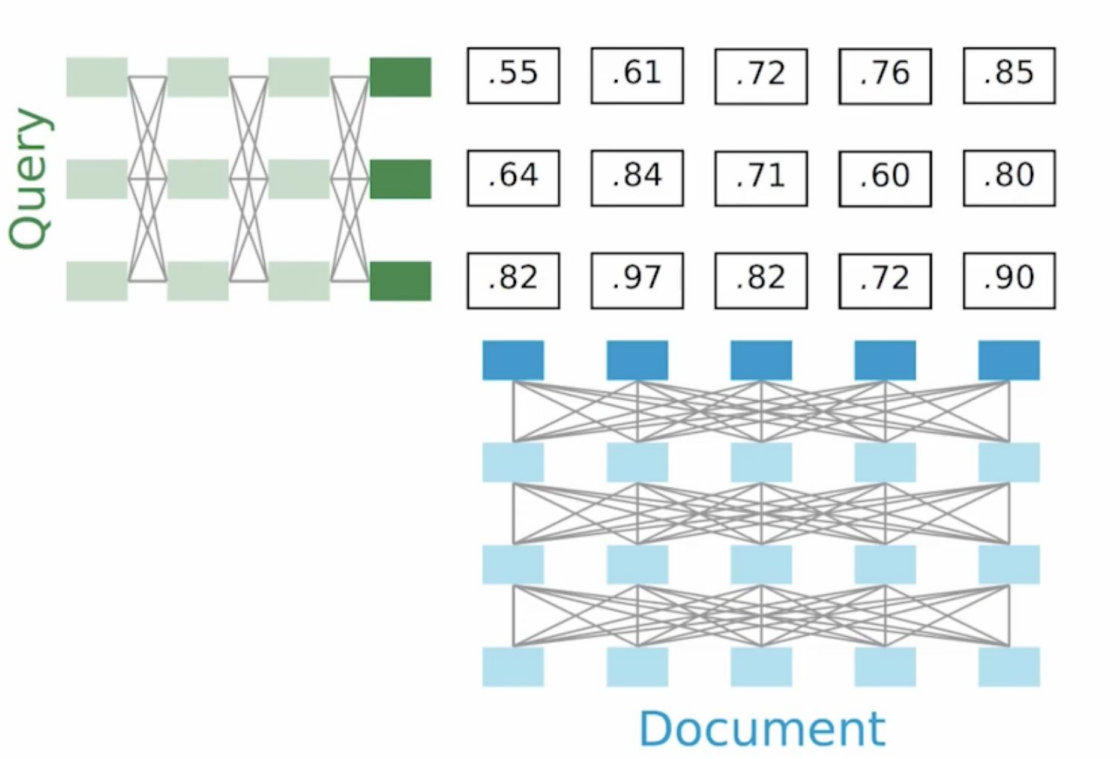
POST-RETRIEVAL OPTIMIZATION

ColBERT model



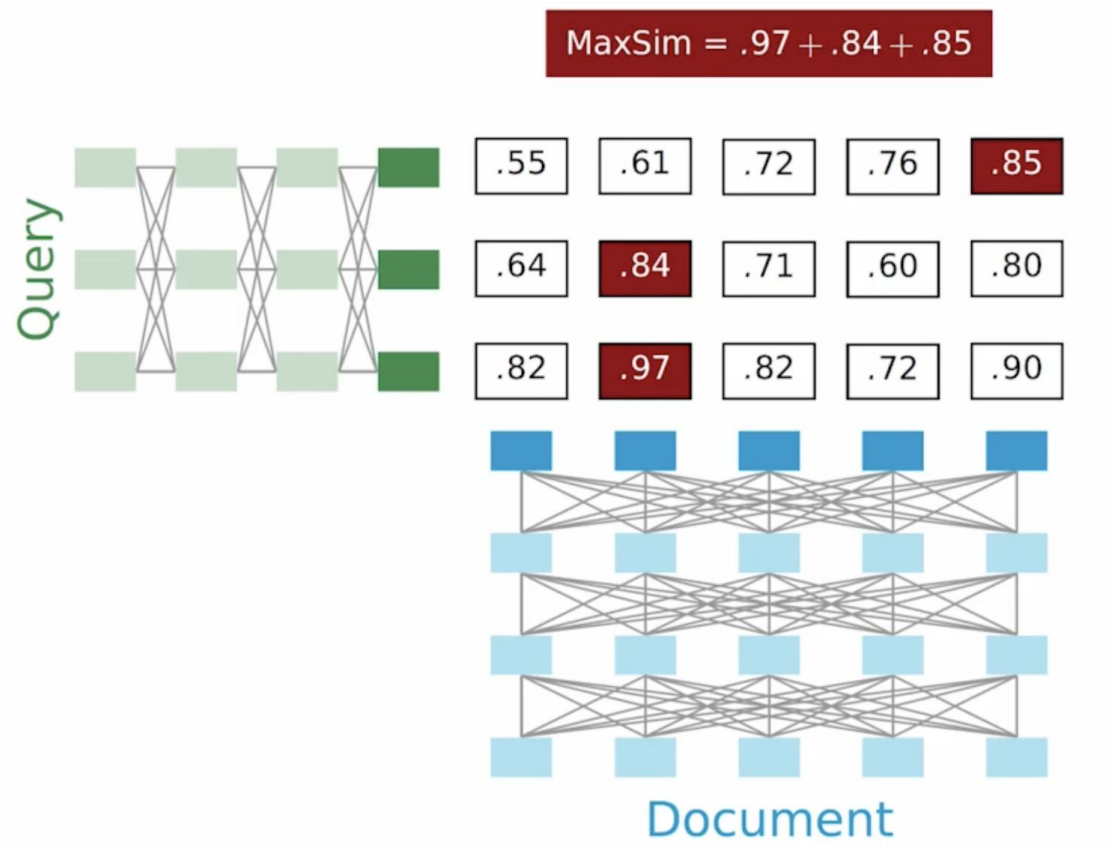
POST-RETRIEVAL OPTIMIZATION

ColBERT model



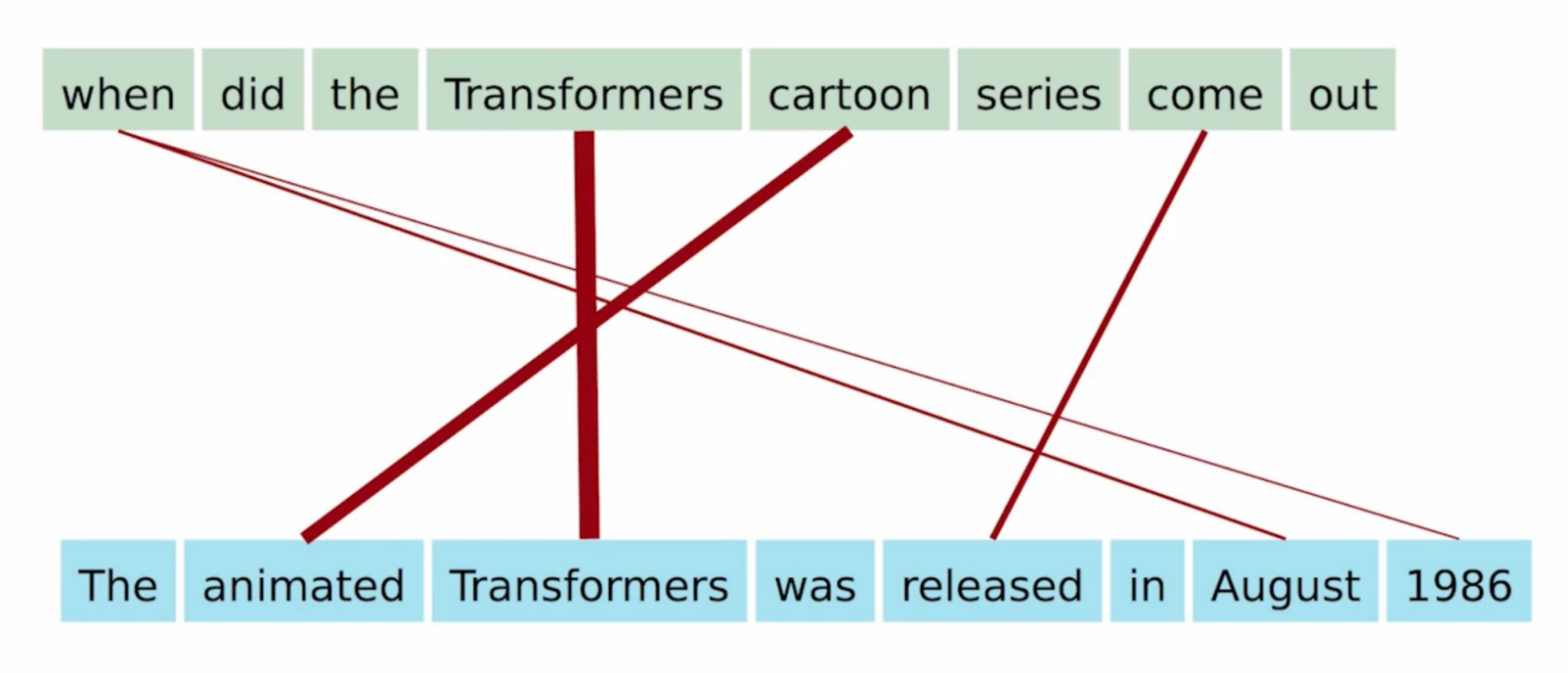
POST-RETRIEVAL OPTIMIZATION

ColBERT model



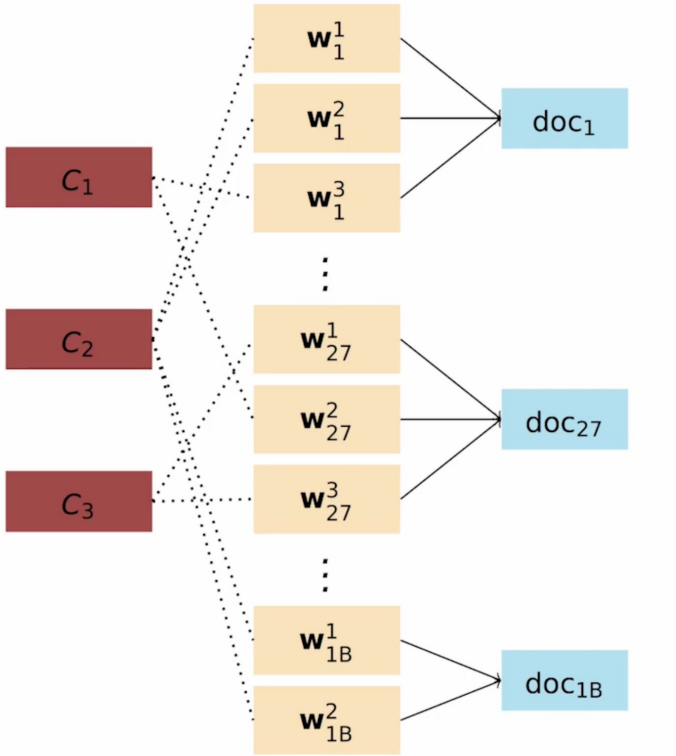
POST-RETRIEVAL OPTIMIZATION

ColBERT model



POST-RETRIEVAL OPTIMIZATION

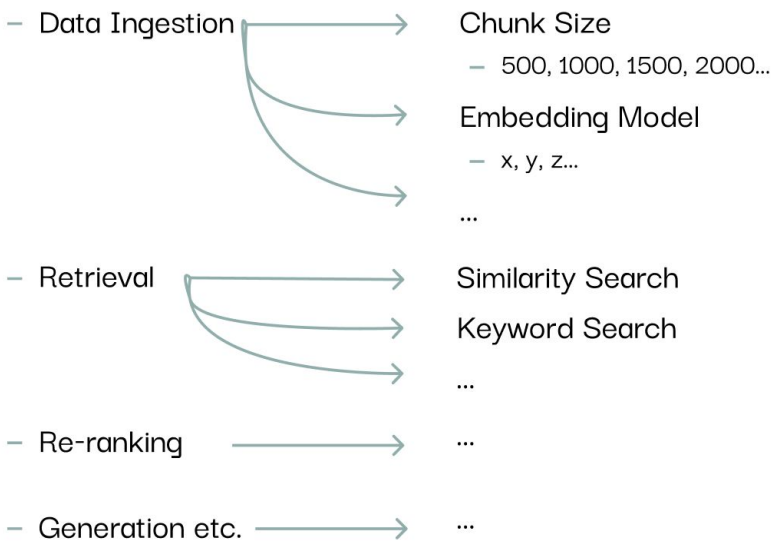
ColBERTv2 - Centroid based re-ranking



OVERALL...

Creating an Optimized RAG is really hard...

- Lots of moving parts with multiple parameters
- So many techniques & strategies
- Takes several months of trial & error



- 5 chunking methods,
- 5 chunk sizes,
- 5 embedding models,
- 5 retrievers,
- 5 re-rankers/compressors
- 5 prompts
- 5 LLMs

=

How many RAG configurations?

- 5 chunking methods,
- 5 chunk sizes,
- 5 embedding models,
- 5 retrievers,
- 5 re-rankers/compressors
- 5 prompts
- 5 LLMs

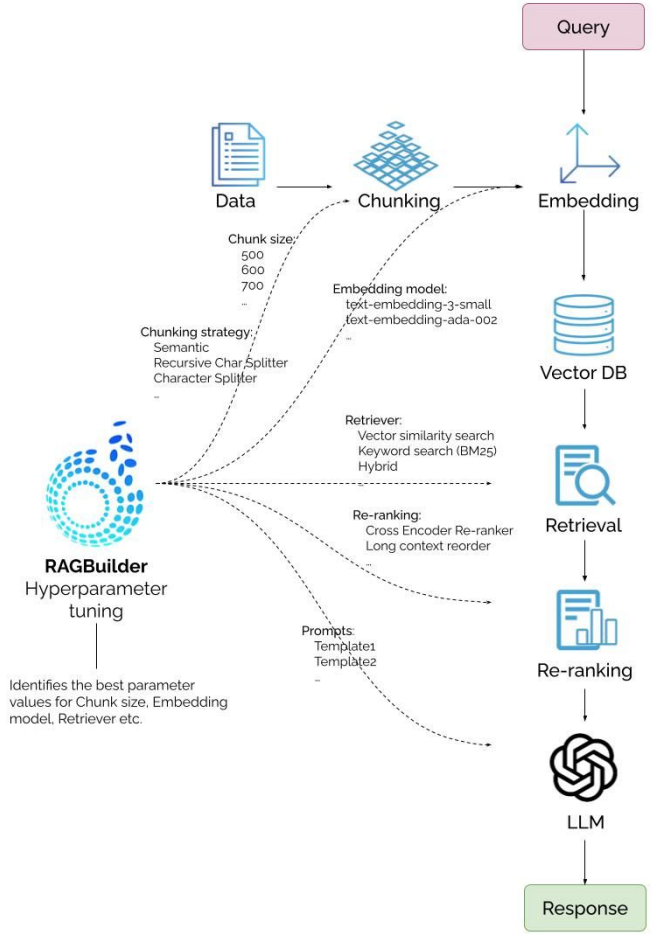
=

# of RAG Configurations

5 mins per config => **271 days**

RAGBuilder

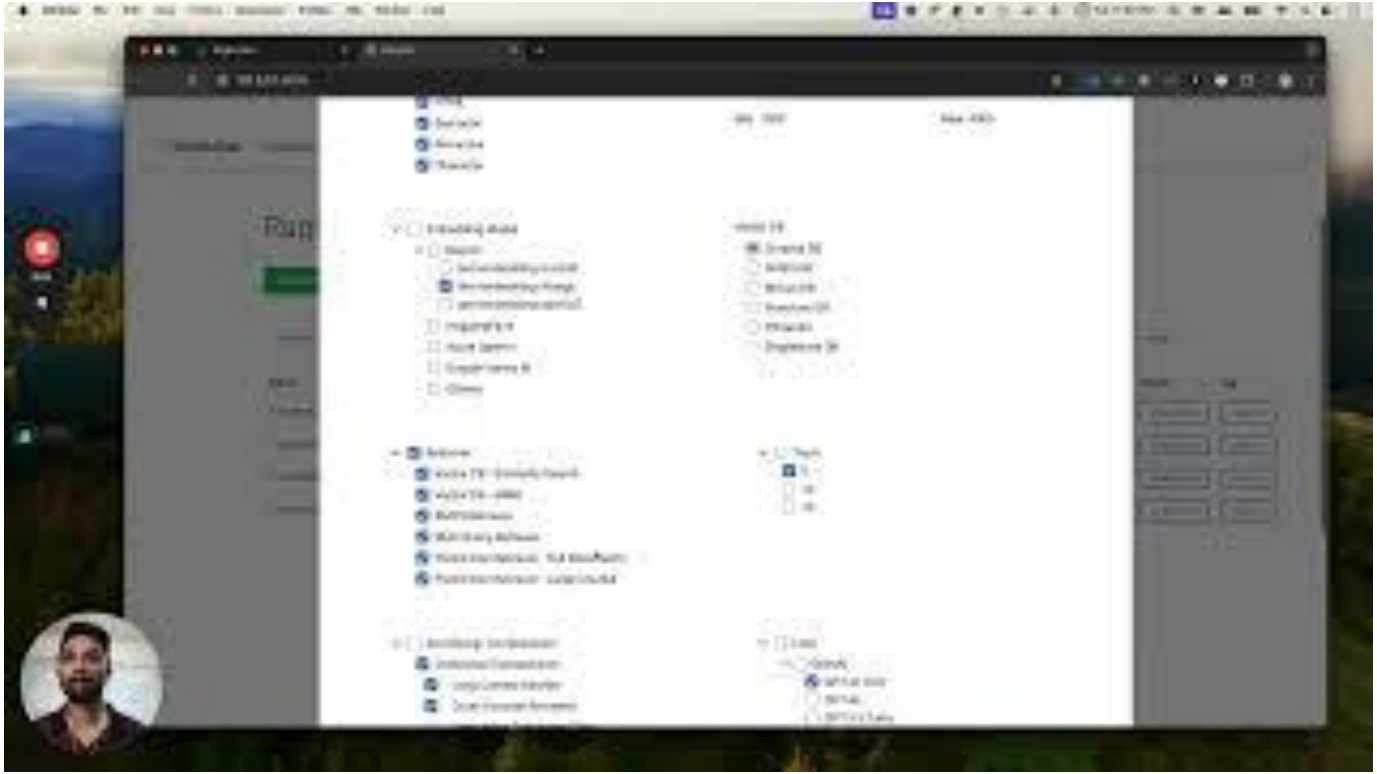
- ✓ Hyperparameter tuning
- ✓ SOTA predefined RAG templates
- ✓ Synthetic test data generation with auto re-use
- ✓ Build with the best models of your choice (open/closed)
- ✓ Knowledge base/ Graph RAG
- ✓ Bring your own RAG or component [coming soon]
- ✓ Cloud hosted solution [coming soon]



11 GITHUB TRENDING
#11 Repository Of The Day

Open source repo: https://github.com/KruxAI/ragbuilder

Demo time



https://youtu.be/-vfBSNZuAPE

Can you take this 1 min survey?