

Beyond Naive RAG: Practical Advanced Methods

Hamel Husain Benjamin Clavié
Nandan Thakur Orion Weller Antoine Chaffin
Bryan Bischof Ayush Chaurasia Kelly Hong

2025-10-17

Table of contents

Preface	1
About This Book	1
1 I Don't Use RAG, I Just Retrieve Documents	3
1.1 The Title	4
1.2 The "RAG is Dead" Controversy	6
1.3 What is RAG Really?	8
1.4 The Standard RAG Flow	9
1.5 The Problem with "2023 RAG"	11
1.6 Single-Vector Search Limitations	12
1.7 Why Long Context Doesn't Replace RAG	13
1.8 Retrieval is Essential	14
1.9 Key Takeaways	15
1.10 Better RAG is the Solution	17
1.11 The Retrieval Landscape	18
1.12 Expert Speakers Preview	20
1.13 Video	24
2 Modern IR Evaluation for RAG	25
2.1 Introduction and Speaker Background	26
2.2 The History of Information Retrieval	29
2.3 The Cranfield Paradigm	32
2.4 The BEIR Benchmark	34
2.5 Problems with Current Benchmarks	36
2.6 The RAG Era Changes Everything	40
2.7 Different Users, Different Goals	42

2.8	The Evaluation Mismatch	43
2.9	Introducing FreshStack	45
2.10	FreshStack Data Sources	47
2.11	The FreshStack Pipeline	49
2.12	FreshStack Evaluation Metrics	52
2.13	Key Results and Findings	53
2.14	Chapter Reflections	56
2.15	Video	57
3	Optimizing Retrieval with Reasoning Models	58
3.1	LLM Capabilities: Instruction Following and Reasoning . .	59
3.2	The Search Paradigm Hasn't Changed	62
3.3	Evolution of Search Paradigms	65
3.4	Understanding Instructions in IR	69
3.5	Introducing Promptriever and Rank1	70
3.6	Promptriever: Instruction-Trained Retrieval	71
3.7	Promptriever Evaluation Results	74
3.8	Rank1: Reasoning-Based Reranking	77
3.9	Rank1 Performance Results	80
3.10	Finding Novel Relevant Documents	82
3.11	Chapter Takeaways	84
3.12	Video	85
4	Late Interaction Models For RAG	86
4.1	Dense Vector Search Architecture	87
4.2	Why Dense Models Became Popular	88
4.3	The Benchmark Problem	89
4.4	Hidden Limitations	90
4.5	Long Context Performance	91
4.6	Complex Retrieval Tasks	92
4.7	BM25's Surprising Strength	94
4.8	The Pooling Problem	95
4.9	Late Interaction Solution	98
4.10	Performance Advantages	100

4.11	Interpretability Benefits	104
4.12	Barriers to Adoption	105
4.13	PyLate: Making Late Interaction Accessible	106
4.14	Future Research Directions	109
4.15	Chapter Summary	112
4.16	Video	113
5	RAG with Multiple Representations	114
5.1	The Map is Not the Territory	115
5.2	Deconstructing RAG Buzzwords	115
5.3	A First-Principles View of RAG	120
5.4	The Three Responsibilities of an IR Engineer	123
5.5	Practical Application: Curving Space	126
5.6	Agents as Routers	130
5.7	Dynamic Representations	132
5.8	Demo: Semantic Dot Art	133
5.9	System Architecture	135
5.10	Integration with Other Techniques	136
5.11	Chapter Principles	137
5.12	Video	137
6	Context Rot: When Long Context Windows Fail	138
6.1	Introduction	139
6.2	The Long Context Promise	140
6.3	Limitations of Needle in a Haystack	142
6.4	Experiment 1: Semantic vs. Lexical Matching	143
6.5	Experiment 2: The Impact of Distractors	145
6.6	Experiment 3: Context Structure Matters	150
6.7	Experiment 4: Conversational Memory	152
6.8	Experiment 5: Simple Tasks Aren't Immune	153
6.9	Key Takeaways	154
6.10	Practical Solutions: Context Engineering	155
6.11	Further Resources	156
6.12	Q&A Session	156

Conclusion	159
Key Takeaways	159
Looking Forward	160
Resources	160

Preface

This book is a compilation of advanced talks on Retrieval Augmented Generation (RAG) that were part of our [AI Evals course](#). The series features leading researchers and practitioners who share cutting-edge techniques for building production-ready RAG systems.

About This Book

This handbook condenses **over 6 hours of expert instruction** into annotated presentations that allow you to quickly learn advanced RAG techniques in under an hour. Each chapter is based on detailed presentations from world-class researchers who are pushing the boundaries of what's possible with retrieval systems.

The field of RAG is rapidly evolving beyond the simple “embed documents and search” paradigm that dominated 2023. This book presents six key areas where the field is advancing:

1. **Why RAG Isn't Dead** - Understanding the evolution beyond naive single-vector search
2. **Modern Evaluation** - New benchmarks and metrics for the RAG era
3. **Reasoning-Enhanced Retrieval** - Incorporating instruction-following and reasoning into search
4. **Late Interaction Models** - Moving beyond single vectors to token-level representations

- 5. **Multiple Representations** - Building flexible systems with diverse data maps
- 6. **Context Rot** - Understanding how LLM performance degrades with longer contexts

Each chapter combines theory with implementation guidance, code examples, and real-world applications. The presentations are annotated with timestamped references to the original videos, allowing you to dive deeper into specific topics that interest you most.

You can also view the web version of this series at hamel.dev/notes/llm/rag/not_dead.html.

*If you want to learn more about AI Evals, check out our **AI Evals course**.* Here is a [35% discount code](#) for readers.

1 I Don't Use RAG, I Just Retrieve Documents

Based on a presentation by Benjamin Clavié

As part of our [LLM Evals course](#), I hosted [Benjamin Clavié](#) to kick off a 6-part mini-series on evaluating and optimizing RAG. Ben is a retrieval researcher who has built widely used tools like [RAGatouille](#) and [rerankers](#) among other things. His talk focused on important developments in RAG and where you should be paying attention (late-interaction, reasoning, evals, multimodal, context engineering, etc.).

1.1 The Title

I don't use RAG, I just retrieve documents

Or How I Learned to Stop Worrying and Love Retrieving

Benjamin Clavié, 24-25 June 2025

(*Timestamp: 00:00:00*)

The cheeky title of the talk.

About me

- I'm based in Japan in Musashino City, a lovely part of Tokyo that's home to the Ghibli Museum.
- I do ML R&D at Answer.AI (and maybe somewhere else soon...), and I particularly enjoy NLP and 🌟 Retrieval 🌟
- If you know me, it's likely because of X, where I'm this guy:



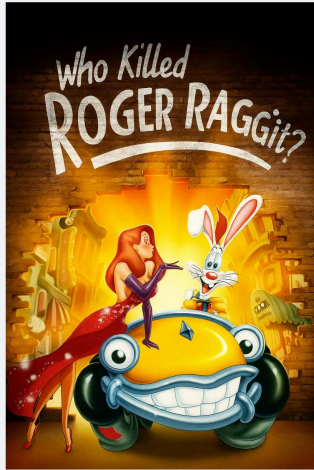
I call this **The Monopicture**. A singular picture of me taken 5 years ago which has gotten a life of its own as the only picture of me floating around.

- I've recently worked on ModernBERT, as well as some niceties related to ColBERT (more on that later!)

(*Timestamp: 00:01:08*)

Ben introduces himself, noting his base in Musashino City, Japan (home of the Ghibli Museum). He currently does ML R&D at Answer.AI. He jokes about his widely-circulated profile picture, which he dubs “The Monopicture,” a single photo from five years ago that has taken on a life of its own. He also mentions his recent work on ModernBERT and niceties related to ColBERT, which he promises to discuss later.

1.2 The “RAG is Dead” Controversy



**RAG
Is
DEAD**

(*Timestamp: 00:02:05*)

Ben discusses the controversial idea that “RAG is dead.” Ben explains that the statement only applies to a very narrow, and often misunderstood, definition of RAG that was popularized by marketing efforts.

RAG Is DEAD But Retrieval Is NOT

(Timestamp: 00:02:33)

The “RAG” that many people came to know in 2023: simplistic, single-vector semantic search approach may be obsolete. However, the underlying concept of **retrieval** is still relevant.

1.3 What is RAG Really?

RAG is... Wait, what's RAG?

Or what's in a name?

- RAG stands for **R**etrieval **A**ugmented **G**eneration
- In effect, it's a simple pipeline:
 - Somehow **R**etrieve documents before
 - Passing them to a **G**enerative model, so its output is **A**ugmented by the content.
- RAG, as a concept, is effectively immortal, because there will **always be a need to give a model access to information it didn't see during training.**



(*Timestamp: 00:02:45*)

Ben breaks down the acronym: **R**etrieval **A**ugmented **G**eneration. He points out that the original RAG paper actually described a process quite different from today's common interpretation, but the name stuck. At its core, the pipeline is simple: you **somehow** retrieve documents and pass them to a generative model to **augment** its output. He argues that there will always be a need to provide models with external information they weren't trained on.

1.4 The Standard RAG Flow

RAG Visualised

This is what a (simplified) LLM call looks like **without** RAG:



(Timestamp: 00:04:08)

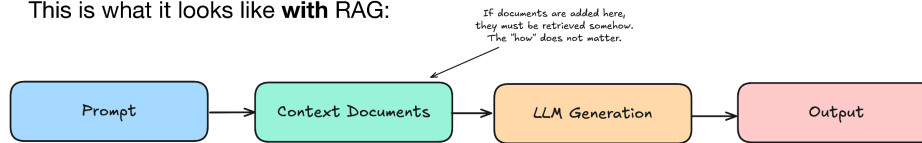
This slide presents a standard, simplified LLM call: a prompt goes in, the LLM generates a response, and an output comes out. This is the baseline for understanding how RAG changes the process.

RAG Visualised

This is what a (simplified) LLM call looks like **without** RAG:



This is what it looks like **with** RAG:



(*Timestamp: 00:04:28*)

With RAG, a new step is inserted: “Context Documents.” Ben emphasizes that the “how” of retrieving these documents doesn’t matter for the definition. If you’ve added external documents to the context window to augment the generation, you’re doing RAG. Even manually copy-pasting text into a prompt is, technically, a form of RAG.

1.5 The Problem with “2023 RAG”

But is RAG not Dead?

I have it on good authority that it is because Claude Code doesn't use it

- The often repeated idea that **RAG is Dead** comes from **confusing definitions**
- In 2023 and a good chunk of 2024, there was a lot of marketing incentives to **push confusing-but-catchy definitions**.
- What you might think of as **RAG** is actually the **most naive possible approach to semantic search**.
- Semantic search is **only one of many retrieval tools**, and **basic single-vector semantic search** is only one of its many forms.
- Claiming RAG is dead because we are adding other retrieval tools is akin to claiming **HTML is dead because we are now using CSS**.



(*Timestamp: 00:05:10*)

Ben addresses the common argument that since tools like Claude Code don't use “RAG,” RAG must be dead. What people often call “RAG” is a naive brute force, single-vector semantic search. This definition was pushed heavily by marketing in 2023-2024 because it was simple to sell. Claiming RAG is dead because we're now using better retrieval tools is, in his words, “akin to claiming HTML is dead because we are now using CSS.”

1.6 Single-Vector Search Limitations

Why is Single-Vector Semantic Search not Good Enough?

The real question is “how does one vector even work so well?”



Good luck trying to explain to your general purpose embedding model that it has to accurately capture the meaning of this

- Single-vector semantic search, where you embed a whole document (or chunk) into just one vector, is inherently limited.
- It has to compress a whole document's worth of meaning into **just a handful of dimensions**.
- This leads to **information loss**, prioritising what the **model assumes will be useful**.
- This assumption will be **informed by its training data** (and things like tokenisation), which most likely **does not look like your data**.

(*Timestamp: 00:06:46*)

Ben explains the limitations of single-vector search. It must compress the meaning of an entire document or chunk into a single, relatively small vector (e.g., ~1000 dimensions). This compression inevitably leads to information loss. The model is trained to prioritize information it assumes will be useful for matching queries to documents based on its training data (like Bing search data), which most likely does not look like your specific, domain-heavy data (e.g., a unique codebase). This mismatch is why general-purpose embedding models often struggle with specialized domains like code retrieval.

1.7 Why Long Context Doesn't Replace RAG

But is RAG not Dead?

I heard that context windows are so big we'll never use RAG again



- Let's assume that we manage to solve all efficiency concerns around attention: **10M context windows are right around the corner. That'll surely do it, right? RAG is dead? No.**
- Imagine someone in 1999 telling you that **Hard Drives are Dead because they're releasing 512MB RAM sticks soon!**
- Firstly, even 10M tokens is **remarkably little for many cases**: a lot of companies' knowledge bases are vastly larger than this, but, more importantly...
- RAG is the natural abstraction to provide a model with external information...

(*Timestamp: 00:09:45*)

Ben tackles the argument that massive context windows (e.g., Gemini's 1M or hypothetical 10M token windows) make RAG obsolete. He uses an analogy: it's like someone in 1999 claiming hard drives are dead because 512MB RAM sticks are coming soon. The reality is that even 10M tokens is a small amount of space for many enterprise knowledge bases or large datasets. Furthermore, the cost and inefficiency of stuffing everything into the context for every query makes it impractical.

1.8 Retrieval is Essential

RAG Lets Models Talk to The World

- Why did we and do we still need hard drives: because they **can store arbitrarily large, continuously updated volumes of information**.
- LLM weights are **inherently frozen**: the only way to add new information to them are **complex training processes**.
 - —> They **do not know anything about** Project X, Company Policy Y, `super_cool_but_broken_script.py`, or this really cool new `fasthtml` library you want to try!
 - And we wouldn't want them to! Space within weights is finite, and storing one-off information permanently is a waste of that space.
- Retrieval, and thus **RAG**, is **the natural way to provide external information when it is needed**.

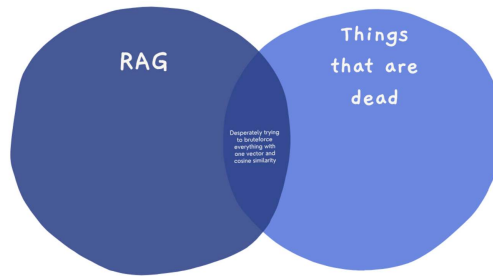
(*Timestamp: 00:12:15*)

Retrieval is never going away. LLM weights are frozen at a point in time. They don't know about your new internal project, your updated company policy, or that “really cool new `fasthtml` library you want to try.” Training a model on every new piece of information is complex and inefficient. Ben argues we wouldn't want models to store all this one-off information permanently anyway; we want their finite weight space to be used for intelligence, not just knowledge storage. Retrieval is necessary to inject this external, up-to-date information.

1.9 Key Takeaways

tl;dr

- So, to sum up:
 - RAG isn't dead and will not die anytime soon.
 - Naive methods are showing limitations, **encouraging us to use better ones.**
 - RAG is needed because it's the **best way to provide up-to-date information to models.**
 - Long context windows will not kill RAG.
- Thank you for coming!



(*Timestamp: 00:14:32*)

Ben summarizes takeaways so far: - RAG isn't going away. - Naive methods (like basic cosine similarity) are showing their limits, pushing us toward better, more sophisticated retrieval techniques. - RAG is the best way to provide models with up-to-date information. - Long context windows are not a replacement for retrieval.

The Venn diagram illustrates that what's "dead" is the oversimplified idea of brute-forcing everything with a single vector, not RAG generally.



Is that it?

(*Timestamp: 00:15:31*)

Classic Ben - more surprises coming!



Not quite.

(*Timestamp: 00:15:52*)

There is more to discuss re: better retrieval methods.

1.10 Better RAG is the Solution

Something did kill hard drives

- 512MB RAM sticks **did not**, in fact, kill hard drives.
- But something eventually did (as far as consumers are concerned)
- Now, what is it that killed hard drives, if more RAM didn't?
- That's right, the **SSD!** But what exactly is an SSD?
- If you ask a normal person, the answer you'll likely get is:
"Uh, they're like, better hard drives? they're faster or something?"
- It's exactly the same for **RAG**: what killed "2023 RAG" is **better RAG**

(*Timestamp: 00:15:54*)

Ben returns to his analogy. While more RAM didn't kill hard drives, SSDs did (for consumers). An SSD is just a "better hard drive." Similarly, what killed "2023 RAG" is simply **better RAG** (and concretely, better forms of retrieval).

1.11 The Retrieval Landscape

Have You Heard About the Many Facets of Retrieval? The Dizzying Overview

- We've established that **RAG** is simply the act of **Retrieving Context** for a model, **no matter how it is retrieved**.
- Retrieval takes many forms, all of them valid:
 - `grep` is a retrieval tool
 - `wget` is a retrieval tool
 - agentic search is a smarter approach to retrieval
 - BM25 and keyword search are retrieval tools
 - ColBERT and late-interaction are retrieval tools
 - Web Search is a retrieval tool
 - Reasoning is a retrieval tool
 - **And yes, simple semantic search is also a retrieval tool**
 - ...
- All of these should be used when relevant, and preferably, they should be used together

(Timestamp: 00:17:03)

To showcase the breadth of retrieval, Ben lists a variety of tools: `grep`, `wget`, agentic search, BM25, ColBERT, web search, and even reasoning. These are all valid retrieval methods. **The best approach often involves using them in combination.**

Have You Heard About the Many Facets of Retrieval?

Let's discuss a few

- Phew! That was a long list, and it doesn't even cover everything.
- The main thing to remember is that **Retrieval is More Relevant Than Ever**, it just **doesn't look like what people used to try to sell you**.
- Retrieval doesn't help itself: there's **new approaches coming everyday**, and everyone is **telling you that the hot new thing is so much better using metrics and benchmarks you've never even heard of before**.
- Don't worry! We've handpicked a few hot topics that some of my favourite people will tell you all about...

(Timestamp: 00:18:20)

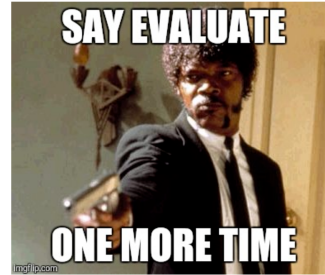
Ben acknowledges the overwhelming landscape of retrieval techniques. It's no longer the simple, one-trick pony it was once marketed as. To help navigate this, he introduces the upcoming speakers who will cover specific "hot topics."

1.12 Expert Speakers Preview

All Retrieval Needs Evaluations

Nandan Thakur

- Retrieval evaluations are difficult (ask Hamel and Shreyal)
- Building your own evals is necessary, but **you can't try out all approaches**
 - —> You need to **have benchmarks you can trust** to pick your candidate methods.
 - High-quality benchmarks have historically been Academia's role.
- In the LLM era, many popular benchmarks, such as BEIR/MTEB, now give weaker signal: **they are in the training set of all base models**.
- ... However, there's no need to worry, academics are resourceful: **Nandan Thakur**, who led the design of BEIR, will tell you all about **his new approaches for non-overfitted, trustable benchmarks**.



(*Timestamp: 00:19:34*)

The first guest expert introduced is [Nandan Thakur](#). With thousands of retrieval approaches available, trustworthy benchmarks are important. However, popular benchmarks like BEIR and MTEB are now part of the training data for all base models, leading to data contamination and giving a weaker signal. Nandan, who led the design of BEIR, will discuss his new approaches to creating non-overfitted, trustable benchmarks, such as the continuously updated **FreshStack**.

It'd be cool if retrievers could reason, someone should do that

Orion Weller

- Reasoning is all the rage nowadays.
- It's a form of **increased test-time compute**: we agree to spend more in exchange for better performance.
- This raises a question: **how does retrieval fit in a world where models can ramble on about their thoughts?**
- Orion Weller has been leading research onto this exact topic.
- **Hear directly from him whether or not Retrievers can think, or even use the reasoning of other models**



(*Timestamp: 00:21:15*)

Next up is [Orion Weller](#), who researches the intersection of reasoning and retrieval. How does retrieval fit into a world where models can “ramble on about their thoughts”? Orion will explore whether retrievers can think or use the reasoning of other models to improve performance.

If single-vector retrieval is so bad, then what's good?

Antoine Chaffin

- Someone on X (likely me) has probably tried to convince you to use “ColBERT” or “ColPali”.
- These models are called **late-interaction** or **multi-vector** model
- ... odds are some of you aren't quite sure what they do, and you might be afraid to ask at this point.
- Short answer: They're **very strong semantic search models**, which mitigate a lot of the issues single-vector models have (with different tradeoffs!)
- Long answer: **Antoine Chaffin**, one of the leading experts in this space, will be telling you **all about them** and **his work to make them dead(unlike RAG)-simple to use**.



(*Timestamp: 00:22:24*)

[Antoine Chaffin](#) will discuss late-interaction and multi-vector models like ColBERT. These models address the information loss of single-vector methods by using a vector for each token. Antoine will explain how they work and introduce his work on ColPali to make these powerful models easy to use.

The French Have a Saying: The Best Soups Are Made In Old Pots

Bryan Bischof and Ayush Chaurasia

- Data is inherently multi-modal: text is fun, but it's even more fun to find graphs and tables!
- Just like for text, there's a lot of new fancy semantic search techniques to retrieve multimodal documents...
- ... But the song remains the same: on their own, they still fall short!
- Join Ayush and Bryan as they tell you all about how they whipped up their best Multimodal Search recipe by learning from the **ANCIENT TOOLS**.



(*Timestamp: 00:23:39*)

The final talk features [Bryan Bischof and Ayush Chaurasia](#) on multimodal search. They'll explain that for multimodal data like graphs and tables, naive semantic search alone is insufficient. They will share how they created their best multimodal search recipe by combining modern techniques with “ancient tools.”

That's all folks!

Thank you for coming :)
Be sure to attend the much more interesting upcoming sessions

(Timestamp: 00:24:54)

You can sign up for the series with the links above, or here: [p2: Evals](#), [p3: Reasoning](#), [p4: Late-Interaction](#), and [p5: Multimodal](#).

1.13 Video

Here is the full video:

https://youtu.be/Evlk9J-B_uc

2 Modern IR Evaluation for RAG

Based on a presentation by Nandan Thakur

[Nandan Thakur](#) is a researcher at the University of Waterloo and a key contributor to major Information Retrieval (IR) benchmarks, including BEIR and the new FreshStack. His talk explains why traditional IR evals designed for search engines may be insufficient for RAG systems. He argues that LLM-generated answers often carry different retrieval goals which necessitate different IR metrics.

2.1 Introduction and Speaker Background



Modern IR Evaluation in the RAG Era

RAG Mini-Series: Guest Lecture #2

July 2, 2025



Nandan Thakur
Fourth-year Ph.D. Student
University of Waterloo

David R. Cheriton School of Computer Science
University of Waterloo

(*Timestamp: 00:00:00*)

The title slide for Nandan’s talk, “Modern IR Evaluation in the RAG Era.”

Hi, I'm Nandan Thakur

Previously (2018–2021):

- Undergraduate from BITS Pilani Goa
- Data Scientist at KNOLSKAPE
- NLP Research Assistant at UKP Lab, TU Darmstadt



Taken inside the MC (math) building at University of Waterloo

Current (2021 onwards):

- PhD Student at University of Waterloo
- Internships: Google Research, Databricks

You'll probably recognize me from:



Twitter (@beirmug)



RAG
TREC Retrieval-Augmented Generation

2

(Timestamp: 00:00:14)

Nandan introduces himself as a fourth-year Ph.D. student at the University of Waterloo. He outlines his background, including research at UKP-TU and internships at Google Research and Databricks. He highlights his work on the BEIR, MIRACL, and FreshStack benchmarks, and the TREC RAG track.

An Overview of my talk today

History: Traditional IR Evaluation

- History in IR Benchmarking: TREC & Cranfield Paradigm.
- Examples of Traditional IR Test Collections.
- What is zero-shot evaluation & the BEIR benchmark.
- Focus on the limitations of existing benchmarks such as BEIR.

10 minutes

Part I: Why IR Evaluation needs to change in the RAG era?

- Traditional IR setup versus IR in the RAG era.
- A comparison of modern-day RAG user versus traditional search user.
- Diversity-focused metrics in RAG & a paradigm shift.
- Synthetic benchmarks versus realistic ones.

5-10 minutes

Part II: A modern-day IR Evaluation Benchmark: FreshStack!

- FreshStack: A realistic & grounded question answering benchmark.
- Motivation & main components in the FreshStack benchmark.
- A complete insight on how FreshStack is created; evaluations used.
- Walkthrough on a Colab on how to evaluate on FreshStack via PyLate & BEIR.

Conclusion & Future Work

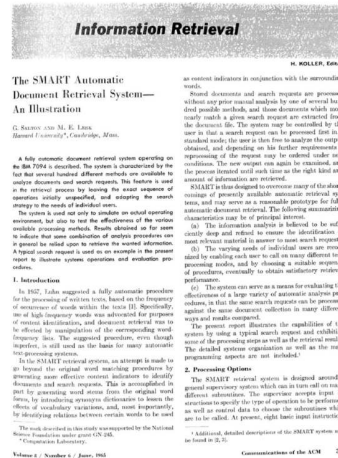
10-15 minutes

3

(*Timestamp: 00:01:09*)

Nandan outlines the presentation's three parts: a history of traditional IR evaluation, an explanation of why evaluation needs to change for RAG, and a deep dive into the FreshStack benchmark as a modern solution.

RAG is New but Information Retrieval is there since past 60 years!



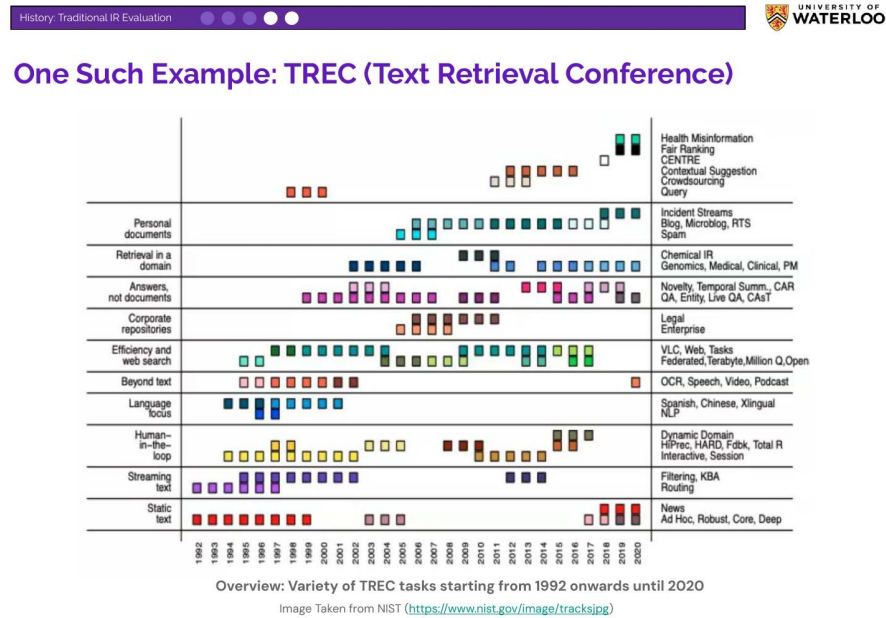
SMART Retrieval System. Published on June 1965!
Taken from <https://dl.acm.org/doi/10.1145/364955.364990>



The Classic Google Search Interface.

(Timestamp: 00:01:50)

Nandan emphasizes IR's history by showing a 1965 paper on the SMART Retrieval System, an early automated document retrieval system. He also introduces the Text Retrieval Conference (TREC), an influential conference since the 1990s that continues to produce IR benchmarks and standards.



6

(Timestamp: 00:03:00)

A diagram from NIST illustrates the breadth of TREC's evaluation tasks from 1992 to 2020. These tracks range from classic ad-hoc retrieval to specialized areas like multilingual search and human-in-the-loop evaluation, demonstrating the field's ongoing evolution.

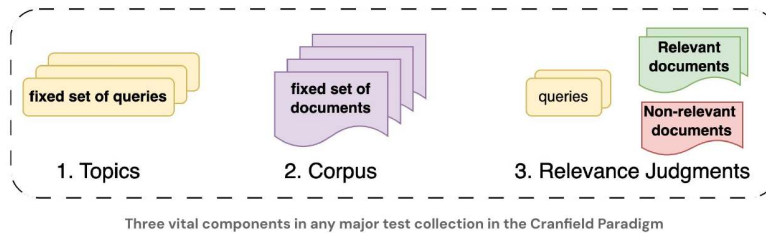
2.3 The Cranfield Paradigm

History: Traditional IR Evaluation



Traditional IR Evaluation: Cranfield Paradigm

- In the 1960s, **Cyril Cleverdon** and his colleagues at the College of Aeronautics (or Cranfield University) ran a series of experiments to evaluate the efficiency of **indexing systems**.
- These experiments are accepted set the standard which search engines use for evaluation of their systems!
- The **Cranfield paradigm** for IR evaluation crucially gave rise to **test collections for evaluation**.



7

(Timestamp: 00:03:54)

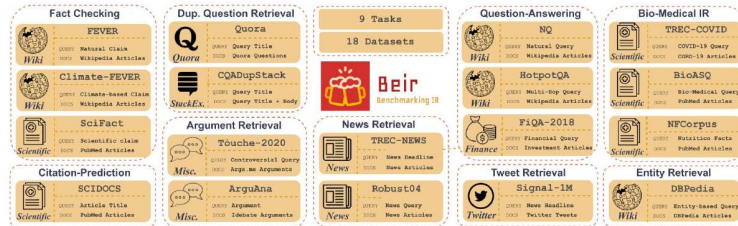
Nandan introduces the **Cranfield Paradigm**, the foundation of traditional IR evaluation developed in the 1960s. It established the concept of a **test collection**, comprising three components:

1. **Topics:** A fixed set of user queries.
2. **Corpus:** A fixed collection of documents.
3. **Relevance Judgments:** Human-annotated labels indicating which documents are relevant to which queries.

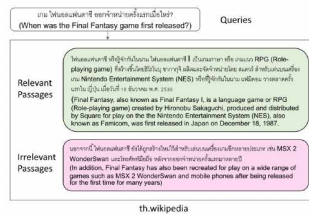
This three-part structure remains the basis for most IR benchmarks today.



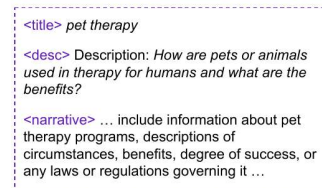
Examples of Test Collections for Retrieval Evaluation



BEIR benchmark contains a variety of test collections focused on different domains.



MIRACL contains a variety of test collections across various languages.



Example of a TREC test collection (query 794)

8

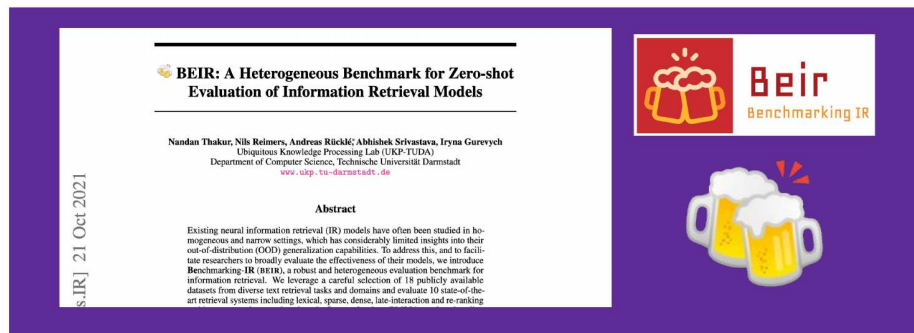
(Timestamp: 00:06:00)

Nandan shows examples of modern test collections. He highlights **BEIR** for its diversity of tasks, **MIRACL** for multilingual retrieval, and the typical **TREC** query structure, which includes a title, description, and detailed narrative.

2.4 The BEIR Benchmark



Case in point: The BEIR Benchmark



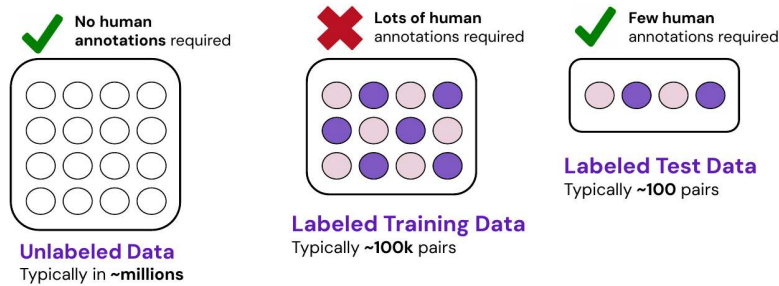
(Timestamp: 00:07:20)

This slide introduces the BEIR (Benchmarking-IR) benchmark, which was among the first to popularize zero-shot evaluation for retrieval models.

What is zero-shot evaluation and why is it important?



Why zero-shot is necessary? *High-quality labeled training data is scarcely available!*

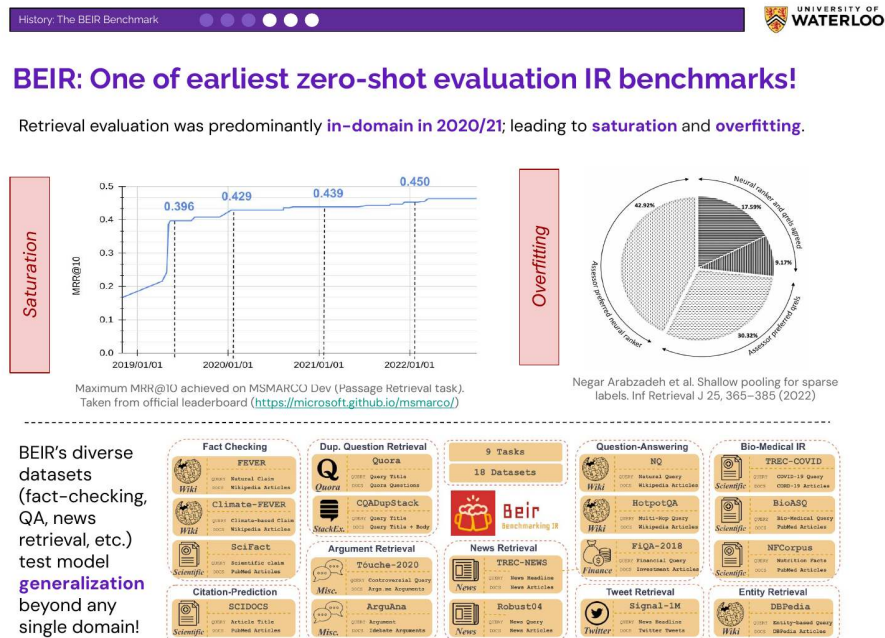


10

(Timestamp: 00:07:38)

Nandan explains **zero-shot evaluation**, where a model is tested on a domain or task it has not seen during training. This contrasts with *in-domain* evaluation (training and testing on similar data). Zero-shot evaluation is more realistic because high-quality, labeled training data for niche use cases is scarce and expensive to create.

2.5 Problems with Current Benchmarks

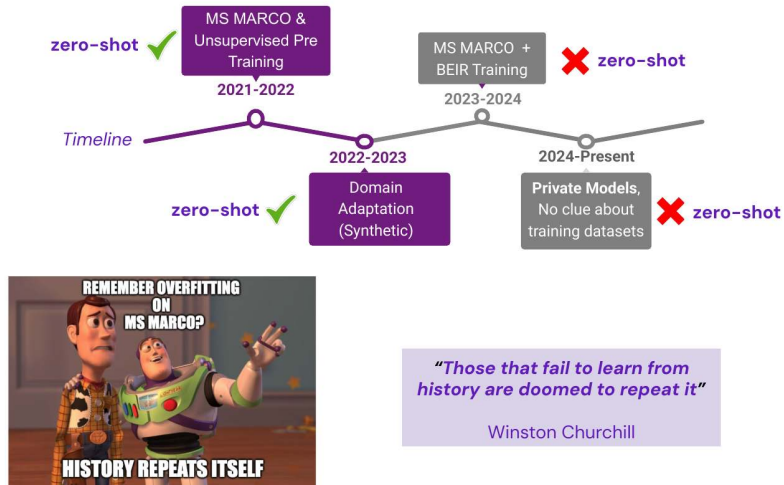


11

(Timestamp: 00:10:09)

Nandan explains the motivation for BEIR. Around 2020-2021, the field focused heavily on the MSMARCO dataset, leading to **saturation** (performance plateaus) and **overfitting**. BEIR was created to combat this by providing a diverse set of datasets to test a model's generalization ability beyond a single domain.

Observation #1: BEIR benchmark is sadly not “zero-shot” anymore



12

(Timestamp: 00:11:10)

Nandan explains that BEIR is no longer a truly “zero-shot” benchmark. Researchers now often include BEIR’s training sets in their model development pipelines. This, along with private models using unknown training data, repeats the overfitting problem that BEIR was designed to solve.

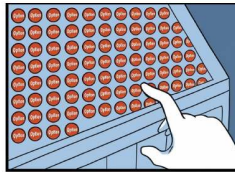
History: The BEIR Benchmark



Observation #2: Choosing a leaderboard model is not useful anymore!

- MTEB (retrieval) legacy or V1 leaderboard has over **400 models** with scores!
- There are about **10-15 models** within 1-2 points of average scores within each other.

I would ask: **top 20 models are all great on BEIR leaderboard**; but how do they perform on some other benchmark, task or niche domain?



Generated using GPT-4o

Overall	Embedding	Classification	Clustering	Pair Classification	Re-ranking	Retrieval	STS	Summarization	Multidimensional Classification	Retrieval v1
Retrieval is the task of finding relevant documents for a query.										
Retrieval English leaderboard #										
Metric: Normalized Discounted Cumulative Gain @ 10 (NDCG@10)										
Language: English										
Rank	Model	Model Size (Million Parameters)	Priority Score (0-1)	Embedding Dimensions	Max Tokens	Average	Argmax	CI-Score	CI-Score	CI-Score
1	openai-gpt-4o	7801	29.25	4096	32768	62.45	79.87	45.39		
2	meta-llama-3.1-70b	7111	26.49	4096	32768	62.15	83.08	45.43		
3	meta-llama-3.1-70b	7111	26.49	4096	32768	61.86	76.02	45.77		
4	meta-llama-3.1-70b	7809	26.33	3584	131072	61.71	84.86	41.83		
5	meta-llama-3.1-70b	1543	5.75	8192	131072	61.85	45.27	46.11		
6	meta-llama-3.1-70b	7111	26.49	4096	512	60.9	68.38	43.47		
7	meta-llama-3.1-70b	7111	26.49	4096	32768	60.76	77.32	44.62		
8	meta-llama-3.1-70b	7111	26.49	4096	32768	60.19	69.45	39.11		
9	meta-llama-3.1-70b	7111	26.49	4096	32768	60.18	62.34	34.43		

MTEB v1 legacy leaderboard showing the retrieval subset containing 15 datasets from the BEIR benchmark.

13

(Timestamp: 00:13:20)

Nandan highlights a practical issue: leaderboards are now too crowded to be useful. The MTEB leaderboard contains over 400 models, with the top contenders separated by marginal scores. This makes it difficult for practitioners to select a model and raises the question of how these models perform on other, more specialized tasks.

Summary: Limitations & Challenges in Test Collections, e.g., BEIR

Static versus Fresh evaluation data



They lack realistic question or answer distributions, e.g., HotPotQA

Paragraph A. Return to Olympus:

[1] Return to Olympus is the only album by the alternative rock band Malfunkshun. [2] It was released after the band had broken up and after lead singer Andrew Wood (later of Mother Love Bone) had died of a drug overdose in 1990. [3] Stone Gossard, of Pearl Jam, had composed the songs and released the album on his label, Longview Records.

Paragraph B. Mother Love Bone:

[4] Mother Love Bone was an American rock band that formed in Seattle, Washington in 1987. [5] The band was active from 1987 to 1990. [6] Frontman Andrew Wood's personality and compositions helped to catapult the group to the top of the burgeoning late 1980s/early 1990s Seattle music scene. [7] Wood died only days before the scheduled release of the band's debut album, "Apple", thus ending the group's hopes of success. [8] The album was finally released a few months later.

Q: What was the former band of the member of Mother Love Bone who died just before the release of "Apple"?

A: Malfunkshun

Supporting facts: 1, 3, 4, 6, 7

Peng Qi

Seven years ago, I co-lead a paper called HotpotQA that has motivated and facilitated many #AI #Agents research works since. Today, I'm asking that you stop using HotpotQA blindly for agents research in 2025 and beyond.

In my new blog post, I revisit the brief history of HotpotQA, why it defined a good problem in its historical context, some of my own research attempts at solving this problem that I believe are on the critical

Incomplete relevance judgments: "shallow labeling" problem!

Model	Original		+ Denoised		++ Post-hoc	
	nDCG@10	hole@10	nDCG@10	hole@10	nDCG@10	Δ inc.
BM25	0.367	61.6%	0.467	51.8%	0.785	Δ 0.418
CITADEL+	0.339	60.2%	0.362	62.5%	0.703	Δ 0.364
SPLADEv2	0.272	66.3%	0.326	63.3%	0.679	Δ 0.407
DRAGON+	0.249	69.2%	0.340	63.9%	0.718	Δ 0.469
Contriever	0.205	71.4%	0.303	65.9%	0.650	Δ 0.445
TAS-B	0.162	77.8%	0.306	67.5%	0.682	Δ 0.520

Taken from "Systematic Evaluation of Neural Retrieval Models on the Touché 2020 Argument Retrieval Subset of BEIR"

Overfitting & Saturation of Benchmarks



Taken from "Maintaining MTEB: Towards Long Term Usability and Reproducibility of Embedding Benchmarks"

(Timestamp: 00:14:43)

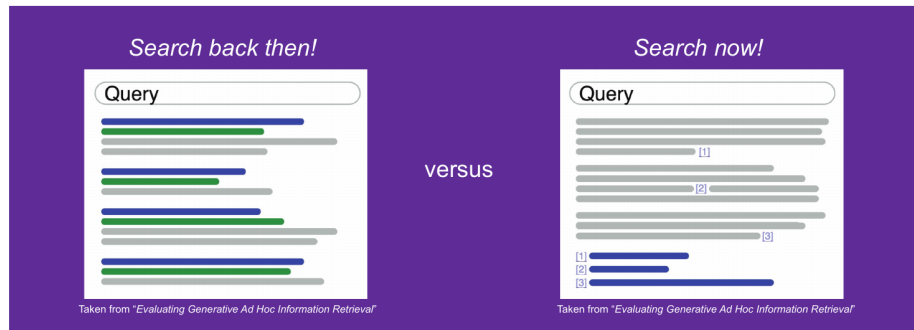
This slide summarizes the limitations of existing test collections like BEIR. They are often static, leading to data contamination risk. They can suffer from incomplete “shallow labeling” from human annotators. They may also lack realistic question distributions, prompting even the creators of benchmarks like HotpotQA to advise against their use for modern agentic systems.

2.6 The RAG Era Changes Everything

Part I: Why IR Evaluation Needs to Change in the RAG Era?



Part I: Why IR Evaluation needs to change in the RAG era?

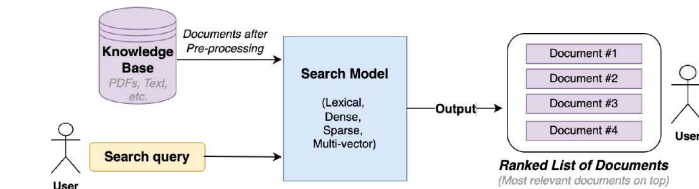


(Timestamp: 00:17:28)

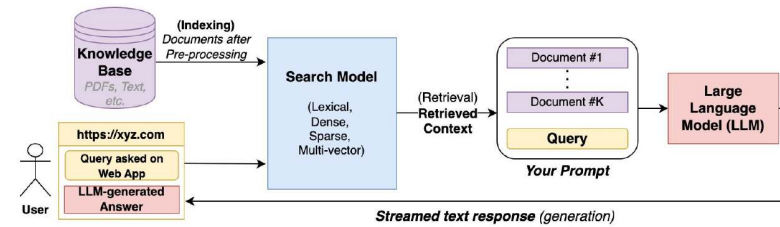
Nandan contrasts the old and new search paradigms. “Search back then” shows a ranked list of links, while “Search now” shows a generated answer block with citations, characteristic of RAG systems.

What has changed in Retrieval in the recent days?

Information Retrieval before the RAG Era



Information Retrieval in the RAG Era



16

(Timestamp: 00:18:20)

This slide diagrams the architectural shift. Before RAG, a search model returned a ranked list of documents to the user. In the RAG era, the search model provides retrieved documents as context to an LLM, which then generates a response for the user.

2.7 Different Users, Different Goals

Part I: Why IR Evaluation Needs to Change in the RAG Era?



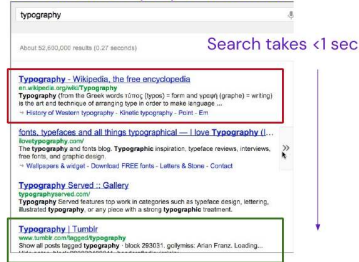
Mismatch between Traditional Search & Modern-day RAG users



Meet our **Traditional Search** user!

Very impatient and wants google search results returned quickly.

user asks short query on search bar



5 - 10 search results

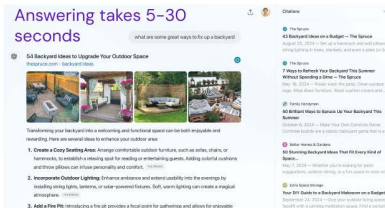
- User wants **quick results** and scans through **ranked results** from top to bottom.
- The user will most likely **click the first relevant link or search result**, according to Cresswell et al. [1]



Meet our **modern-day RAG** user!

Patiently waits until question has been answered by ChatGPT.

user can ask longer queries here



Answer summary with relevant citations

- **User waits patiently** for the answer to complete; **no more ranked list of results**.
- The user might **click on the link** to assess its source importance or to verify information.

17

(Timestamp: 00:19:10)

Nandan contrasts the two user types. A traditional search user is impatient, asks short queries, and scans a ranked list to click the first relevant link. A modern RAG user is patient, asks longer queries, and waits for a synthesized summary with citations, which they may use for verification.

2.8 The Evaluation Mismatch

Part I: Why IR Evaluation Needs to Change in the RAG Era?

Mismatch between Traditional Search & Modern-day RAG users



Meet our **Traditional Search** user!

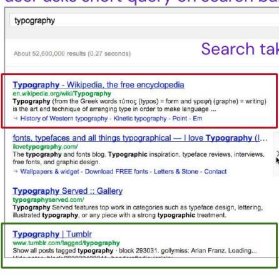
Very impatient and wants google search results returned quickly.



Meet our **modern-day RAG** user!

Patiently waits until question has been answered by ChatGPT.

user asks short query on search bar

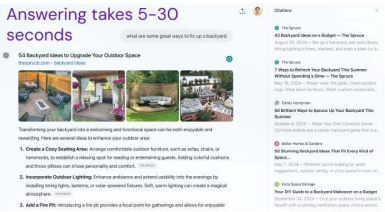


Search takes <1 sec

Objective: "did we rank the relevant page at #1?"

- ✓ **MRR** measures the position of the first relevant document (user which is likely to click).
- ✓ **NDCG** measures the rank of relevant documents in top-K results (user sees the search results)

user can ask longer queries here



Answering takes 5-30 seconds

Objective: "did we fetch every piece of evidence needed for the LLM to answer this question?"

- ✗ Should we use **MRR** here for evaluation?
(user doesn't click search results anymore)
- ✗ Should we use **NDCG** here for evaluation?
(user doesn't see a ranked list of search results from top to bottom)

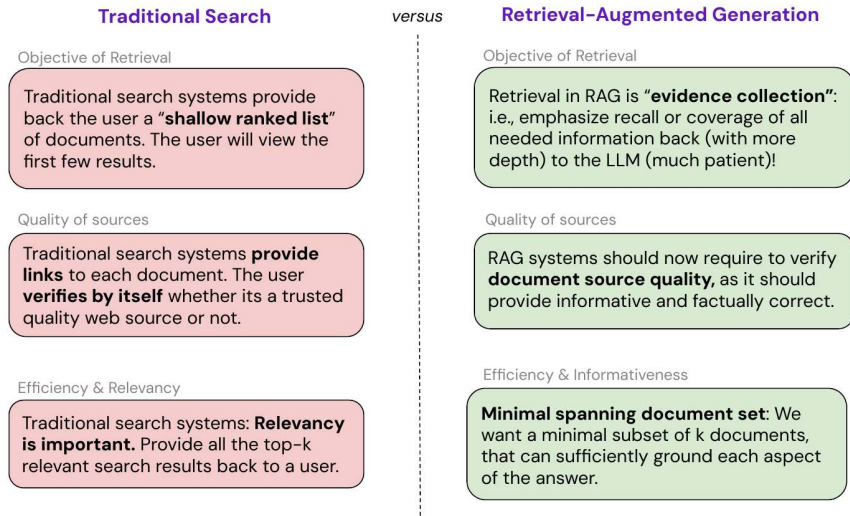
18

(Timestamp: 00:21:08)

This slide presents the talk's central argument. Traditional metrics like **MRR** (Mean Reciprocal Rank) and **NDCG** (Normalized Discounted Cumulative Gain) were designed for the traditional objective: "Did we rank the relevant page at #1?" The new RAG objective is: "Did we fetch *every piece of evidence* needed for the LLM to answer this question?" For this new goal, MRR and NDCG may be insufficient on their own, as they do not measure comprehensive evidence collection or redundancy.

43

Why metrics need to change to evaluate RAG systems?



19

(*Timestamp: 00:23:20*)

The argument is not to discard traditional relevance but to expand the evaluation criteria for RAG. While **Relevancy is [still] important**, it must now be balanced with new goals like finding a **minimal spanning document set**. This concept captures the need for a set of documents that is not only relevant but also comprehensively covers all aspects of an answer without being redundant.

2.9 Introducing FreshStack

Part II: A Deep dive into FreshStack



UNIVERSITY OF
WATERLOO



databricks

Part II: Modern-day IR Benchmarks: A deep dive into FreshStack

FreshStack: Building Realistic Benchmarks for Evaluating Retrieval on Technical Documents

Nandan Thakur^{1*} Jimmy Lin¹ Sam Havens²
Michael Carbin² Omar Khattab² Andrew Dzondov²
¹University of Waterloo, Canada ²Databricks, USA
<https://fresh-stack.github.io>

Abstract

We introduce FreshStack, a holistic framework for automatically building information retrieval (IR) evaluation benchmarks by incorporating challenging questions and answers. FreshStack conducts the following steps: (1) automatic corpus collection from code and technical documentation, (2) suggest generation from

**fresh
stack**

(Timestamp: 00:24:50)

Nandan introduces **FreshStack**, a modern IR benchmark developed with Databricks. It is designed to evaluate retrieval for RAG on technical documents.

Why we built FreshStack – Motivation?

Most academic RAG benchmarks suffer 3 things:

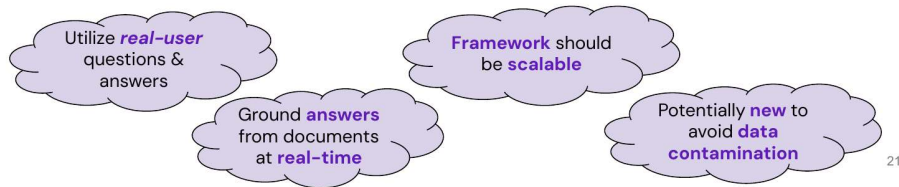
1. They lack **realistic** questions and/or **answer** distributions.
2. They are **artificially** easy because they're built as "**RAG**" datasets.
3. They are **static** and **unspecialized**.



When your RAG system looks great on paper but fails in production.

Problem Definition

RQ: Can we build an **automatic framework** to construct **realistic RAG benchmarks**?



21

(Timestamp: 00:25:00)

The motivation for FreshStack was to create a realistic RAG benchmark that overcomes the limitations of existing academic benchmarks, which are often static and artificially easy. The framework was designed to use real user questions, ground answers in real-time documents, be scalable, and be new to avoid data contamination.

2.10 FreshStack Data Sources

Part II: A Deep dive into FreshStack



How did we start building FreshStack — Queries?

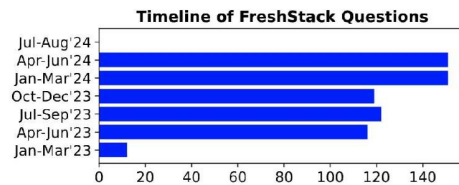
Where to get our queries?



Benefits of StackOverflow:

1. Users ask long & real-world questions
(not generated for RAG purposes)
2. Community maintains the **top-voted answer** in Stackoverflow
3. **Open-source**** and not static.

We source queries from **recent & niche topics** which are at a **low risk of decontamination** by LLMs.



We picked 5 topics:

1. LangChain (ML/NLP)
2. Laravel (Backend)
3. AngularJS (Front-end)
4. Yolo (Vision)
5. Godot (Game development)

22

(Timestamp: 00:26:16)

FreshStack sources its queries from **Stack Overflow**, an ideal source for long, complex, real-world questions with community-vetted answers. To mitigate data contamination, the benchmark uses questions from five recent and niche topics asked primarily in 2023 and 2024.

How did we start building FreshStack — Corpus?

Where to get our documents?

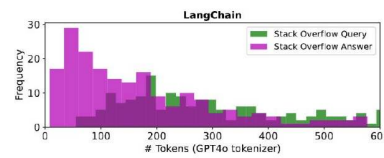
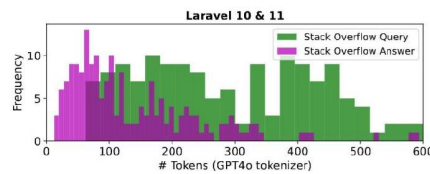


GitHub Repositories

Benefits of GitHub Repositories:

1. Constantly updated and evolved by the maintainer of the repository.
2. Contains code, text and all forms of technical documentation.
3. **Open-source**** and not static.

FreshStack queries are **quite long** (code snippets, outputs, text); can be even longer than **answers** themselves!



23

(*Timestamp: 00:27:30*)

The document corpus comes from the **GitHub Repositories** of the corresponding topics. This provides a constantly updated source of technical documentation and code. An interesting finding is that for technical queries, the questions can be significantly longer than the answers.

2.11 The FreshStack Pipeline

Part II: A Deep dive into FreshStack



Let's breakdown the FreshStack Pipeline

Step I: Nuggetization → Step II: Oracle Retrieval → Step III: Support w/ Nuggets

Nuggetization breaks down the Stackoverflow answer into multiple **statements or facts** which are essential in the RAG answer.

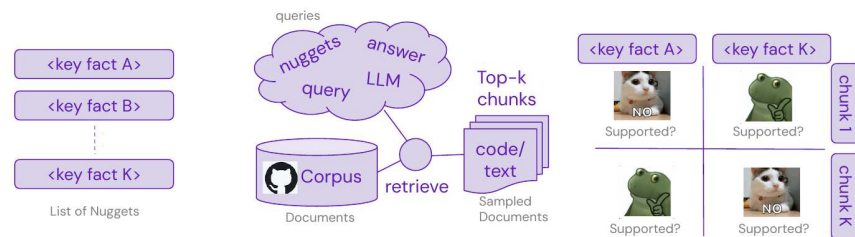
Use GPT-4o to generate nuggets!

Oracle Retrieval provides **document chunks** from a corpus containing **code-snippets** and documentation.

Uses BM25, E5, BGE & Voyage.

Grounding techniques provides **relevance judgements** of retrieved document chunks for each individual **nugget**.

Uses GPT-4o as a CoT Judge



24

(Timestamp: 00:28:18)

Nandan explains the three-step automated pipeline for building FreshStack:

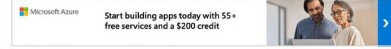
1. **Nuggetization:** A Stack Overflow answer is broken down by GPT-4o into essential, atomic facts or “nuggets.”
2. **Oracle Retrieval:** A diverse pool of candidate documents is retrieved from the corpus using a hybrid of models.
3. **Support w/ Nuggets:** A GPT-4o judge checks which retrieved document chunks support each individual nugget, creating fine-grained relevance judgments.

Step I: Nuggetization or Important Facts/Entities

Stackoverflow Question

Chromadb from_documents function giving error

Asked 7 months ago · Modified 7 months ago · Viewed 544 times



The following function was working till a few days ago but now gives this error:
ValueError: Expected EmbeddingFunction.__call__ to have the following signature: odict_keys(['self', 'input']), got odict_keys(['args', 'kwargs']) Please see <https://docs.trychroma.com/embeddings> for details of the EmbeddingFunction interface. Please note the recent change to the EmbeddingFunction interface: <https://docs.trychroma.com/migration-to-0416---november-7-2023>

```
def create_chromadb(link):
    embedding_function = SentenceTransformerEmbeddings(model_name="all-MiniLM-L6-v2")
    loader = TextLoader(link)
    documents = loader.load()

    # Split the documents into chunks (no changes needed here)
    text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
    chunks = text_splitter.split_documents(documents)

    # Update for new EmbeddingFunction definition
    # 0 is set to the type of documents (Text in this case)
    # 0 is set to the type of documents (Text in this case)
    embedding_function = EmbeddingFunction[0] = embedding_function

    # Initialize Chroma with the embedding function and persist the database
    db = Chroma.from_documents(chunks, embedding_function, persist_directory='./chroma_db')
    db.persist()
    print(f"Saved {len(chunks)} chunks")
    return db
```

docs.trychroma.com/migration-to-0416---november-7-2023

python langchain chromadb

Stackoverflow Answer

1 Answer

Sorted by: Highest score (default)

I slightly modify your code, using HuggingFaceEmbeddings instead of SentenceTransformerEmbeddings

```
from langchain_community.embeddings import HuggingFaceEmbeddings
embedding = HuggingFaceEmbeddings(
    model_name="sentence-transformers/all-MiniLM-L6-v2")

from langchain_community.vectorstores import Chroma
db = Chroma.from_documents(
    documents=chunks,
    embedding=embedding,
    persist_directory='./tmp/chroma_db')
db.persist()
```

FreshStack Nuggets (facts highlighting different parts in your "gold" answer)

The error is due to a mismatch in the function signature expected by 'Chroma.from_documents' when using SentenceTransformerEmbeddings.

Import 'HuggingFaceEmbeddings' from 'langchain_community.embeddings'

Initialize 'HuggingFaceEmbeddings' with the model name "sentence-transformers/all-MiniLM-L6-v2"

Pass the initialized 'HuggingFaceEmbeddings' to the 'Chroma.from_documents' function.

25

(Timestamp: 00:29:50)

This slide shows a concrete example of nuggetization. An answer to a **Chroma.from_documents** error is broken down into four key facts: the cause of the error, the required import, the initialization step, and the function call.

Step II & III: Oracle Retrieval & Nugget-level Support

FreshStack Nuggets

1 The error is due to a mismatch in the function signature expected by 'Chroma.from_documents' when using SentenceTransformerEmbeddings.

2 Import 'HuggingFaceEmbeddings' from 'langchain_community.embeddings'

3 Initialize 'HuggingFaceEmbeddings' with the model name "sentence-transformers/all-MiniLM-L6-v2"

4 Pass the initialized 'HuggingFaceEmbeddings' to the 'Chroma.from_documents' function.

Stackoverflow Answer

1 Answer

I slightly modify your code, using HuggingFaceEmbeddings instead of SentenceTransformerEmbeddings

```
from langchain_community.embeddings import HuggingFaceEmbeddings
embedding = HuggingFaceEmbeddings(
    model_name="sentence-transformers/all-MiniLM-L6-v2"
)
from langchain_community.vectorstores import Chroma
db = Chroma.from_documents(
    documents=documents,
    embedding=embedding,
    persist_directory="tmp/chroma_db"
)
db.persist()
```

Supports?



Sampled Doc #1

langchain / templates / intel-rag-seen / ingest.py

```
ccurme community[patch]: deprecate langchain_community Chroma in favor of la... X
```

```
Code 40 lines (38 loc) - 1.45 KB
```

```
1 import os
2
3 from langchain.text_splitter import RecursiveCharacterTextSplitter
4 from langchain_chroma import Chroma
5 from langchain_community.document_loaders import UnstructuredFileLoader
6 from langchain_community.embeddings import HuggingFaceEmbeddings
7 from langchain_core.documents import Document
8
9
10 def ingest_documents():
11     """
12     Ingest PDF to Redis from the data/ directory that
13     contains Edgar 10K filings data for Nike.
14     """
15     # Load list of pdfs
16     data_path = "data/"
17     doc = list_path(list(data_path, file) for file in os.listdir(data_path))
18
19     print(f"Parsing 10K filing doc for NOKS", doc)
20
21     text_splitter = RecursiveCharacterTextSplitter(
22         chunk_size=500, chunk_overlap=50, add_start_index=True
23     )
24     loader = UnstructuredFileLoader(doc, mode="single", strategy="Fast")
25     chunks = loader.load_and_split(text_splitter)
26
27     print(f"Done preprocessing. Created, len(chunks), {len(chunks)} chunks of the original pdf")
28
29     # Create vectorstore
30     embedding = HuggingFaceEmbeddings(
31         model_name="sentence-transformers/all-MiniLM-L6-v2"
32     )
33
34     documents = []
35     for chunk in chunks:
36         doc = Document(page_content=chunk.page_content, metadata=chunk.metadata)
37         documents.append(doc)
38
39     # Add to vectorstore
40     _ = Chroma.from_documents(
41         documents=documents,
42         collection_name="open-rag",
43         embedding=embedding,
44         persist_directory="tmp/open_rag_db",
45     )
```

26

(Timestamp: 00:30:50)

This slide illustrates the final steps. After a document is retrieved, the system checks which of the four nuggets it supports. This process creates nugget-level relevance labels, forming the basis for the new evaluation metrics.

2.12 FreshStack Evaluation Metrics

Part II: A Deep dive into FreshStack



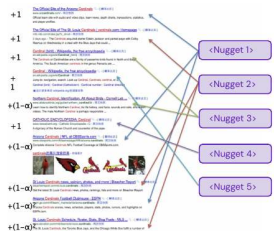
FreshStack Evaluation (Metrics & Definitions)

For RAG evaluation, we require “a minimal spanning document set which can cover each nugget or aspect required to answer the question”.

Diversity

(i) α -nDCG@10

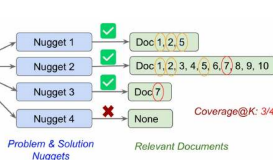
A diversity-oriented nDCG metric. Penalizes $(1-\alpha)^k$ for every k th relevant document supporting a “nugget”.



Grounding

(ii) Coverage@20

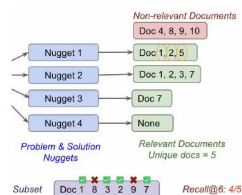
Calculated as the % of unique “nuggets” supported by the retrieved documents.



Relevance

(iii) Recall@50

Calculated as % of relevant documents in top 50 / total relevant documents.



27

(Timestamp: 00:31:26)

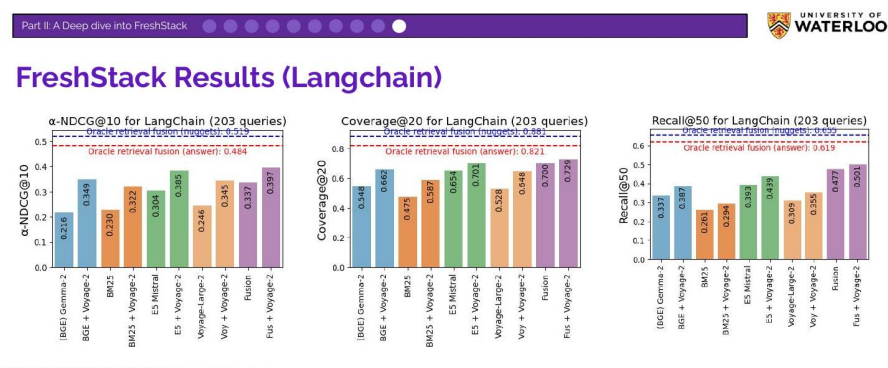
Nandan introduces the three metrics used in FreshStack, which provide a holistic view of RAG retrieval performance:

1. **Diversity (α -nDCG@10):** Measures non-redundancy, penalizing the retrieval of multiple documents that support the same fact.
2. **Grounding (Coverage@20):** Measures the percentage of unique nuggets supported by the retrieved documents, directly evaluating evidence collection.

3. **Relevance (Recall@50)**: A traditional metric that serves as a foundational check on whether the retrieved documents are on-topic.

This multi-faceted approach augments traditional relevance with metrics tailored to the specific goals of RAG.

2.13 Key Results and Findings



FreshStack Results highlights & Key Takeaways:

1. Current retrieval techniques (using raw questions) **are unable** to retrieve relevant documents.
2. No individual model performs best across all topics in FreshStack.
3. Rerankers **enhance** retrieval performance (voyage-2), however **not** without explicit fine-tuning.
4. Oracle approaches using **Nuggets/Answer** score much higher — there's plenty of headroom!

28

(Timestamp: 00:33:19)

Nandan presents results from the benchmark. Key findings include that current retrieval techniques struggle on these realistic tasks, and

no single model performs best across all topics. The large gap between current model performance and the theoretical “Oracle” maximum indicates significant room for improvement.

Part II: A Deep dive into FreshStack

Leaderboard & Google Colab <https://fresh-stack.github.io>

☒ Oracle (Stack Overflow Nuggets) ☒ Open Source ☒ Proprietary

Retriever	Size	Date	Arg (C)	LangChain	Yolo v7 & v8	Laravel 10 & 11	Angular 16, 17 & 18	GoD44
			cd@10	cd@20	cd@50	cd@10	cd@20	cd@50
Oracle: Fusion (BM25, ...)	-	2024-11-01	0.541	0.868	0.755	0.519	0.881	0.855
Oracle: BM25 (Nuggets)	-	2024-11-01	0.628	0.760	0.556	0.467	0.739	0.446
Oracle: Voyage Large 2 (Nuggets)	-	2024-11-01	0.404	0.750	0.586	0.419	0.753	0.528
Oracle: BGE (Gemini 2) (Nuggets)	9B	2024-11-01	0.389	0.735	0.547	0.398	0.667	0.465
Qwen3-4B (Embed)	8B	2025-06-05	0.365	0.689	0.525	0.331	0.694	0.423
Qwen3-4B (Embed)	8B	2025-06-05	0.367	0.656	0.49	0.32	0.675	0.415
Fusion (BM25, BGE, ES, Voyage)	-	2024-11-01	0.343	0.669	0.539	0.337	0.7	0.477
Oracle: ES (Mistral-7B) (Nuggets)	7B	2024-11-01	0.337	0.664	0.486	0.323	0.484	0.432
Stella-1.5B-v5	1.5B	2025-01-01	0.317	0.615	0.479	0.315	0.66	0.388
Voyage Large 2	-	2024-11-01	0.289	0.589	0.438	0.246	0.528	0.308
Jina V4 (Embed)	3.8B	2024-06-24	0.282	0.584	0.425	0.277	0.596	0.379
Stella-400M-v5	400M	2025-01-01	0.276	0.578	0.422	0.285	0.608	0.356
BGE (Gemini-2)	9B	2024-11-01	0.269	0.569	0.427	0.216	0.548	0.337
Qwen3-0.8B (Embed)	590M	2025-06-05	0.262	0.543	0.384	0.259	0.588	0.369
ES (Mistral-7B)	7B	2024-11-01	0.255	0.553	0.397	0.304	0.654	0.393
OpenAI text-embedding-3-large	-	2024-01-25	0.248	0.537	0.373	0.267	0.507	0.282
Jina V5 (Embed)	570M	2024-09-18	0.227	0.515	0.344	0.223	0.533	0.299
GTE (large) v1.5	434M	2024-01-09	0.226	0.494	0.318	0.206	0.47	0.252
BM25	-	2024-11-01	0.218	0.448	0.316	0.23	0.475	0.261
None embed (Code)	7B	2025-02-24	0.218	0.488	0.348	0.224	0.518	0.292
OpenAI text-embedding-3-small	-	2024-01-25	0.203	0.48	0.33	0.213	0.523	0.283
CodellamaEmbed	137M	2024-11-03	0.104	0.279	0.162	0.099	0.271	0.128

Results of different retrieval models across datasets on FreshStack. The best-performing model in each category is in bold, and the second best is underlined.
cd@10 (alpha-nDCG@10) measures retrieval diversity & relevance. **cd@20** (Coverage@20) measures fraction of nuggets supported by retrieved documents, **R@50** (Recall@50) measuring % of relevant documents present in retrieved documents.

[Download CSV](#) [Download JSON](#)

https://colab.research.google.com/drive/1eaB_5cF62kW4E3g0bCXw9_4nC4xt-tX?usp=sharing

29

(Timestamp: 00:34:55)

Nandan shares the public FreshStack leaderboard and a Google Colab notebook. The notebook provides a script for users to evaluate their own models on FreshStack using its multi-dimensional metrics.

So what did we learn today?

- The traditional Cranfield paradigm **is useful for traditional retrieval evaluation** but misses out on evaluating critical aspects of modern RAG applications.
- Benchmarks like BEIR encourage **zero-shot evaluation**, but are stale now with **extensive leaderboard overfitting**, providing a very low signal in terms of benchmark usability.

- The traditional search and the RAG setup has a mismatch. Users **don't see ranked results anymore** and retrieved results are **evidence collection for the LLM**.
- Metrics need to change for evaluating retrieval in RAG systems, as good RAG retriever needs to provide documents from **relevant, diverse, informative** and **correct** sources.

- FreshStack is a **modern IR evaluation** benchmark, containing realistic and recent user-sourced queries that continuously updates. Queries & answers are sourced from **real-users!**
- FreshStack breaks the answer into facts (nuggets) and conducts evaluation using nuggets based on multiple modern IR metrics, including diversity, coverage, and relevance.



USES NDCG,
MRR FOR RETRIEVAL
EVALS IN RAG

USES RECALL
FOR RETRIEVAL
EVALS IN RAG

USES DIVERSITY,
INFORMATIVENESS, RECALL
AND FAITHFULNESS FOR
RETRIEVAL EVALS IN RAG



30

(Timestamp: 00:36:01)

This slide summarizes the talk's main points. Traditional IR evaluation may be insufficient for RAG depending on the use case. Benchmarks like BEIR are now suffering from overfitting. Often, the goal of RAG retrieval is evidence collection, requiring metrics that evaluate diversity, informativeness, and correctness in addition to relevance.

**Beir**
Benchmarking IR

**MIRACL**
Multilingual Information Retrieval Across a Continuum of Languages



Thank you for listening!

I would like to thank my FreshStack co-authors, **Omar Khattab**, **Andrew Drozdov**, **Sam Havens**, **Michael Carbin** and **Jimmy Lin**!

Next, I thank **Ben**, **Hamel** and **Shreya** for inviting me as a guest lecturer!

Lastly, I would like to thank **Xueguang**, **Crystina** and **Shivani** for providing me feedback on this presentation.



**fresh stack**

**RAG**
TREC Retrieval-Augmented Generation

31

(*Timestamp: 00:37:15*)

Nandan concludes by thanking his collaborators. The slide's meme reinforces his message: good evaluations are essential for developing better models.

2.14 Chapter Reflections

Nandan's message was to consider other retrieval metrics beyond relevance based on your product's needs. He argued that we must sometimes reconsider what "good" retrieval means. For the stack overflow use case, he considered multiple dimensions of performance:

- **Grounding (or Coverage):** Did the retrieval system fetch *all* the evidence needed to construct a complete and accurate answer? A missing fact can lead to an incomplete or incorrect generation, even if the retrieved documents are otherwise highly relevant.
- **Diversity:** Are the retrieved documents efficiently informative? Retrieving multiple documents that repeat the same information is less valuable than retrieving a set of documents that each contribute a unique and essential fact.
- **Relevance:** Is the retrieved information on-topic? This remains a fundamental check. A diverse and well-grounded set of documents is useless if it pertains to the wrong subject.

This is not a call to discard traditional metrics but to **augment** them. The FreshStack benchmark, with its blend of Recall, Coverage, and Diversity metrics, is an example of this.

2.15 Video

Here is the full video:

<https://youtu.be/Trps2swgeOg>

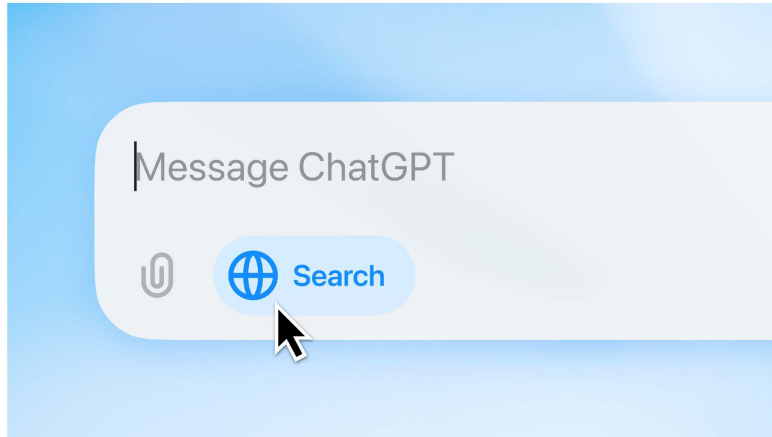
3 Optimizing Retrieval with Reasoning Models

Based on a presentation by Orion Weller

[Orion Weller](#) from Johns Hopkins University focuses on embedding the instruction-following and reasoning capabilities of modern Large Language Models (LLMs) directly into the retrieval process.

In his talk, Orion argues that while LLMs have improved RAG, the core retrieval step has remained static. He introduces a paradigm where instruction-following and reasoning are baked directly into retrieval models, a fundamental shift from using LLMs for query rewriting or as generic rerankers.

3.1 LLM Capabilities: Instruction Following and Reasoning



2

(Timestamp: 00:00:18)

The talk begins by highlighting the user-facing interfaces of modern LLMs like ChatGPT, which have set new expectations for how we interact with AI. One key capability of LLMs is **instruction following**: executing complex, multi-part natural language instructions with high fidelity.

Instruction Following

Generate a haiku about information retrieval in the style of a pirate and mention RAG.

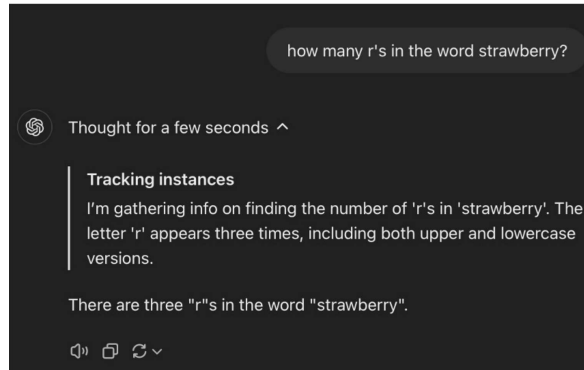
Scourin' the sea wide,
RAG finds treasure in the fog—
Answers hoist with pride!

3

(Timestamp: 00:00:36)

Orion shows the result of a pirate-themed haiku prompt. The model successfully adheres to all constraints: it generates a haiku, maintains a pirate style, and mentions “RAG,” demonstrating a level of instruction following that is a recent and significant advancement.

Reasoning



4

(*Timestamp: 00:00:58*)

A second key capability is **reasoning**, also known as test-time compute or “thinking.” The slide shows a model verbalizing its thought process to solve a problem, generating intermediate “thinking tokens” that outline its step-by-step logic before providing the final answer.

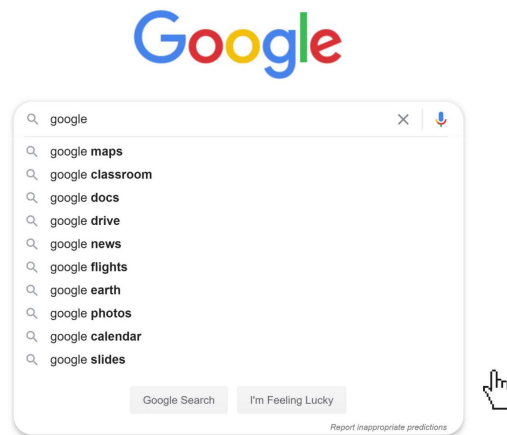
3.2 The Search Paradigm Hasn't Changed



6

(*Timestamp: 00:01:52*)

To illustrate how little the search paradigm has changed, Orion shows Google's interface from 1999.



7

(*Timestamp: 00:01:58*)

He contrasts it with a modern Google search bar. Despite 26 years of development, the fundamental interaction remains the same: a user types keywords, and the system matches them to return a list of links.

Search really hasn't changed

Despite using LLMs, we're just adding a wrapper around the results

9

(Timestamp: 00:02:46)

Despite the interface, Orion argues the underlying retrieval process has not evolved. Even in advanced systems, the LLM is often just a “wrapper.” The system sends the query to a traditional search engine, gets back a standard list of results, and then uses the LLM to summarize them. The retrieval step itself hasn't gained the new capabilities of the LLM.

3.3 Evolution of Search Paradigms

Keyword Search

Query

Find websites
explaining data privacy

Documents

Data Encryption Standards
www.nist.gov/standards/



Wolves Outside Your Data
www.janerodgers.blog/wolves



Digital Protection
www.clearlaw.net/digital



11

(*Timestamp: 00:03:58*)

To illustrate current limitations, Orion starts with **Keyword Search**, which relies on exact lexical matching. Given a query and three documents, keyword search matches “Data Encryption Standards” and “Wolves Outside Your Data” because they contain the keyword “data.”

It fails to retrieve “Digital Protection” because it lacks the keyword “data,” even though “digital” is semantically similar.

Semantic Search

Query

Find websites
explaining data privacy

Documents

Data Encryption Standards
www.nist.gov/standards/



Wolves Outside Your Data
www.janerodgers.blog/wolves



Digital Protection
www.clearlaw.net/digital



12

(*Timestamp: 00:04:11*)

The next evolution is **Semantic Search**, which matches based on meaning, often by representing queries and documents as vectors in a shared semantic space. A good semantic search model would retrieve all three documents, as it understands the relationship between “data” and “digital,” and “privacy” and “protection.”

Instruction-based Search

Query

Find websites
explaining data privacy
**and uses extended
metaphors**

Documents

Wolves Outside Your Data 
www.janerodgers.blog/wolves

Data Encryption Standards 
www.nist.gov/standards/

Digital Protection 
www.clearlaw.net/digital

14

(*Timestamp: 00:05:25*)

Orion introduces **Instruction-based Search**, where the query is a nuanced command. The user wants to find documents about data privacy that also use “extended metaphors.”

An instruction-based search system should understand this meta-level constraint and retrieve only the “Wolves Outside Your Data” document, which uses a metaphorical title. It correctly identifies that the other two documents, while topically relevant, do not meet the stylistic instruction.

Prompt and Reasoning-based Search

Query

Find websites
explaining data privacy
**and uses extended
metaphors. Have
really high recall or I
will lose my job**

Documents

Wolves Outside Your Data 
www.janerodgers.blog/wolves

Data Encryption Standards 
www.nist.gov/standards/

Digital Protection 
www.clearlaw.net/digital

15

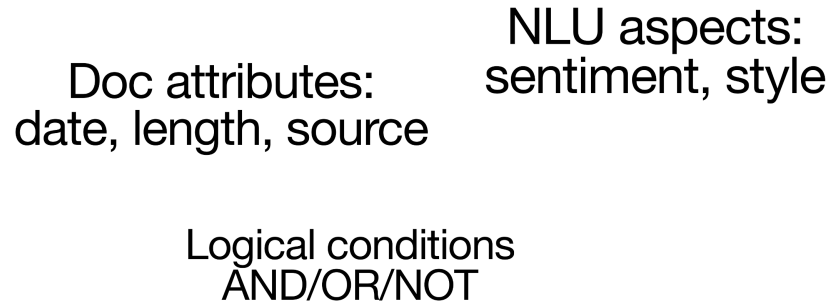
(*Timestamp: 00:06:16*)

Orion pushes the concept to its extreme with **Prompt and Reasoning-based Search**. The query now includes instructions about the desired *behavior* of the search engine, such as “Have really high recall or I will lose my job.”

A traditional search engine would misinterpret this, likely searching for documents containing the word “recall.” An advanced, reasoning-based retriever should understand the user’s intent and adjust its retrieval strategy.

3.4 Understanding Instructions in IR

What is an instruction in IR?



16

(Timestamp: 00:06:42)

What is an instruction in the context of IR? Orion breaks it down into several categories:

- **Document attributes** like date, length, or source
- **NLU aspects**, such as document sentiment or writing style
- **Logical conditions**, combining multiple constraints with operators like AND, OR, and NOT

The space of possible instructions mirrors the complexity of natural language.

3.5 Introducing Promptriever and Rank1



Promptriever
fast embedder



Rank1
strong but slow

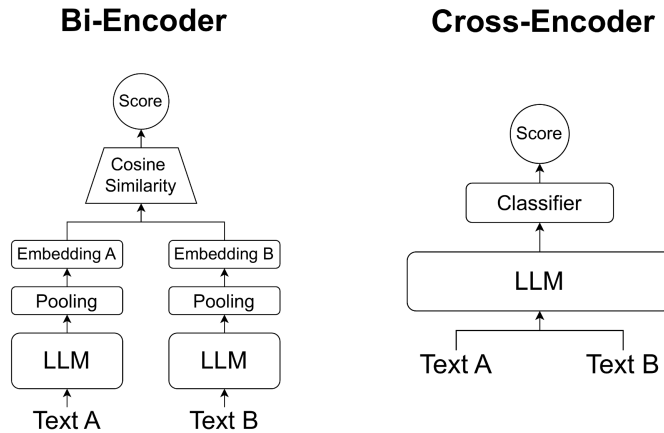
19

(*Timestamp: 00:07:45*)

Orion introduces two models from his research that embody these principles:

1. **Promptriever**: A fast embedding model for following instructions during initial retrieval
2. **Rank1**: A powerful but slower reranker that uses reasoning and test-time compute for nuanced relevance judgments

3.6 Promptriever: Instruction-Trained Retrieval



21

(*Timestamp: 00:08:23*)

Orion explains the two main retrieval architectures. A **Bi-Encoder** (dense retriever) creates separate query and document embeddings for fast comparison, making it highly scalable. A **Cross-Encoder** (reranker) processes the query and document together for deeper interaction at a higher computational cost. Promptriever is a bi-encoder.

Can we make promptable retrievers?

Can bi-encoders take instructions?



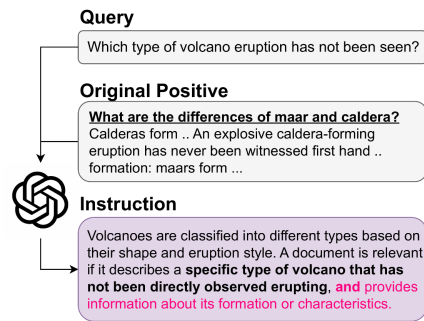
Training data for instruction-following

22

(*Timestamp: 00:09:10*)

The main research question was how to enable fast, scalable bi-encoders to understand complex instructions. The missing ingredient was **training data**. Existing retrieval datasets like MSMARCO lack instructions because users don't type them into traditional search engines.

Generating Training Data



We have to generate synthetic instructions

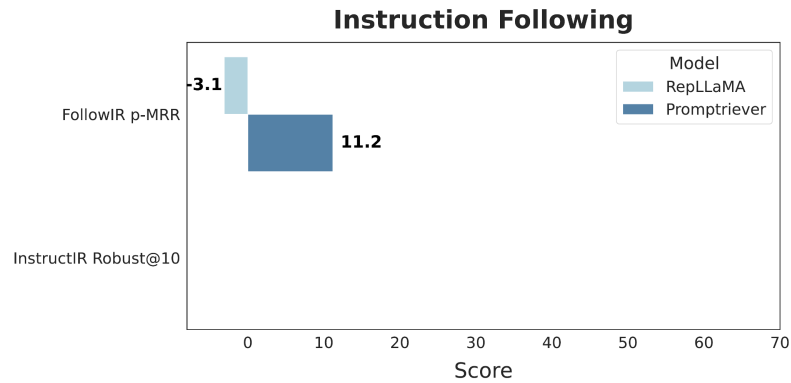
25

(Timestamp: 00:10:07)

This slide illustrates the process of generating the training data, starting with a standard query. The process uses an existing query-document pair from a standard dataset and uses an LLM to generate a detailed **instruction** that makes the relevance criteria more specific. A crucial part was also generating **instruction negatives** - documents that are relevant to the query but irrelevant to the *instruction*.

3.7 Promptriever Evaluation Results

Results

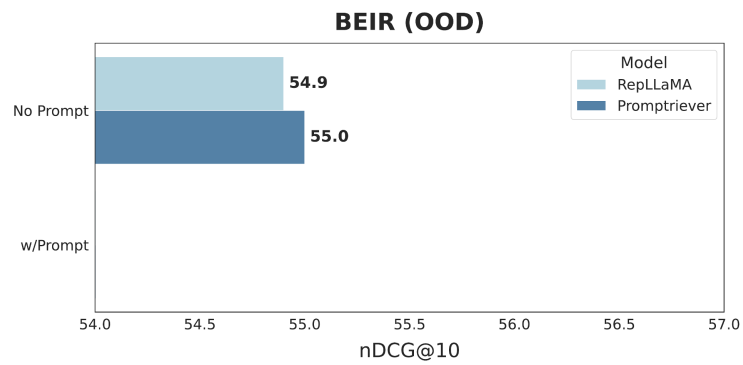


32

(*Timestamp: 00:12:36*)

On FollowIR, the baseline RepLLaMA (and all prior embedding models) scored negatively, performing *worse* when given an instruction. Promptriever is the first to achieve a positive score, demonstrating that bi-encoders can learn to follow instructions.

Results

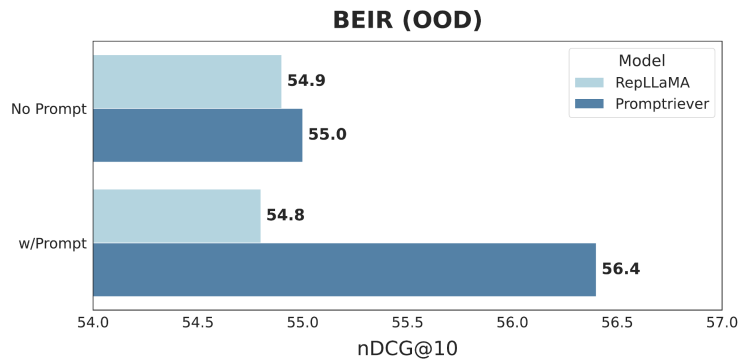


36

(*Timestamp: 00:13:58*)

Without a prompt, Promptriever performs comparably to the RepLLaMA baseline, showing that instruction-following capabilities don't hurt performance on traditional tasks.

Results



37

(*Timestamp: 00:14:13*)

When a generic instruction is added, Promptriever's performance increases significantly, while the baseline's degrades slightly. This demonstrates that Promptriever's retrieval strategy can be controlled with natural language - a form of **zero-shot hyperparameter optimization via prompting**.

3.8 Rank1: Reasoning-Based Reranking



Rank1: Test-Time Compute for Information Retrieval



Orion Weller



Kathryn Ricci



Eugene Yang



Andrew Yates



Dawn Lawrie



Benjamin
Van Durme



43

(*Timestamp: 00:16:08*)

The focus now shifts to Rank1, the reasoning-based model. The associated paper's title is "Rank1: Test-Time Compute for Information Retrieval," highlighting its focus on reasoning in the reranking stage.

What does test-time compute look like in IR?



do snow leopards change color



Snow leopards have long, thick fur, and their base color varies from smoky gray to yellowish tan, with whitish underparts ...



<think> The user is asking if snow leopards change color. The passage provided describes their fur color as varying from smoky gray to yellowish tan, with some patterns. Hmm, the key here is whether "varies" implies that they change color over time, like seasonally or due to some factors.

But wait, "varies" might just mean that different individuals have different base colors, not that a single leopard changes its color. The passage doesn't mention anything about changing color; it's just describing their appearance. ...

Therefore, the passage isn't directly relevant to the question of whether they change color. So the answer should be false.

</think> **false**

48

(*Timestamp: 00:17:08*)

This slide shows what the reasoning process looks like in information retrieval. Given a query and a document, the model generates a detailed reasoning trace, identifying key phrases, analyzing the relationship between query and document, and questioning its own interpretations before arriving at a final judgment.

What does this look like?

Example:
Can you find a
similar
leetcode
problem?

Model Reasoning

<think>

Okay, let's tackle this problem. The user is asking whether the given passage is relevant to their query about finding problems with similar data structures or algorithms. The query specifically mentions problems that share similar data structures (like arrays, trees) or algorithms (DFS, DP, sorting, traversals). The example problem is about the rainwater trapping problem, which uses a two-pointer approach.

Now, looking at the passage provided, it's about the "Max Area" problem, which also uses a two-pointer approach. Both problems involve arrays and solving them with a two-pointer technique. The rainwater problem uses a two-pointer approach to find the maximum trapped water, while the max area problem does the same to find the maximum area between vertical lines.

The key here is that both problems share the same algorithmic approach (two pointers) and similar data structures (arrays). Even though the problems are different (trapping rain vs. max area), the underlying algorithm and data structure similarities make them relevant. The user is looking for problems that share these similarities, so the passage does fit that criteria. Therefore, the answer should be true

</think> true

51

(Timestamp: 00:18:50)

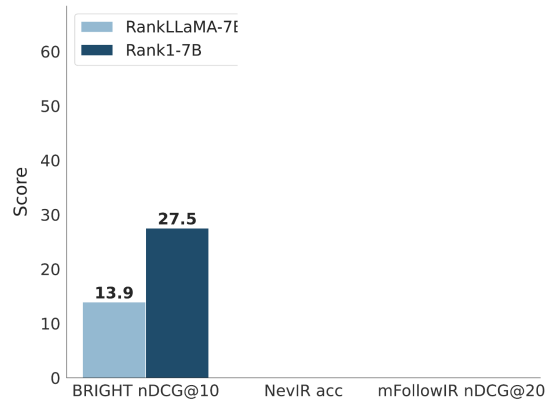
This slide shows Rank1's reasoning on a LeetCode problem. Asked to find a similar problem, it correctly identifies the core "two-pointer approach" algorithm in the provided document and recognizes that the candidate document also uses the same technique, demonstrating a deep, algorithmic level of understanding.

3.9 Rank1 Performance Results

Results

We evaluate on a broad range of tasks:

- BRIGHT (*reasoning*)
- NevIR (*negation*)
- mFollowIR (*instructions*)



54

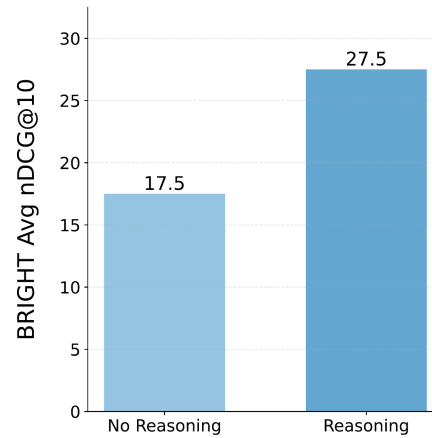
(*Timestamp: 00:19:38*)

The evaluation covers tasks testing reasoning (BRIGHT), negation (NevIR), and instruction following (mFollowIR). The baseline model, RankLLaMA, was trained on **10 times more data** than Rank1. Despite being trained on far less data, Rank1 nearly doubles the performance of the baseline on the BRIGHT reasoning benchmark.

Results

What about a direct comparison of the reasoning chain?

Same data, same model, no chain



58

(*Timestamp: 00:20:16*)

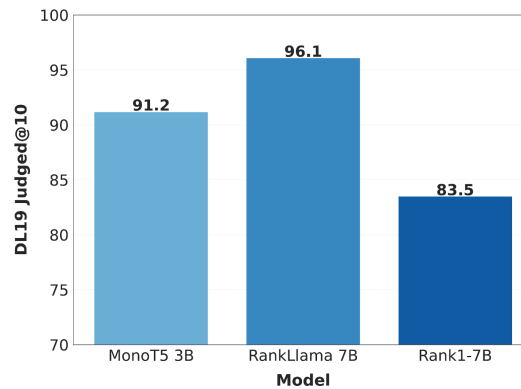
To isolate the impact of the reasoning chain, they compared training the same model on the same data, with and without the “thinking” part of the training examples. The results show that training the model to generate the reasoning chain leads to a massive 10-point gain in performance. The act of “thinking” itself unlocks these advanced capabilities.

3.10 Finding Novel Relevant Documents

Results

We were surprised by the low initial scores on DL19/DL20

Judged@10 is significant less!



60

(*Timestamp: 00:20:44*)

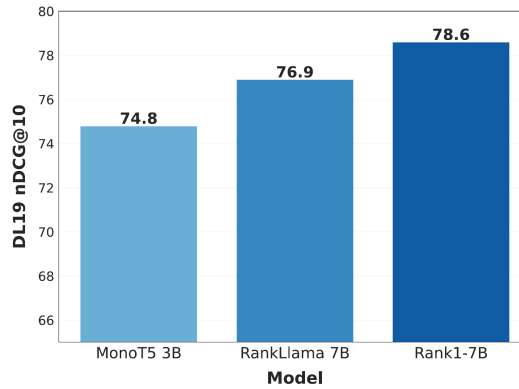
They were surprised by low scores on the DL19/DL20 datasets, discovering their model was finding many documents that had never been judged by human annotators because older systems had never retrieved them.

Results

We re-judged the
unjudged +
incorrect preds
for all models

Perhaps the community
should move on to newer
eval datasets (**DL19 was
before BERT!**)

It's finding **new docs that
previous systems didn't**
— that are relevant!



61

(*Timestamp: 00:21:39*)

Reasoning-based models are not just improving scores on old benchmarks; they are **finding new, relevant documents** that previous systems missed. This also suggests the IR community should move on from older evaluation datasets as they may not be equipped to measure modern model capabilities.

3.11 Chapter Takeaways

The goal: IR that works like LLMs

Query

Find websites
explaining data privacy
and uses extended
metaphors. Have
really high recall or I
will lose my job

Documents

Wolves Outside Your Data 
www.janerodgers.blog/wolves

Data Encryption Standards 
www.nist.gov/standards/

Digital Protection 
www.clearlaw.net/digital

64

(Timestamp: 00:22:37)

Orion concludes that the overall goal is to create IR systems that work like LLMs, capable of handling queries that combine topic, style, and behavioral instructions.

Key insights from this work:

1. **Promptriever** enables fast bi-encoder retrievers to follow complex instructions through specialized training data
2. **Rank1** uses reasoning chains to achieve superior performance on complex retrieval tasks
3. Both models demonstrate that LLM capabilities can be successfully integrated into retrieval systems
4. Reasoning-based models discover novel relevant documents that traditional systems miss

New retrievers can directly benefit from rapid LLM advancements. As LLMs get better at reasoning and instruction following, so will the retrieval systems built upon them.

3.12 Video

Here is the full video:

<https://youtu.be/YB3b-wPbSH8>

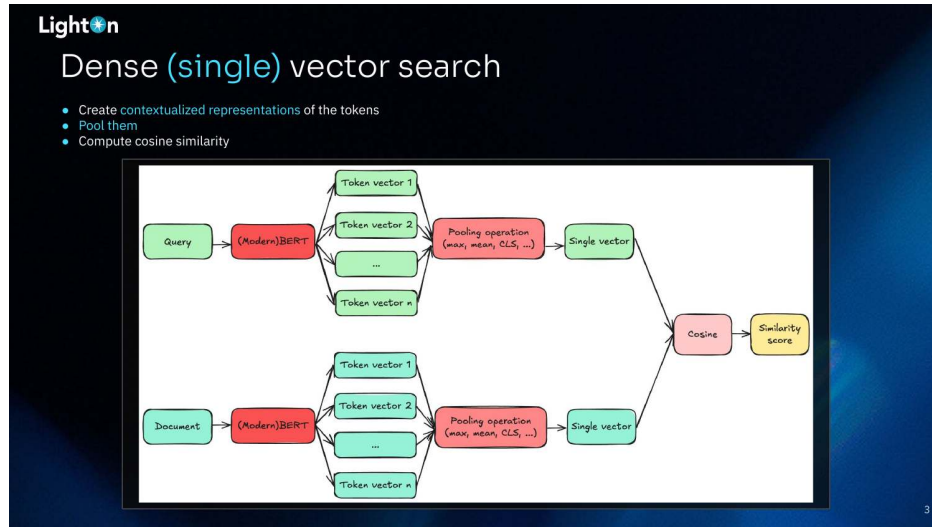
4 Late Interaction Models For RAG

Based on a presentation by Antoine Chaffin

[Antoine Chaffin](#), a researcher at LightOn, contributed to impactful open-source tools like [ModernBERT](#) and [PyLate](#), a library for working with late-interaction models.

His talk explains the intrinsic limitations of single-vector search, such as information loss from pooling, and introduces late interaction models as a more powerful alternative for modern RAG use cases like out-of-domain generalization and long context retrieval.

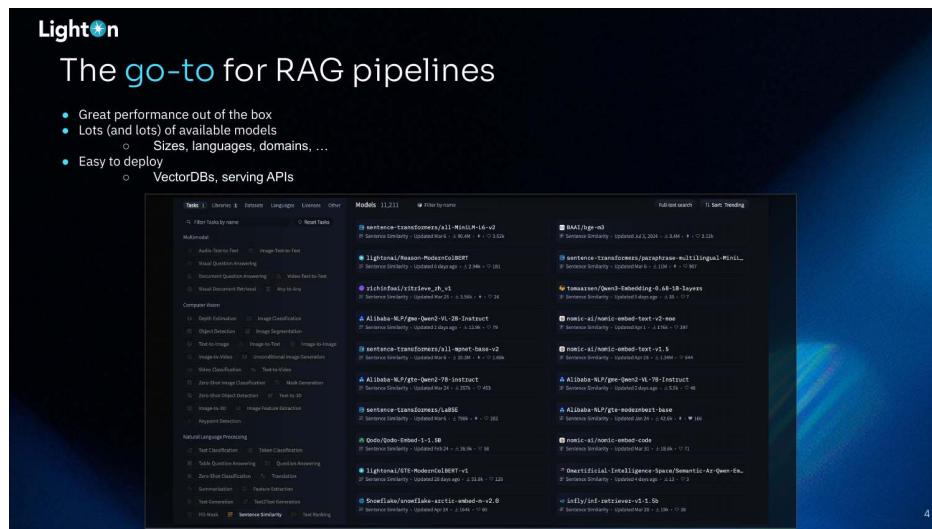
4.1 Dense Vector Search Architecture



(Timestamp: 00:03:07)

This slide diagrams the standard architecture for dense (single) vector search. A query and a document are separately fed through an encoder model (like BERT) to generate contextualized vector representations for each token. A pooling operation (e.g., max, mean, [CLS] token, etc.) then compresses all these token vectors into a single vector for the query and a single vector for the document. Finally, a similarity score (typically cosine similarity) is computed between these two vectors to determine relevance. The information loss in the pooling step is a key limitation of this approach.

4.2 Why Dense Models Became Popular



(Timestamp: 00:03:54)

Dense vector search has become the standard for RAG pipelines for several reasons. It offers strong out-of-the-box performance, and a vast number of pre-trained models are available on platforms like the Hugging Face Hub, catering to different sizes, languages, and domains. Furthermore, these models are easy to deploy using the growing ecosystem of vector databases and serving APIs.

4.3 The Benchmark Problem

The screenshot shows a presentation slide with a dark blue background. At the top left is the 'Lightn' logo. The title 'BEIR: a school case of Goodhart's law' is in white. Below it is a bulleted list of points. In the center is a white rectangular box containing the title 'BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models' and the authors' names and affiliations. A small number '6' is in the bottom right corner of the slide.

Lightn

BEIR: a school case of Goodhart's law

- Big challenge of ML is generalization: models can perform well in-domain but totally fail on data they have not been trained on
- BEIR introduced various benchmarks to evaluate out-of-domain performance
- Few years and thousands of models later: the hill has been climbed
- **Run your own evaluations!**

BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models

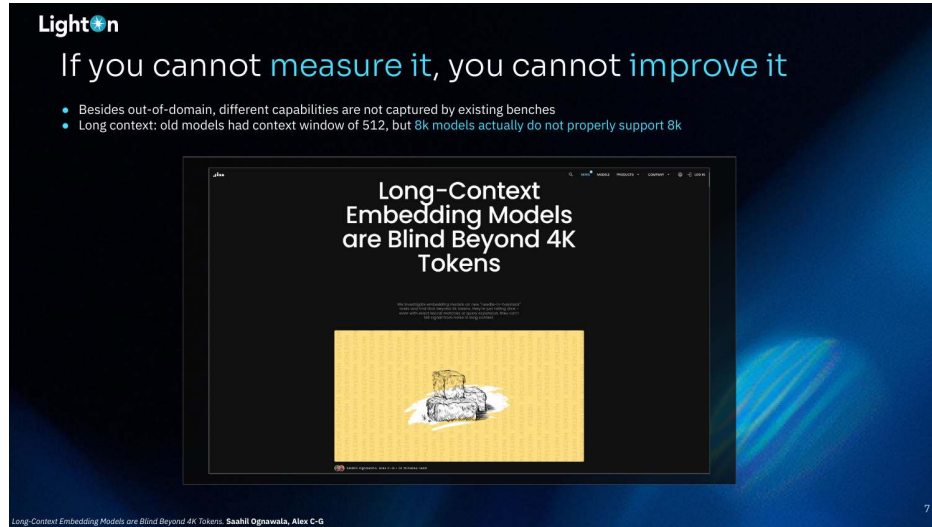
Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, Iryna Gurevych
Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
www.ukp.tu-darmstadt.de

6

(Timestamp: 00:04:17)

Antoine uses the BEIR benchmark as an example of Goodhart's Law in action. BEIR was introduced to evaluate the out-of-domain generalization of retrieval models. However, as it became the standard benchmark to beat, models began to overfit to its specific datasets. Consequently, top-performing models on the BEIR leaderboard may not generalize well to new, unseen use cases, underscoring the importance of running your own evaluations on your specific data.

4.4 Hidden Limitations



(*Timestamp: 00:05:36*)

Antoine argues that if you cannot measure a capability, you cannot improve it. Existing benchmarks often miss important aspects of model performance. For instance, most older models were evaluated with a context window of only 512 tokens. While many newer models claim to support 8k tokens, recent evaluations have shown that their performance degrades significantly beyond 4k, a limitation that was not captured by older benchmarks.

4.5 Long Context Performance

Lightn

If you cannot **measure it**, you cannot **improve it**

- Besides out-of-domain, different capabilities are not captured by existing benches
- Long context: old models had context window of 512, but 8k models actually do not properly support 8k

Model	Param.	CTX Len.	Synthetic (Acc@1)		Real (nDCG@10)				Avg.
			Paskey	Needle	NQA	QMS	SFD	WQA	
<i>512 Context Models</i>									
E5 _{base} (Wang et al., 2022)	110M	512	38.0	28.5	25.3	23.8	74.7	55.8	41.0
E5-RoP _{base}	110M	512	38.5	31.5	24.6	23.2	66.6	38.8	40.9
GTE _{base} (Li et al., 2023)	110M	512	31.0	24.5	28.6	21.8	55.8	47.3	34.8
BGE _{base} (Xiao et al., 2023)	110M	512	18.0	25.3	25.6	22.4	60.3	51.7	33.9
Contexter (Jacovi et al., 2021)	110M	512	38.5	29.0	26.7	25.5	73.5	47.3	40.1
GTR _{base} (Ni et al., 2022)	110M	512	38.5	26.3	26.5	18.3	63.7	52.2	36.5
<i>≥ 4k Context Models</i>									
E5-Mistral (Wang et al., 2023b)	7B	4,096	71.0	48.3	44.6	43.6	96.8	82.0	64.4
Jina-V2 (Glincher et al., 2023)	137M	8,192	50.3	54.5	37.9	38.9	93.5	74.0	58.2
Noemie-V1 (Nouhoum et al., 2024)	137M	8,192	60.7	39.5	41.2	36.7	93.0	73.8	57.5
BGE-M3 (Chen et al., 2024)	568M	8,192	59.3	40.5	45.8	35.5	94.0	78.0	58.9
OpenAI-Ada-002	-	-	50.8	36.8	41.1	40.0	91.8	80.1	56.8
<i>Our Extended Models</i>									
E5 _{base} + Tuning (4k)	110M	4,096	67.3	41.5	30.4	35.7	95.2	69.2	56.6
E5-RoP _{base} + SelfExtend (4k)	110M	4,096	72.5	53.5	32.3	39.1	91.9	74.6	60.8
E5-Mistral + NTK (32k)	7B	32,768	93.8	66.8	49.8	49.2	97.1	95.2	75.3

Table 2: Results (%) of existing and extended embedding models on LongEmbed. NQA, QMS, SFD, WQA is short for NarrativeQA, QMSum, SumScreenFD, 2WikiMultiHopQA, respectively. We show that context window extension can effectively improve existing embedding models in processing long context.

LongEmbed: Extending Embedding Models for Long Context Retrieval. Dawei Zhu, Liang Wang, Nan Yang, Yifan Song, Wenhao Wu, Furu Wei, Sujian Li

(Timestamp: 00:06:24)

This table from the LongEmbed paper illustrates the performance of various embedding models on long-context retrieval tasks. It shows that extending models with techniques like SelfExtend or NTK can significantly improve their ability to handle long contexts, with the E5-Mistral + NTK model achieving the highest average score.

4.6 Complex Retrieval Tasks

Lightn

If you cannot **measure it**, you cannot **improve it**

- Besides out-of-domain, different capabilities are not captured by existing benches
- Long context: old models had context window of 512, but 8k models actually do not properly support 8k
- Retrieval beyond keyword/semantic matching is challenging

Level 1: Keyword-based Retrieval

Query: Who sang the song "I Think We're Alone Now"?

Relevance: Keyword matching

Positive Document: "I Think We're Alone Now" is a song ... released by the American recording artists Tony Danza and the Shondells ...

Level 2: Semantic-based Retrieval

Query: How human activities influence climate system?

Relevance: Semantic matching

Positive Document: Deforestation and urbanization result in increased emissions, urban heat island effects and changes in natural water cycle.

Level 3: Reasoning-based Retrieval - BRIGHT

Query: Sustainable Living - post: At home, after I water my plants, the water goes to plates below the pots. Can I reuse it for my plants next time?

Relevance: Risk of using recycled plant water.

Positive Document: Sustainable Living - post: Soluble salts are commonly found in soils. When they build up, they destroy the soil structure and cause direct damage to roots ...

Code - issue: I have this table and need to transform it to ... I don't like UNPIVOT, is there a better function in snowflake for this?

Relevance: Alternative function.

Positive Document: The function `FLATTEN` flattens (explodes) compound values into multiple rows ... `FLATTEN` INPUT => <expr> ...

MATH - question: Let $k=2005^2+2^2+2008$. What is the units digit of k^2+2^2+k ?

Relevance: Uses the same theorem.

Positive Document: Determine all positive integers relatively prime to all the terms of the infinite sequence $a, a+2, a+2^2, a+2^3, \dots$.

LongEmbed: Extending Embedding Models for Long Context Retrieval. Dawei Zhu, Liang Wang, Nan Yang, Yifan Song, Wenhao Wu, Furu Wei, Sujian Li

(Timestamp: 00:06:33)

Retrieval goes beyond simple keyword or semantic matching. Modern RAG systems require more complex, reasoning-based retrieval. For example, a query asking for a different Snowflake function than UNPIVOT requires understanding the function's purpose, not just matching keywords. Similarly, a math question might require retrieving a document that uses the same theorem, even if the numbers are different.

Lightn

If you cannot **measure** it, you cannot **improve** it

- Besides out-of-domain, different capabilities are not captured by existing benches
- Long context: old models had context window of 512, but 8k models actually do not properly support 8k
- Retrieval beyond keyword/semantic matching is challenging

Table 2: The performance of retrieval models on BRIGHT. We report nDCG@10 for all datasets: Biology (Bio.), Earth Science (Earth.), Economics (Econ.), Psychology (Psy.), Robotics (Rob.), Stack Overflow (Stack.), Sustainable Living (Sust.), Text-to-Code (Text.), Poetry, ArtPS, TheoremQA with question retrieval (TheoQ) and with theorem retrieval (TheoT). Avg. denotes the average score across 10 datasets. The best score on each dataset is shown in bold and the second best is underlined. We show that reasoning-intensive retrieval is challenging for current retrievers, where the best model only achieves an nDCG@10 score of 24.3 on average. Model details are in Appendix A.1

	StackExchange					Coding		Theorem-based			Avg.		
	Bio.	Earth.	Econ.	Psy.	Rob.	Stack.	Sust.	Text.	Poetry	ArtPS	TheoQ	TheoT	
<i>Spore model</i>													
BM25	18.9	27.2	14.9	12.5	13.6	18.4	15.0	24.4	7.9	6.2	10.6	4.9	14.5
<i>Open-sourced models (<1B)</i>													
BGE	11.7	24.6	16.6	17.5	11.7	10.8	13.3	26.7	5.7	6.0	13.0	6.9	13.7
Inst-L	15.2	21.2	14.7	22.3	11.4	13.3	13.3	19.5	1.3	8.1	20.9	9.1	14.2
SBERT	15.1	20.4	16.6	22.7	8.2	11.0	15.3	26.4	7.0	8.3	20.0	10.8	14.9
<i>Open-sourced models (>1B)</i>													
ES	18.6	26.0	15.5	15.8	16.3	11.2	18.1	28.7	4.9	7.1	26.1	26.8	17.9
SPR	19.1	26.7	17.8	19.0	16.3	14.4	10.2	27.4	1.0	7.4	24.3	26.0	18.3
Inst-XL	21.6	34.3	22.4	27.4	18.2	21.2	19.1	27.5	5.0	8.5	15.6	7.9	18.9
Inst-M	25.8	32.5	18.9	19.8	22.1	15.6	17.8	29.2	22.0	8.8	25.2	21.2	21.0
Qwen	30.6	36.4	17.8	24.6	13.2	22.2	14.8	25.5	9.0	14.4	27.8	32.9	22.8
<i>Proprietary models</i>													
Cohere	18.7	28.4	20.4	21.6	16.3	18.3	17.6	26.8	1.9	6.3	15.7	7.2	16.6
OpenAI	23.3	26.7	19.5	22.6	12.8	14.3	20.8	23.6	2.4	8.5	23.5	11.7	17.9
Voyage	23.1	25.4	19.9	24.9	10.8	16.8	15.4	30.6	1.5	7.5	27.4	11.6	17.9
Google	32.7	34.8	19.6	27.8	15.7	20.1	17.1	29.6	3.6	9.2	23.8	15.9	20.0

LongEmbed: Extending Embedding Models for Long Context Retrieval. Dawei Zhu, Liang Wang, Nan Yang, Yifan Song, Wenhao Wu, Furu Wei, Sujian Li
BRIGHT: A Realistic and Challenging Benchmark for Reasoning-Intensive Retrieval. Honglin Su, Howard Yen, Mengzhou Xia, Wei Ji Shi, Niklas Muenninghoff, Hao-yu Wang, Haisu Liu, Quan Shi, Zachary S. Siegel, Michael Tang, Ruosi Sun, Jinsung

(Timestamp: 00:07:27)

This table shows the performance of various retrieval models on the BRIGHT benchmark, which is designed for reasoning-intensive tasks. The results show that even large, powerful models struggle, with the best model achieving an average nDCG@10 of only 24.3.

93

4.7 BM25's Surprising Strength

Light

If you cannot **measure it**, you cannot **improve it**

- Besides out-of-domain, different capabilities are not captured by existing benches
- Long context: old models had context window of 512, but 8k models actually do not properly support 8k
- Retrieval beyond keyword/semantic matching is challenging
- BM25 (lexical search) is surprisingly strong on those challenges

Method	Synthetic		Real				Avg.
	P	N	NQA	QMS	SFD	WQA	
BM25	100	95.3	71.5	81.3	97.6	96.5	90.4
E5-Mistral	71.0	48.3	44.6	43.6	96.8	82.0	64.4
+NTK (32k)	93.8	66.8	49.8	49.2	97.1	95.2	75.3

Performance of BM25 on the LongEmbed benchmark

	StackExchange					Coding		Theorem-based		Avg.			
	Bus.	Earth.	Econ.	Psy.	Rob.	Stack.	Sus.	Leet.	Pony		AdP	ThesQ.	TheoT.
Sparse model													
BM25	18.9	27.2	14.9	12.5	13.6	18.4	15.0	24.4	7.9	6.2	10.4	4.9	14.5
Open-sourced models (< 8B)													
BGE	11.7	24.6	16.6	17.5	11.7	10.8	13.3	26.7	5.7	6.0	13.0	6.9	13.7
Inte-L	15.1	21.2	14.7	22.3	11.4	13.3	13.5	19.5	1.3	8.1	20.9	9.1	14.2
SBERT	15.4	20.4	16.6	22.7	8.2	11.0	15.3	26.4	7.0	5.3	20.0	10.8	14.9
Open-sourced models (> 8B)													
E5	18.6	26.0	15.5	15.8	16.3	11.2	18.1	28.7	4.9	7.1	26.1	26.8	17.9
SFR	19.1	26.7	17.8	19.0	16.3	14.4	15.2	27.4	2.0	7.4	24.3	20.0	18.3
Inte-XL	21.6	34.3	22.4	27.4	18.2	21.2	15.1	27.5	5.0	8.5	15.6	5.9	18.5
Gr8LM	24.8	32.3	18.9	19.8	12.1	13.6	17.8	29.9	22.0	8.8	25.2	21.2	21.0
Quora	30.8	36.4	17.8	24.6	13.2	22.2	18.8	25.5	9.8	14.4	27.8	32.9	22.5
Proprietary models													
Cohere	18.7	28.4	20.1	21.6	16.3	18.3	17.6	26.8	1.9	6.3	15.7	7.2	16.6
OpenAI	23.1	26.7	19.5	22.6	12.8	14.3	20.8	23.6	2.4	8.5	23.5	11.7	17.9
Voyage	23.1	25.4	19.9	24.9	10.8	16.8	15.4	20.6	1.5	7.5	22.8	11.6	17.9
Google	22.7	34.8	19.6	27.8	15.7	20.1	17.1	29.6	3.6	9.3	23.8	15.9	20.0

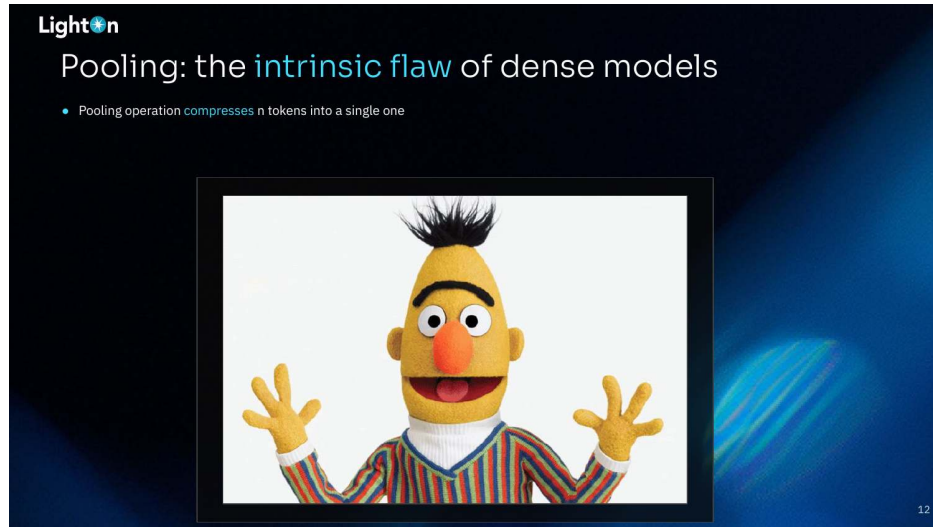
Performance of BM25 on the BRIGHT benchmark

LongEmbed: Extending Embedding Models for Long Context Retrieval. Dawei Zhu, Liang Wang, Nan Yang, Yifan Song, Wenhao Wu, Furu Wei, Sujian Li
BRIGHT: A Realistic and Challenging Benchmark for Reasoning-Intensive Retrieval. Hongjin Su, Howard Yen, Mengzhou Xia, Wei Jin Shi, Niklas Hummelshoff, Hao-yu Wang, Haisu Liu, Quan Shi, Zachary S. Siegel, Michael Tang, Ruoxi Sun, Jinsung

(Timestamp: 00:07:50)

Interestingly, BM25, a simple lexical search method that does not use deep learning, performs surprisingly well on these more challenging long-context and reasoning-intensive benchmarks. Its strength lies in its lack of compression; by matching exact keywords, it avoids the information loss that plagues dense models, making it a robust baseline for out-of-domain tasks.

4.8 The Pooling Problem



(Timestamp: 00:08:24)

Pooling is the core flaw of dense models. The process of compressing all the token vectors from a document into a single vector is inherently lossy. This compression forces the model to be selective about what information it retains.

Lightn

Pooling: the **intrinsic flaw** of dense models

- Pooling operation compresses n tokens into a single one
- Selective behavior is learned during training (mainly through data)
- Gets more extreme with longer context
- Learn one notion of similarity

Dense Models Learn Selective Information Encoding

Original Movie Review

- 👤 **Actors:** Tom Hanks, stellar performance
- 📖 **Plot:** Complex time-travel narrative
- 🎵 **Music:** Hans Zimmer's epic score
- 🎭 **Themes:** Love, sacrifice, redemption
- 🎬 **Visual Effects:** Groundbreaking CGI

Training with
actor queries

→

Encoded Representation

- ✓ **Actors:** Tom Hanks stellar performance
- ✗ **Plot:** Complex time-travel narrative
- ✗ **Music:** Hans Zimmer's epic score
- ✗ **Themes:** Love, sacrifice, redemption
- ✗ **Visual Effects:** Groundbreaking CGI

Problems in Production

User asks about plot

"What happens in the movie?"

✗ Information not encoded!

Model can't retrieve relevant docs

Cooking Recipe Input

👤 Ingredients: flour, eggs, butter

👤 Chef: Gordon Ramsay

? Model confused - no actors!

Larger Documents

More information to compress

↓ Even more selective

↓ More information lost

Reasoning-Intensive Retrieval

"Solve this mathematical problem"

Requires abstract matching

✗ Fixed similarity notion fails

(Timestamp: 00:08:41)

This slide illustrates how dense models learn selective information encoding. If a model is trained on a movie review dataset where queries are mostly about actors, it will learn to prioritize and encode information about actors while discarding details about the plot, music, or themes. This selective behavior leads to poor performance on out-of-domain queries (e.g., asking about the plot) or when applied to new domains entirely (e.g., cooking recipes).

Lightn

Pooling: the **intrinsic flaw** of dense models

- Pooling operation compresses n tokens into a single one
- Selective behavior is learned during training (mainly through data)
- Gets more extreme with longer context
- Learn one notion of similarity

BM25: Term Frequency + Inverse Document Frequency

Query: "Leonardo DiCaprio disaster"

"Titanic features **Leonardo** **DiCaprio** in a romantic **disaster** film..."

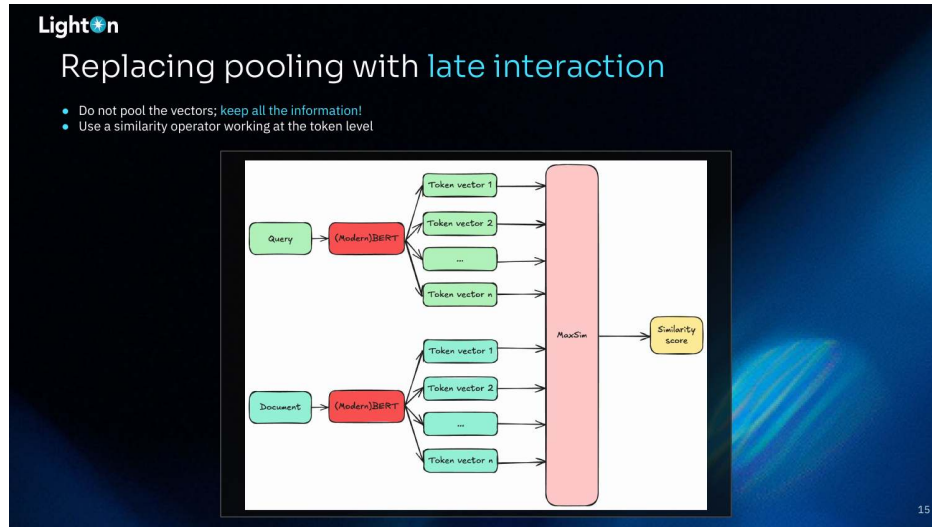
✓ No pooling/compression
✗ Exact matching

14

(Timestamp: 00:10:42)

BM25 is effective in certain cases because it avoids pooling and compression, relying on exact keyword matching. In the example, "Leonardo DiCaprio disaster" in the query directly matches the terms in the document. However, this approach fails when there's no direct lexical overlap, such as with synonyms or different languages.

4.9 Late Interaction Solution



(Timestamp: 00:11:32)

Late interaction models offer a solution by replacing the pooling step. Instead of compressing token vectors into a single one, they keep all the token-level information. A token-level similarity operator, such as MaxSim, is then used to compute the final score. MaxSim works by finding the maximum similarity between each query token and all document tokens, then summing these maximum scores.

Lightn

This is **not** just a big single vector

- If it is a compression issue, I can just use a big embedding vector right?
- Not quite, the different "views" of a document will conflict into this single representation
- MaxSim only select one document token per query token
 - Only modify the document tokens that *matched* the query
- Can fall back to "soft lexical matching" (learned BM25)

The diagram compares two models: Dense (Single Vector) and Late Interaction.

Dense (Single Vector): Shows a document snippet: "Titanic is a masterpiece featuring Leonardo DiCaprio and Kate Winslet. The romantic storyline between Jack and Rose unfolds as the ship succumbs to disaster. Their forbidden love drives the narrative." Below this, two queries are shown: "Query: 'Leonardo DiCaprio movies'" and "Query: 'romantic disaster films'". These queries lead to a "POOLING" step, resulting in a "Single Compressed Vector". A note states: "All information forced into one representation" and "⚠ Actor and plot signals interfere with each other".

Late Interaction: Shows the same document snippet. Below, the same two queries are shown. These queries lead to a "NO POOLING" step, resulting in "Token-Level Representations". A table shows the document tokens: "Titanic", "is", "masterpiece", "featuring", "Leonardo", "DiCaprio", "and", "Kate", "Winslet". Below this, a "MaxSim Operation" box lists: "• Actor query matches specific actor tokens", "• Plot query matches specific plot tokens", and "• Each aspect gets clear, uninterrupted signal".

17

(Timestamp: 00:12:27)

This slide provides a clear comparison between dense and late-interaction models. A dense model forces different concepts (e.g., actors and plot) into a single, conflicted representation. In contrast, a late-interaction model maintains separate token-level representations. The MaxSim operator can then match a query about actors to the specific actor tokens and a query about the plot to the plot tokens, resulting in clean, uninterrupted signals for each aspect of the document.

4.10 Performance Advantages

Lighten

A simple, yet effective, difference

- Strong performance out-of-domain, outperforming in-domain dense models
- Out-of-domain is tough to define, feedback from practitioners
 - Test on your own data!

Our model prior to the post-training step, JaColBERTv2.4, also outperforms all other existing approaches, even while being fully out-of-domain on every evaluation dataset.

similarity tasks. On 22 of 28 out-of-domain tests, ColBERTv2 achieves the highest quality, outperforming the next best retriever by up to 8% relative gain, while using its compressed representations.

of our XMOD-based dense retrievers. We observe that ColBERT-XM surpasses DPR-XM in the training language (i.e., English) by 4.5% on MRR@10. Furthermore, it consistently outperforms DPR-XM across the other 13 languages not encountered during training by an average of 4.9%. Supported by findings from Santhanam et al. (2022), our results confirm that multi-vector models bypass the restrictive information bottleneck inherent in single-vector models, enabling a richer and more nuanced representation of queries and passages, thereby yielding higher retrieval performance.

JaColBERTv2.5: Optimising Multi-Vector Retrievers to Create State-of-the-Art Japanese Retrievers with Constrained Resources. Benjamin Clavié
ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. Keshav Santhanam, Omar Khattab, Jon Saad-Falouci, Christopher Potts, Mufit Zaharia
ColBERTv1: A Modular Multi-Vector Representation Model for Zero-Shot Multilingual Information Retrieval. Antoine Louis, Yegor Shvachkin, Gipsy van Dijk, Gerassimos Spanakis

18

(*Timestamp: 00:13:51*)

Late-interaction models like ColBERT have demonstrated strong out-of-domain performance, even outperforming in-domain dense models. Antoine emphasizes that because “out-of-domain” is hard to define, the best approach is to test these models on your own specific data to see the benefits.

Lightn

A simple, yet **effective**, difference

- GTE-ModernColBERT is SOTA on the Long Embed by a large margin while being trained on document length of 300
 - Performance improves even from 8k -> 16k while even the base model has been trained at 8k max

Model	Mean
GTE-ModernColBERT	88.39
voyage-multilingual-2	79.17
snowflake-arctic-embed-l-v2,0	63.73
gte-multilingual-base	62.12
bge-m3	58.73

Performance (NDCG@10) of different models on the Long Embed benchmark

19

(Timestamp: 00:15:04)

The GTE-ModernColBERT model, which uses late interaction, achieves state-of-the-art results on the LongEmbed benchmark. Notably, it outperforms other models by a large margin, even though it was trained on documents with a maximum length of only 300 tokens, while the base models it's compared against were trained with an 8k context window.

Lightn

A simple, yet **effective**, difference

- GTE-ModernColBERT is SOTA on the Long Embed by a large margin while being trained on document length of 300
 - Performance improves even from 8k -> 16k while even the base model has been trained at 8k max
- Reason-ModernColBERT outperforms all 7B models (45 times bigger) on BRIGHT, competitive with ReasonIR-8B trained on the same data

Model / Metric	Full mean
BIGBench	14.53
~ 1B GDS	
DCB	13.71
Inst-L	14.21
Inst-OT	14.9
> 1B GDS	
ES	17.01
SPB	16.3
Inst-M	16.89
Inst-M	16.95
Quora	21.01
Proprietary	
Cohere	16.6
OpenAI	17.07
Wang	17.01
Google	20
ReasonIR data	
ReasonIR-8B	24.36
Reason-ModernColBERT (150M)	25.01

Performance (NDCG@10) of different models, different on the BRIGHT benchmark

20

(Timestamp: 00:15:52)

On the reasoning-intensive BRIGHT benchmark, the 150M-parameter Reason-ModernColBERT outperforms all 7B-parameter models (which are 45 times larger). It is even competitive with the proprietary ReasonIR-8B model, which was trained on the same data.



A simple, yet **effective**, difference

- GTE-ModernColBERT is SOTA on the Long Embed by a large margin while being trained on document length of 300
 - Performance improves even from 8k -> 16k while even the base model has been trained at 8k max
- Reason-ModernColBERT outperforms all 7B models (45 times bigger) on BRIGHT, competitive with ReasonIR-8B trained on the same data

Model	Mean
Dense (single vector) model	12.31
Late-interaction (multi vector model)	19.61

Comparison of single and multi vector models on the BRIGHT benchmark

21

(*Timestamp: 00:16:30*)

This slide provides a direct, apples-to-apples comparison on the BRIGHT benchmark. A late-interaction model achieves a mean score of 19.61, while a dense (single vector) model with the same backbone and training data scores only 12.31.

4.11 Interpretability Benefits



The screenshot shows the ColBERT web interface. At the top, it says "Lightn Interpretability: a nice little bonus". Below this, there are two bullet points: "Granular matching: understand why the document matched" and "Identify sub-chunk of the document that matches". The main interface displays a query: "Effects of climate change on marine ecosystems". Below the query, it shows a passage: "The changing climate has profound impacts on marine ecosystems, rising temperatures, ocean acidification, and altered precipitation patterns all contribute to shifts in the distribution and behavior of marine species, influencing the delicate balance of underwater ecosystems." The interface then shows the ColBERT scoring results: "MaxSim Score: 27.90" and "Estimated Relevance: 87.17%". At the bottom, there are "Contextualized Highlights" showing specific words and phrases from the passage that contributed to the match, such as "profound impacts", "marine ecosystems", "rising temperatures", "ocean acidification", "altered precipitation patterns", "shifts in the distribution", "behavior of marine species", and "delicate balance of underwater ecosystems". The Vespa logo is visible at the bottom right of the interface.

(Timestamp: 00:16:48)

Interpretability is a valuable bonus of late-interaction models like ColBERT. Because the MaxSim operator performs granular, token-level matching, it's possible to see exactly which parts of a document contributed to the match. This allows you to identify the specific sub-chunk of text that is most relevant, which is useful for debugging and for providing more precise context to an LLM in a RAG pipeline.

4.12 Barriers to Adoption

Light 

Why people are **still using** dense then?

- **Storing cost:** storing n tokens instead of one
 - Adapted indexes, quantization, footprint reduction techniques
- **Vectordb support:** not originally supported by vectordb providers
 - Most of them now support it (Vespa, Weaviate, QDrant, LanceDB, Milvus, Elastic)
- **Lack of accessible tools** to train, use and experiment with these models
 - Sentence transformers is a big reason why there is are many dense models!



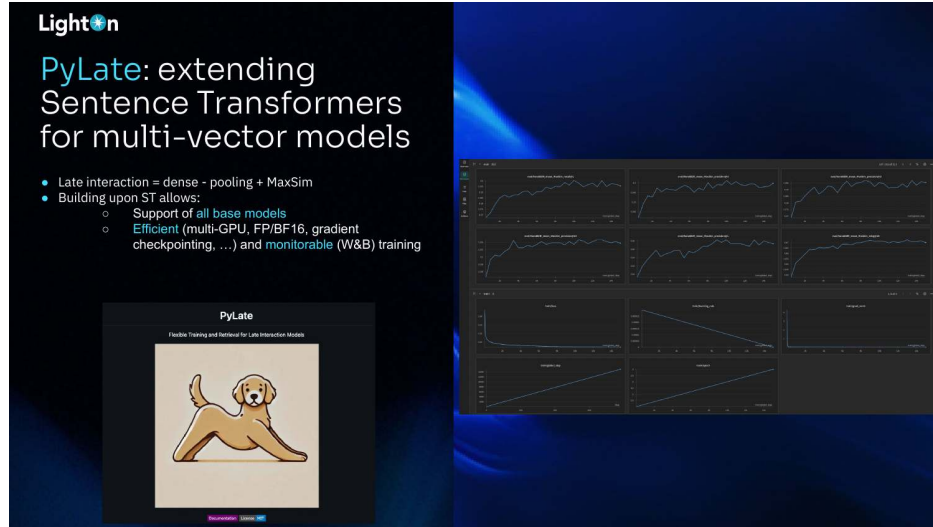
23

(*Timestamp: 00:17:42*)

Despite their advantages, dense models are still mainstream. Antoine attributes this to three main factors:

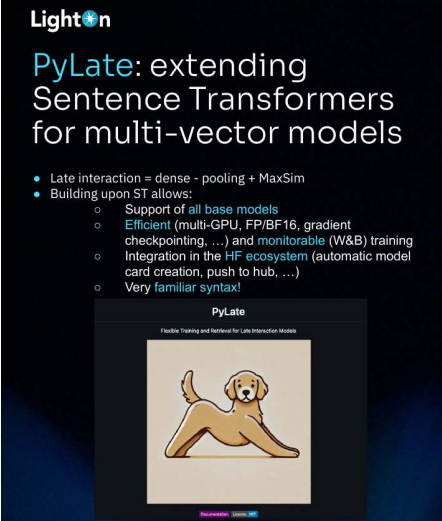
1. **Storing cost:** Storing n token vectors instead of one is more expensive, though techniques like quantization and footprint reduction are making this more manageable.
2. **VectorDB support:** Initially, most vector databases did not support the different search mechanism required by late-interaction models. However, this is changing, with major providers like Vespa, Weaviate, and LanceDB now offering support.
3. **Lack of accessible tools:** The widespread availability of libraries like Sentence Transformers made it very easy to work with dense models.

4.13 PyLate: Making Late Interaction Accessible



(*Timestamp: 00:18:43*)

To address the lack of accessible tools, Antoine and his collaborators created **PyLate**, a library that extends the popular Sentence Transformers framework for multi-vector models. Since late interaction is essentially a dense model without pooling and with a MaxSim operator, PyLate can leverage the existing Sentence Transformers ecosystem.



Lightn

PyLate: extending Sentence Transformers for multi-vector models

- Late interaction = dense + pooling + MaxSim
- Building upon ST allows:
 - Support of all **base models**
 - **Efficient** (multi-GPU, FP16/BF16, gradient checkpointing, ...) and **monitorable** (W&B) training
 - Integration in the **HF ecosystem** (automatic model card creation, push to hub, ...)
 - Very **familiar syntax**!

PyLate

Enables Training and Inference for Late Interaction Models



```

1 from datasets import load_dataset
2 from sentence_transformers import (
3     SentenceTransformerTrainer,
4     SentenceTransformerTrainingArguments,
5 )
6
7 from pylate import evaluation, losses, models, utils
8
9 # Define model architecture for contrastive training
10 model_name = "sentence-transformers-base" # Choose the pre-trained model you want to use as base
11 batch_size = 32 # Larger batch size often improves results, but requires more memory
12
13 num_epochs = 10 # Adjust based on your requirements
14 # You can use None for "None" and model strategy
15 run_name = "Contrastive-Base-Base-Base"
16 model_dir = "models/PyLate-Base"
17
18 # Here we define our custom model. If you a custom model, will add a custom layer to the base encoder.
19 model = models.CustomModel(model_name, model_dir)
20
21 # Load dataset
22 dataset = load_dataset("sentence-transformers/semco-base", "triplet", split="train")
23 # Split the dataset (this dataset has not been a validation set, so we split the training set)
24 splits = dataset.train_test_split(split_ratio=0.9)
25 train_dataset = splits["train"]
26 eval_dataset = splits["test"]
27
28 # Define the loss function
29 train_loss = losses.ContrastiveModelLoss()
30
31 # Initialize the evaluator
32 eval_evaluator = evaluation.SentenceTransformerEvaluator()
33
34 # Configure the training arguments (e.g., batch size, evaluation strategy, logging steps)
35 args = SentenceTransformerTrainingArguments(
36     output_dir=model_dir,
37     num_epochs=num_epochs,
38     per_device_train_batch_size=batch_size,
39     per_device_eval_batch_size=batch_size,
40     logging_dir=None, # Set to None if you get an error that your GPU can't run on CPU
41     bf16=True, # Set to True if you have a GPU that supports BF16
42     fp16=False, # Set to True if you have a GPU that supports FP16
43     num_workers=0, # Set to None if you want to use the default
44     logging_steps=10,
45 )
46
47 # Initialize the trainer for the contrastive training
48 trainer = SentenceTransformerTrainer(
49     model=model,
50     args=args,
51     train_dataset=train_dataset,
52     eval_dataset=eval_dataset,
53     loss_fn=train_loss,
54     data_loader_args=datasets.GpuDataLoader(args.device, args.num_workers),
55 )
56
57 # Start the training process
58 trainer.train()
                    
```

(Timestamp: 00:20:08)

The syntax for training models with PyLate is designed to be very similar to Sentence Transformers. This familiarity makes it easy for developers to adapt their existing boilerplates and workflows. The example code shows how to define a model, load a dataset, configure training arguments, and start training with just a few modifications to a standard Sentence Transformers script.



Training is cool, show me the evals

- Built-in efficient PLAID indexes to easily and efficiently perform retrieval
- Helper functions to evaluate the output using `ranx`
- Standard `queries/documents/qrels` format
 - MTEB
 - IR-datasets
 - BEIR
- In-training evaluator
- Integrated in FreshStack, soon in MTEB

```

1 """Evaluation script for BEIR datasets with a PLAID index."""
2 import argparse
3
4 from pylate import evaluation, indexes, models, retrieval
5
6 if __name__ == "__main__":
7     parser = argparse.ArgumentParser(description="PyLate eval")
8     parser.add_argument(
9         "-i", "--index_name",
10         default="C400000",
11         help="Name of the dataset to evaluate on (default: 'C400000')."
12     )
13     args = parser.parse_args()
14     dataset_name = args.index_name
15     model_name = "lightn/phi-3-medium-128k-instruct"
16     model = models.LLMModel(
17         model_name=model_name,
18         model_size=args.model_size,
19     )
20     documents, queries, qrels = evaluation.load_beir(
21         dataset_name=dataset_name, split="test"
22     )
23     index = indexes.PLAID(
24         index_name=f"lightn_{dataset_name}_{model_name}_{args.model_size}_{args.index_name}"
25     )
26     retriever = retrieval.C400000(index=index)
27
28     documents_embeddings = model.encode(
29         documents[dataset_name] for document in documents,
30         batch_size=1000,
31         in_memory=True,
32         show_progress_bar=True,
33     )
34
35     index.add_documents(
36         documents_dict[dataset_name] for document in documents,
37         documents_embeddings=documents_embeddings,
38     )
39     queries_embeddings = model.encode(
40         queries[dataset_name] for query in queries,
41         batch_size=100,
42         show_progress_bar=True,
43         in_memory=True,
44     )
45
46     scores = retriever.retrieve_queries(queries_embeddings=queries_embeddings, k=100)
47
48     evaluation_scores = evaluation.evaluate(
49         documents_embeddings=documents_embeddings,
50         queries_embeddings=queries_embeddings,
51         qrels=qrels,
52         metrics=["map", "recall@10", "recall@20", "recall@50", "recall@100"],
53     )
54     print(evaluation_scores)

```

(Timestamp: 00:21:28)

PyLate is not just for training; it also provides tools for evaluation. It includes a built-in, efficient index based on PLAID for fast retrieval. It also has helper functions that use the `ranx` library to easily compute standard IR metrics (like NDCG and Recall) on the retrieval output.

4.14 Future Research Directions

Lightn

What are the future **avenues**?

- Reduce the **cost** of multi-vector models, find the optimal trade-off

	Index Size
16-bit Dense Vector	345 MB
2-bit PLAID Index	760 MB
pool 2	388 MB
pool 3	260 MB
pool 4	195 MB
pool 6	131 MB

Table 3: An overview of index size (in Megabytes) between single-vector representation and PLAID ColBERT indexes.

	cfever	dbpedia	fever	figa	hotpot	nf	nq	scidocs	scifact	covid	touche	Avg.
nDCG@10												
ColBERTv2	0.163	0.434	0.751	0.338	0.679	0.329	0.554	0.154	0.638	0.705	0.261	0.455
Hierarch. Pooling₂	0.170	0.438	0.744	0.331	0.657	0.327	0.553	0.153	0.629	0.719	0.294	0.456
Hierarch. Pooling ₃	0.168	0.414	0.740	0.311	0.631	0.310	0.526	0.152	0.614	0.730	0.321	0.447
Hierarch. Pooling ₄	0.171	0.407	0.738	0.285	0.604	0.310	0.503	0.148	0.578	0.734	0.333	0.437
Hierarch. Pooling₆	0.171	0.374	0.731	0.258	0.544	0.293	0.468	0.144	0.574	0.707	0.348	0.419

Reducing the Footprint of Multi-Vector Retrieval with Minimal Performance Impact via Token Pooling. Benjamin Clavié, Antoine Chaffin, Griffin Adams

(Timestamp: 00:22:49)

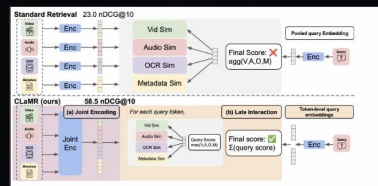
One future research avenue is to reduce the storage cost of multi-vector models. Techniques like hierarchical pooling and quantization are being explored to find the optimal trade-off between index size and performance.

Light

What are the future avenues?

- Reduce the cost of multi-vector models, find the optimal trade-off
- Apply it to other modalities

This dual approach provides greater effectiveness with multi-vector representations, particularly in visually rich document retrieval, while maintaining efficiency for standard similarity tasks. The consistent 7-10% performance advantage of multi-vector over single-vector mode across visual tasks suggests that late interaction provides fundamentally better semantic matching for multimodal content.



ColPali: Efficient Document Retrieval with Vision-Language Models. Manuel Faysse, Hugues Sibille, Tony Wu, Bilal Omrani, Gautier Viaud, Céline Hudelot, Pierre Colombo
 Sim-embeddings-v4: Universal Embeddings for Multimodal Multilingual Retrieval. Michael Gelfond, Saba Shams, Mohammad Kalin Akram, Isabelle Helly, Andrei Ungureanu, Bo Wang, Sedigheh Eslami, Scott Martens, Maximilian Werk, Nan Wang, Han Xiao
 CLaMR: Contextualized Late-Interaction for Multimodal Content Retrieval. David Wan, Han Wang, Elias Stengel-Esklin, Jaemin Cho, Mohit Bansal

(Timestamp: 00:23:34)

Another promising direction is applying late interaction to other modalities beyond text. Approaches like ColPali have already used ColBERT for OCR-free RAG with text and images. The diagram shows the CLaMR model, which uses late interaction for multimodal content retrieval across video, audio, OCR, and metadata.



What are the future avenues?

- Reduce the cost of multi-vector models, find the optimal trade-off
- Apply it to other modalities
- Find better similarity functions

Efficient Document Ranking with Learnable Late Interactions

Ziwei Ji Himanshu Jain Andreas Veit Sashank J. Reddi
Sadeep Jayasumana Ankit Singh Rawat Aditya Krishna Menon Felix Yu
Sanjiv Kumar

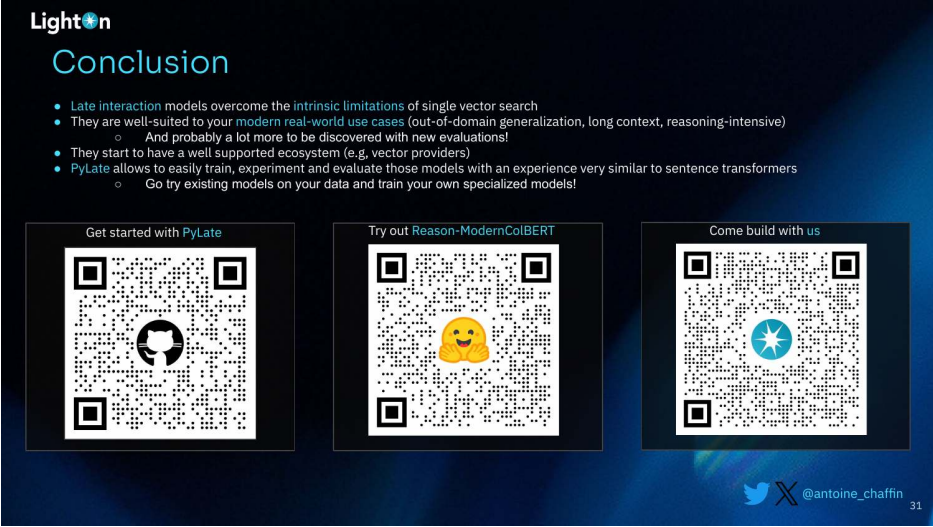
Google
{ziweiiji,himj,aveit,sashank,sadeep,ankitsrawat,
adityakmenon,felixyu,sanjivk}@google.com

30

(*Timestamp: 00:24:21*)

The final future avenue is to develop better similarity functions. While the MaxSim operator is effective and has nice properties, it is relatively naive. Research into learnable late interaction functions presents an opportunity to further improve the performance of these models.

4.15 Chapter Summary



The slide is titled "LightOn Conclusion". It features a bulleted list of points about late interaction models. Below the list are three QR codes with labels: "Get started with PyLate" (with a GitHub icon), "Try out Reason-ModernColBERT" (with a thinking face emoji), and "Come build with us" (with a star icon). The bottom right corner has a Twitter logo and the handle "@antoine_chaffin" with the number "31".

LightOn
Conclusion

- Late interaction models overcome the **intrinsic limitations** of single vector search
- They are well-suited to your **modern real-world use cases** (out-of-domain generalization, long context, reasoning-intensive)
 - And probably a lot more to be discovered with new evaluations!
- They start to have a well supported ecosystem (e.g. vector providers)
- PyLate allows to easily train, experiment and evaluate those models with an experience very similar to sentence transformers
 - Go try existing models on your data and train your own specialized models!

Get started with PyLate

Try out Reason-ModernColBERT

Come build with us

@antoine_chaffin 31

(*Timestamp: 00:24:36*)

Key takeaways from late interaction models:

1. **Overcome limitations:** Late interaction models address the fundamental information loss problem of single-vector search
2. **Modern use cases:** They excel at out-of-domain, long context, and reasoning-intensive retrieval tasks
3. **Growing ecosystem:** Tools like PyLate and vector database support make them increasingly accessible
4. **Performance gains:** Consistent improvements over dense models across challenging benchmarks
5. **Interpretability:** Token-level matching provides valuable debugging and explanation capabilities

The technology is mature enough for production use, and the performance benefits often outweigh the minor increases in storage and

computational costs.

4.16 Video

Here is the full video:

<https://youtu.be/1x3k0V2IITo>

5 RAG with Multiple Representations

Based on a presentation by Bryan Bischof and Ayush Chaurasia

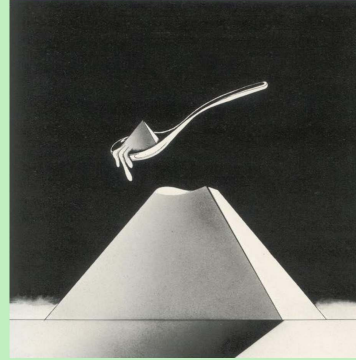
[Bryan Bischof](#) and [Ayush Chaurasia](#) argued that effective retrieval lies not in finding a single, perfect data representation, but in creating and leveraging multiple, diverse representations with a router to better serve user intent.

5.1 The Map is Not the Territory

The Map Is Not The Territory

(And That's Good)

Dr. Bryan Bischof – Ayush Chaurasia



11/07/2025

(Timestamp: 01:00)

The presentation's central theme is that a “map” (a data representation) is not the same as the “territory” (the real-world data). In machine learning, this distinction is an advantage. Models and embeddings are our maps, and we can create many different maps of the same territory to serve different purposes.

5.2 Deconstructing RAG Buzzwords

The RAG landscape is filled with terms that can obscure fundamental principles. These terms can be deconstructed into simpler concepts.

Part I

“_____ RAG”

+ *Naive RAG*

(*Timestamp: 02:37*)

Naive RAG is better understood as **Simple RAG**: the foundational approach of searching a vector store with a vector to find similar items.

Part I

“_____ RAG”

- + *Naive RAG*
- + *Agentic RAG*

(*Timestamp: 04:04*)

Agentic RAG involves an LLM choosing *how* to search, giving the impression that the model can remove the engineer from designing the retrieval pipeline.

Part I

“_____ RAG”

- + *Naive RAG*
- + *Agentic RAG*
- + *Hybrid RAG*

(*Timestamp: 04:41*)

Hybrid RAG combines Simple RAG with classic retrieval techniques like keyword matching, allowing for searches with multiple signals simultaneously.

Part I

“_____ RAG”

- + *Naive RAG*
- + *Agentic RAG*
- + *Hybrid RAG*
- + *Graph RAG*

(*Timestamp: 05:05*)

Graph RAG uses the relationships *between* objects to improve retrieval, such as identifying stores that sell “home goods” to find a coffee filter, also considering proximity.

Part I

“_____ RAG”

- + *Naive RAG*
- + *Agentic RAG*
- + *Hybrid RAG*
- + *Graph RAG*
- + *Multi-Modal RAG*

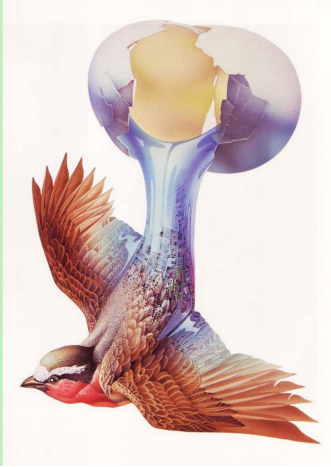
(*Timestamp: 06:06*)

Multi-Modal RAG has two meanings: searching with multiple data types (text and images) or searching across multiple locations (“modes”) within the same latent space for a single item.

5.3 A First-Principles View of RAG

Advanced RAG techniques can be reframed as core engineering pipelines.

Part II



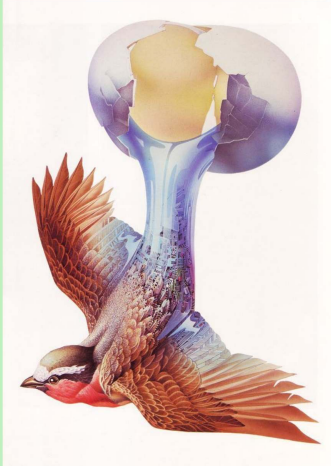
ONE

Hypothetical Document Embeddings with semantic search is a document enrichment pipeline, not a retrieval technique.

(*Timestamp: 07:58*)

Hypothetical Document Embeddings (HyDE) is a **document enrichment pipeline**. It uses an LLM to rewrite documents into the language that users are likely to search with. A dense, technical document can be rewritten into a simpler description, creating a new, more searchable “map” of the original.

Part II



ONE

Hypothetical Document Embeddings with semantic search is a document enrichment pipeline, not a retrieval technique.

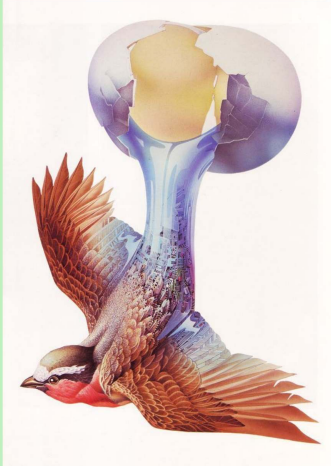
TWO

Agentic RAG is a query enrichment pipeline, not a retrieval technique.

(*Timestamp: 10:14*)

Agentic RAG is a **query enrichment pipeline**. When a query is ambiguous, an agent decides how to search. For example, it determines whether “V60 filter” refers to a product or a restaurant, routing the query to the correct search process.

Part II



ONE

Hypothetical Document Embeddings with semantic search is a document enrichment pipeline, not a retrieval technique.

TWO

Agentic RAG is a query enrichment pipeline, not a retrieval technique.

THREE

Rank fusion (knn) is multi-stage processing, not a retrieval technique.

(*Timestamp: 11:38*)

Rank fusion is **multi-stage processing**. It involves running multiple different searches and then combining, or “stitching,” the results together in a subsequent stage.

5.4 The Three Responsibilities of an IR Engineer

These advanced techniques can be derived by focusing on three core responsibilities.

Part II

— You Could Have Invented —

The core responsibility of the IR engineer is to understand:

- What is the most likely representation of the user's desire?

(Timestamp: 12:43)

1. **Predict user intent:** What is the most likely representation of what the user is looking for?

Part II

— You Could Have Invented —

The core responsibility of the IR engineer is to understand:

- What is the most likely representation of the user's desire?
- What representations can you generate ahead of time from the entities?

(*Timestamp: 13:08*)

2. **Generate multiple representations (maps):** Create different views of the source data ahead of time (document enrichment).

Part II

— You Could Have Invented —

The core responsibility of the IR engineer is to understand:

- What is the most likely representation of the user's desire?
- What representations can you generate ahead of time from the entities?
- How can you correctly match these up?

(*Timestamp: 13:39*)

3. **Match intent to representation:** Correctly match the user's query with the appropriate pre-generated representation.

5.5 Practical Application: Curving Space

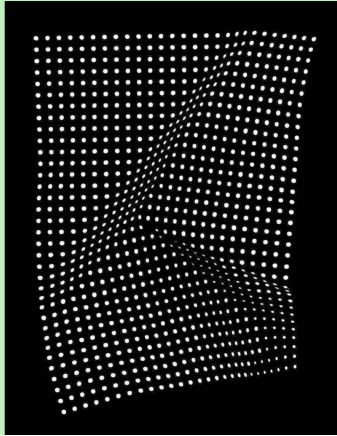
“Curving space” means shaping data representations and indices to improve search.

Part III

Let's retrieve Financial Documents

ONE

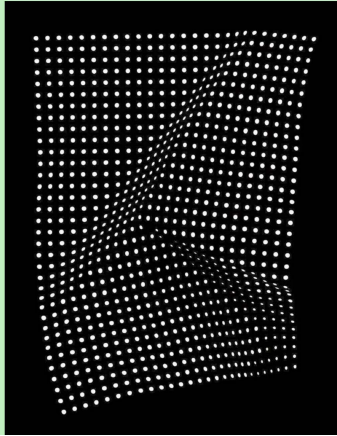
What are the different concepts one may construct from these docs? Summaries?
Tables of Data? Lists of Named Entities?
Form types/categories? What else?



(Timestamp: 13:58)

For financial documents, one could create multiple “maps” from a single corpus, such as summaries, tables of data, lists of named entities, and form types. This is document enrichment.

Part III



Let's retrieve Financial Documents

ONE

What are the different concepts one may construct from these docs? Summaries? Tables of Data? Lists of Named Entities? Form types/categories? What else?

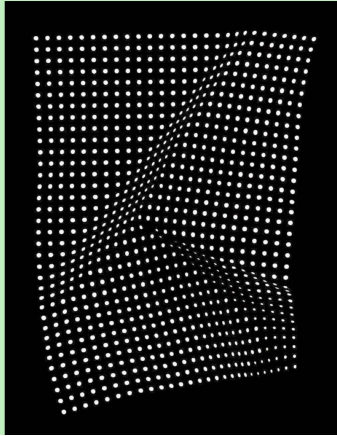
TWO

What is the appropriate index strategy for each? Semantic Search? BM25? Keyword Matching? Pre-filters? When should you use multiple on one index? When not?

(*Timestamp: 16:34*)

Once multiple representations exist, the right indexing strategy must be chosen for each. This involves deciding between semantic search, keyword matching, pre-filters, and whether to use a single index or separate ones.

Part III



Let's retrieve Financial Documents

ONE

What are the different concepts one may construct from these docs? Summaries? Tables of Data? Lists of Named Entities? Form types/categories? What else?

TWO

What is the appropriate index strategy for each? Semantic Search? BM25? Keyword Matching? Pre-filters? When should you use multiple on one index? When not?

THREE

When is another "stage" required? Is the easiest thing to retrieve what the agent needs? Can one retrieval enable another?

(*Timestamp: 18:08*)

Sometimes, a second retrieval step, informed by the first, is required. This is a form of staged processing.

5.6 Agents as Routers



(*Timestamp: 18:54*)

Agents are **transformers** in their function: they transform incoming data and instructions into a structured output.

Part IV

Agents Are Transformers

(no, not like that)

In both Agentic RAG and Multi-Agent RAG, the core idea is to use general LM's to transform incoming data and instructions into a structured output.

Thinking of them like 'routers' can be very helpful: route the data to different indices based on instructions, or route upstream stages to more retrieval stages.



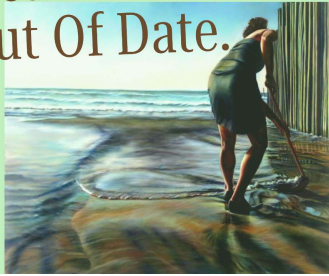
(*Timestamp: 19:04*)

An effective way to use agents is as **routers**. They take incoming data and route it to different indices or subsequent retrieval stages. This is the core idea behind both Agentic RAG and Multi-Agent RAG.

5.7 Dynamic Representations

Part V

The Danger In All Of This, Is That Many Documents Are Dynamic. This Means That The Representations Get Out Of Date.

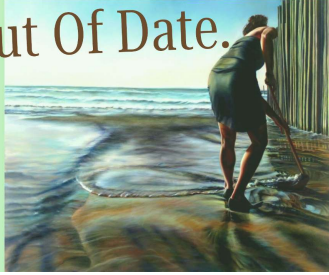


(Timestamp: 20:51)

A significant danger in RAG is that documents are dynamic. A static embedding will become outdated as the context of a document changes based on new business or world events.

Part V

The Danger In All Of This, Is That Many Documents Are Dynamic. This Means That The Representations Get Out Of Date.



The Upshot Here Is That You Need To Design The System To Shard Each Entity In The Easiest Way Possible To Understand What's Changed, And Then Re-Index That Which Has.

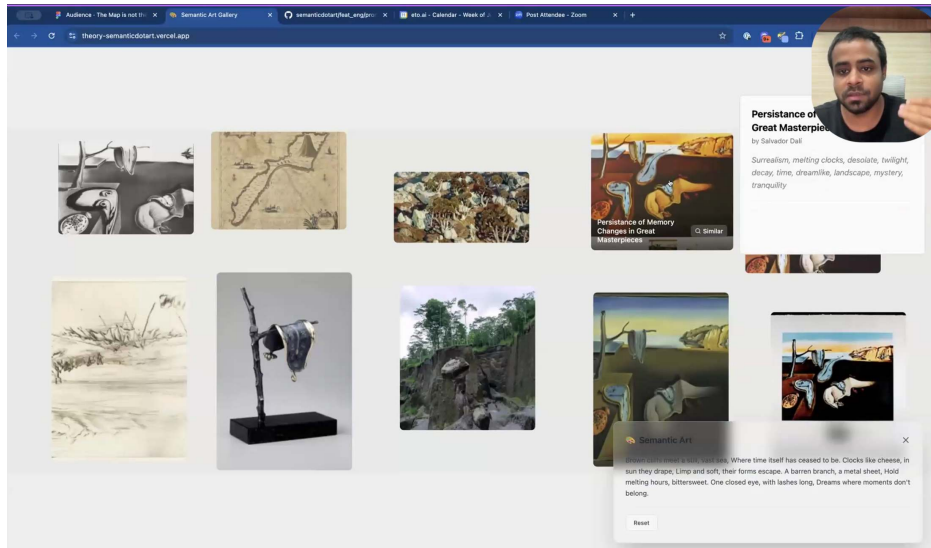
(*Timestamp: 22:49*)

The solution is to design a system that can detect what has changed and re-index only those parts, requiring an architecture built for dynamic updates.

5.8 Demo: Semantic Dot Art

(*Timestamp: 23:21*)

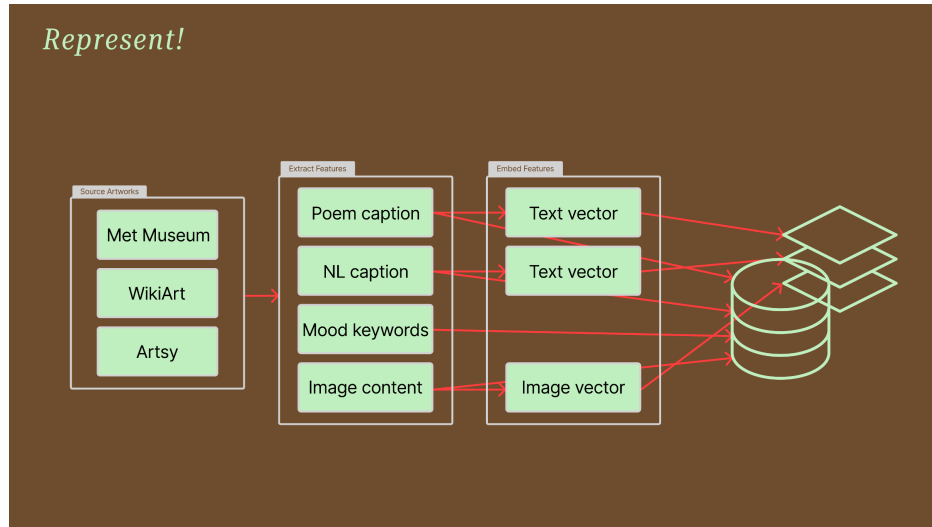
To make these concepts concrete, Ayush Chaurasia demonstrated **semanticdotart**, an application for discovering artworks. The demo shows how a system can serve diverse user intents by creating and searching over multiple representations of the same underlying data.



A user can search for art using different “maps” of the data. For example, they can search with a literal description (“multiple clocks melting in a desert”), a poetic description, or even a similar image. The system retrieves not only the original artwork but also derivative pieces and other thematically related works. This is made possible by a rich document enrichment pipeline that creates multiple vector and keyword-based indices for each artwork, capturing everything from mood and style to literal object descriptions.

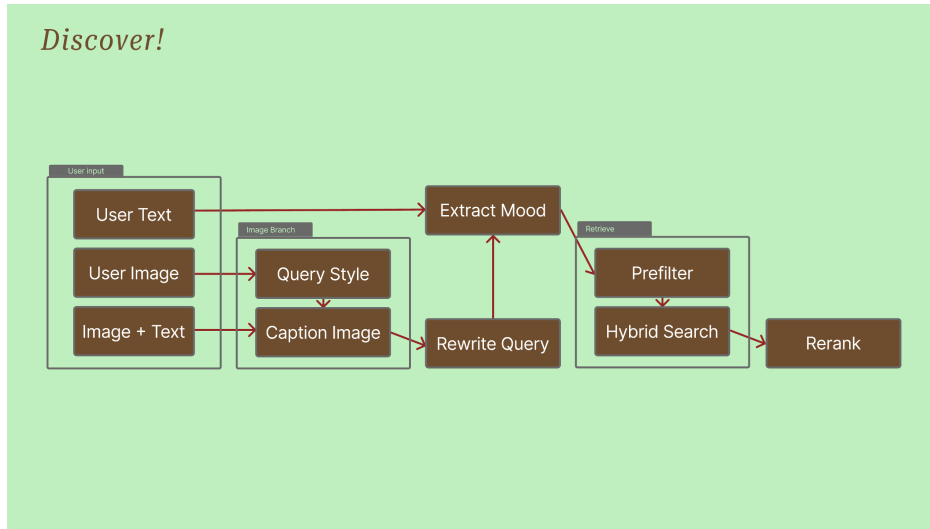
The retrieval process is agentic. The system routes the user’s query to the most appropriate index or combination of indices. A poetic query might be routed to an index of artistic descriptions, while an image query would use a multimodal embedding. This dynamic routing, combined with multi-stage processing and rank fusion, allows the system to handle a wide variety of user needs and deliver more satisfying, diverse results.

5.9 System Architecture



(*Timestamp: 27:26*)

The “Represent!” diagram visualizes the document enrichment pipeline. Data from various sources is processed to extract different features (poem captions, NL captions, mood keywords, image content). These are embedded into different vector types and stored, creating multiple “maps” for the same territory.



(*Timestamp: 29:10*)

The “Discover!” diagram shows the retrieval pipeline. User input is routed through various stages where it can be enriched (e.g., mood extraction, query rewriting). These enriched queries are then used in a multi-stage retrieval process involving pre-filtering and hybrid search before final reranking.

5.10 Integration with Other Techniques

The multiple representations approach naturally integrates with other advanced RAG techniques:

Reasoning Models are a form of query enrichment, rewriting confusing queries with necessary context.

Late-interaction models like ColBERT create diversity *within the model itself* by generating multiple representations (a vector for each

token) for a single document. This creates different “modes” in the latent space for more diverse and nuanced search results.

Routing is a retrieval stage where you decide which process or index to use based on the query. For a small number of routes, an LLM can act as a simple classifier. As the number of routes grows, a dedicated, trained classifier becomes more appropriate.

5.11 Chapter Principles

The key insights from this approach:

1. **No Single Perfect Map:** Instead of seeking one perfect representation, build multiple specialized representations of your data
2. **Intent-Based Routing:** Use agents or classifiers to route queries to the most appropriate representation
3. **Dynamic Updates:** Design systems that can detect changes and re-index incrementally
4. **Multi-Stage Processing:** Combine results from different representations and processing stages
5. **Domain Specialization:** Create representations tailored to specific types of queries or user intents

This framework provides a principled way to think about advanced RAG techniques as variations on the theme of multiple representations and intelligent routing.

5.12 Video

Here is the full video:

<https://youtu.be/hf9B3knU9vc>

6 Context Rot: When Long Context Windows Fail

Based on a presentation by Kelly Hong

[Kelly Hong](#) is a researcher at Chroma who investigates how LLMs handle long-context inputs. Despite marketing claims that models with 1M+ token context windows make RAG obsolete, Kelly’s [research](#) reveals that performance degrades significantly as context length increases—a phenomenon she coined “Context Rot.” Her experiments across 18 state-of-the-art models show that simply having information in context isn’t enough; how that information is presented matters critically for reliable AI applications.

6.1 Introduction

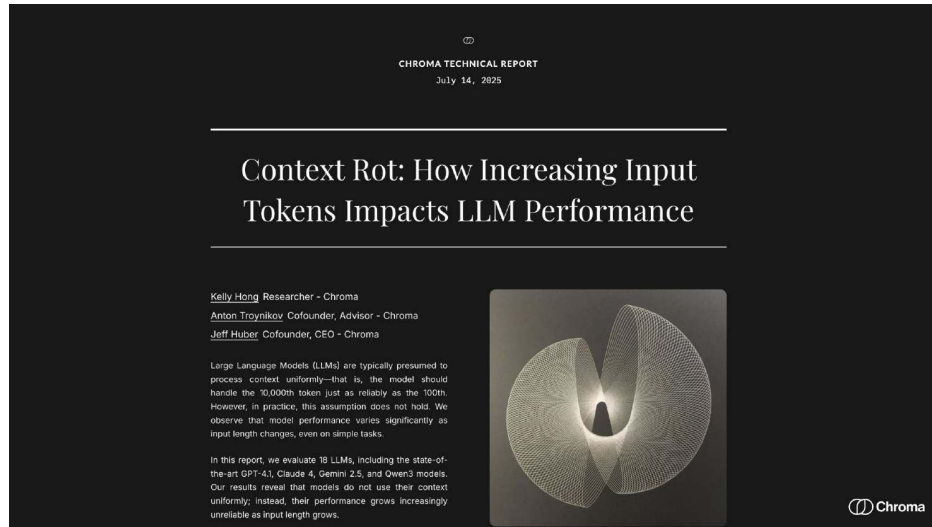


Figure 6.1: Title Slide

(*Timestamp: 00:00:00*)

This slide introduces the concept of “Context Rot,” a term coined by Chroma to describe how an LLM’s performance becomes increasingly unreliable as the length of its input context grows. The research evaluates 18 state-of-the-art LLMs and finds that, contrary to the assumption of uniform context processing, performance degrades significantly with longer inputs.

6.2 The Long Context Promise

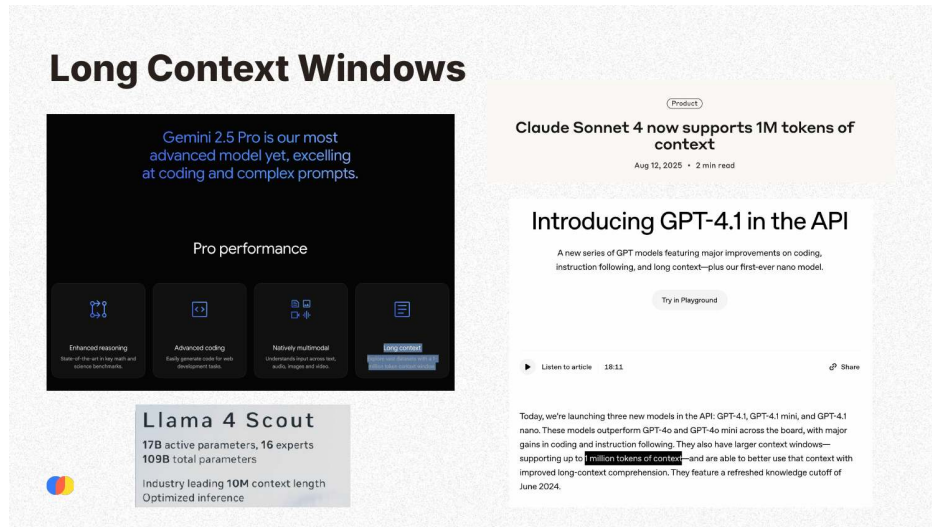


Figure 6.2: The Rise of Long Context Windows

(*Timestamp: 01:47*)

Major LLM providers prominently advertise massive context windows—often 1 million tokens or more—as a key feature of their frontier models like Gemini, Claude, and GPT-4.1. This marketing suggests that models can effectively process and utilize vast amounts of information.

Common Assumption

Maintain performance across all input lengths → giving model more context will only be beneficial (?)

Claude Sonnet 4 now supports up to 1 million tokens of context on the Anthropic API—a 5x increase that lets you process entire codebases with over 75,000 lines of code or dozens of research papers in a single request.

Long context support for Sonnet 4 is now in public beta on the Anthropic API, Amazon Bedrock, and Google Cloud's Vertex AI.

Longer context, more use cases

With longer context, developers can run more comprehensive and data-intensive use cases with Claude, including:

- **Large-scale code analysis:** Load entire codebases including source files, tests, and documentation. Claude can understand project architecture, identify cross-file dependencies, and suggest improvements that account for the complete system design.
- **Document synthesis:** Process extensive document sets like legal contracts, research papers, or technical specifications. Analyze relationships across hundreds of documents while maintaining full context.
- **Context-aware agents:** Build agents that maintain context across hundreds of tool calls and multi-step workflows. Include complete API documentation, tool definitions, and interaction histories without losing coherence.



GPT-4.1, GPT-4.1 mini, and GPT-4.1 nano needle in a haystack accuracy

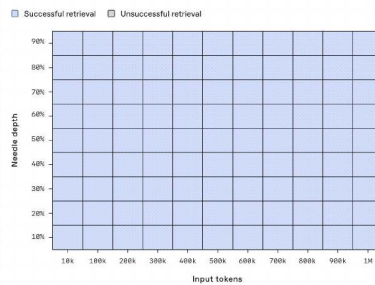


Figure 6.3: The Common Assumption: More Context is Better

(*Timestamp: 02:07*)

The availability of large context windows has led to the common assumption that providing more context is always beneficial. This has inspired new use cases, such as large-scale code analysis and extensive document synthesis. Benchmarks like the “needle in a haystack” test, which often show near-perfect retrieval accuracy across the entire context window, appear to reinforce this assumption, creating a potentially misleading picture of model capabilities.

6.3 Limitations of Needle in a Haystack

Needle in a Haystack (NIAH)

Task:

- Place random fact (needle) in a long context window (haystack)
- Ask model to retrieve the fact

Primarily assesses direct lexical matching

However, real tasks are more complex and semantically oriented

[Question]
"What was the best writing advice I got from my college classmate?"

[Needle]
The best writing advice I got from my college classmate was to write every week.

[Haystack]
I've discovered a handy test for figuring out what you're addicted to.
Imagine you were going to spend the weekend at a friend's house on a little island off the coast of Maine. There are no shops on the island and you won't be able to leave while you're there. Also, you've never been to this house before, so you can't assume it will have more than any house might.
What, besides clothes and toiletries, do you make a point of packing? That's what you're addicted to.
For example, if you find yourself packing a bottle of vodka (just in case), you may want to stop and think about that. For me the list is four things: books, earplugs, a notebook, and a pen. There are other things I might bring if I thought of it, like music, or tea, but I can live without them.
I'm not so addicted to caffeine that I wouldn't risk the house not having any tea, just for a weekend.

Figure 6.4: Explaining the “Needle in a Haystack” (NIAH) Benchmark

(*Timestamp: 03:33*)

The Needle in a Haystack (NIAH) test is a simple retrieval task where a specific fact (the “needle”) is placed within a long document (the “haystack”), and the model is asked to retrieve it. Kelly explains that this benchmark primarily assesses direct **lexical matching**. As seen in the example, the query and the needle share many of the same words (“best writing advice,” “college classmate”). This makes the task relatively easy and not representative of real-world scenarios, which often require more complex **semantic** understanding where direct word overlap is minimal.

6.4 Experiment 1: Semantic vs. Lexical Matching

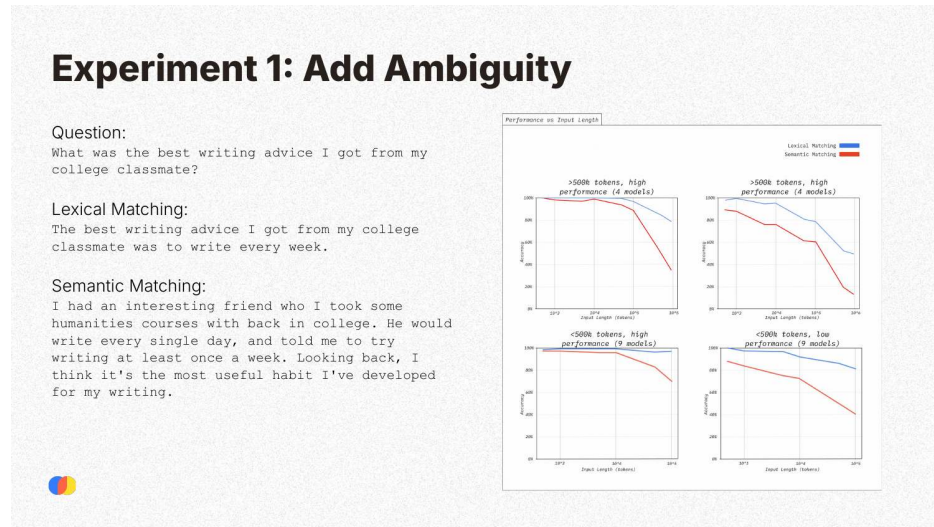


Figure 6.5: Experiment 1: Adding Ambiguity (Semantic vs. Lexical Matching)

(Timestamp: 04:49)

To test performance on more realistic tasks, Chroma's first experiment introduced ambiguity. They compared a **lexical matching** task (similar to the original NIAH) with a **semantic matching** task, where the answer contained the same core information but was phrased differently, requiring the model to understand meaning beyond direct word overlap. The results show a clear trend: while performance on lexical matching remains relatively high, performance on the more complex semantic matching task degrades significantly as the input context grows longer.

Experiment 1: Implications

Real applications of these models will often use ambiguous queries that demand processing beyond lexical matching

Example: financial report analysis

Document snippet:

Germany contributed \$45M (23% of international revenue), Japan contributed \$38M (19%), and the United Kingdom contributed \$32M (16%). Combined international markets totaled 67% of revenue in fiscal 2024, up from 43% in fiscal 2023.

Lexical Matching:

How much did Germany, Japan, and the United Kingdom contribute to our revenue in fiscal 2024?

Semantic Matching:

How is our overseas expansion going?



Figure 6.6: Implications of Ambiguity in Real-World Applications

(*Timestamp: 08:08*)

This slide illustrates the real-world implications of the previous experiment using a financial report analysis example. A user is unlikely to know the exact phrasing in a document to formulate a perfect lexical query. Instead, they will ask a more ambiguous, semantic question like “How is our overseas expansion going?” This requires the model to connect “overseas expansion” to specific countries and revenue figures. As Experiment 1 showed, this is precisely the kind of task where performance degrades with longer contexts.

6.5 Experiment 2: The Impact of Distractors

Experiment 2: Add Distractors

Question:
What was the best writing advice I got from my college classmate?

Needle:
I think the best writing tip I received from my college classmate was to write every week.

Distractors:

- The best writing tip I received from my college professor was to write everyday.
- The worst writing advice I got from my college classmate was to write each essay in five different styles.
- The best writing advice I got from my classmate was to write each essay in three different styles, this was back in high school.
- I thought the best writing advice I got from my college classmate was to write each essay in four different styles, but not anymore.

[Question]
"What was the best writing advice I got from my college classmate?"

[Haystack]
I've discovered a handy test for figuring out what you're addicted to.
Imagine you were going to spend the weekend at a friend's house on a little island off the coast of Maine. There are no shops on the island and you won't be able to leave while you're there. Also, you've never been to this house before, so you can't assume it will have more than any house might.

[Irrelevant Content]

[Needle]
The best writing advice I got from my college classmate was to write every week.

[Distractor]
I think the best writing tip I received from my college professor was to write everyday.
For example, if you find yourself packing a bottle of vodka (just in case), you may want to stop and think about that. For me the list is four things: books, earplugs, a notebook, and a pen. There are other things I might bring if I thought of it, like music, or tea, but I can live without them.

Figure 6.7: Experiment 2: Adding Distractors

(*Timestamp: 09:39*)

The second experiment investigates how performance is affected by **distractors**—pieces of information that are semantically similar to the correct answer but are incorrect. In the example, the correct “needle” is writing advice from a “college classmate.” The distractors include similar advice from a “college professor” or advice about writing essays in different styles. These distractors mimic the kind of noise often found in real-world documents.



Figure 6.8: Visualizing the Distractor Setup

This slide provides a simple visual model of the experiment. The researchers tested the LLM's performance under three conditions: with no distractors, with one distractor, and with four distractors placed in the context alongside the correct needle.

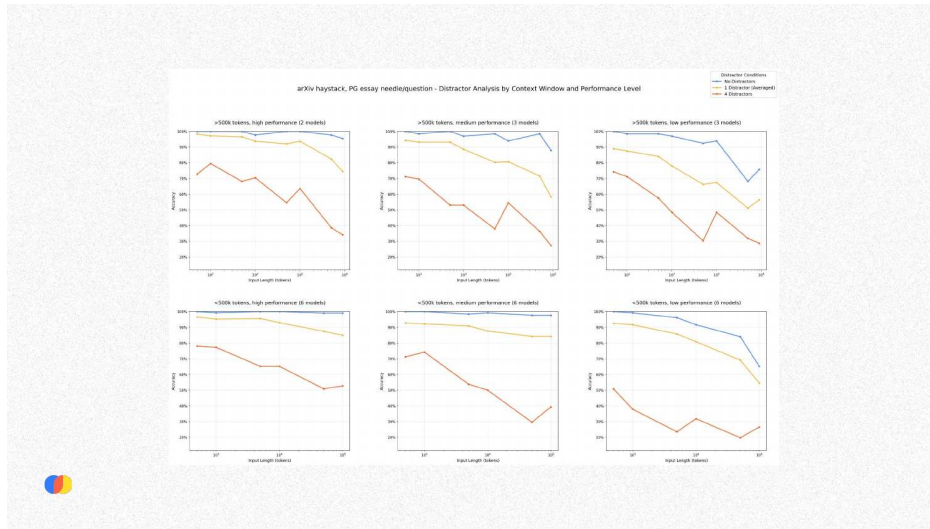


Figure 6.9: Results: Performance Degrades with More Distractors

(*Timestamp: 11:00*)

The results of the distractor experiment show two clear trends. First, across all model groups, performance degrades as the input length increases. Second, performance also degrades as the number of distractors increases. The combination of long context and distracting information proves particularly challenging for LLMs, causing a significant drop in accuracy.

Experiment 2: Implications

Common to have similar information in context, especially for domain-specific applications

Example:

Question:

How much did Germany contribute to revenue in fiscal 2024?

Answer:

Germany contributed \$45M (23% of international revenue) in fiscal 2024

Distractors:

- Germany contributed \$35M (17% of international revenue) in fiscal 2023
- France contributed \$38M (19% international revenue) in fiscal 2024



Figure 6.10: Implications of Distractors in Domain-Specific Contexts

(*Timestamp: 11:43*)

This experiment is highly relevant to real-world applications, especially in domain-specific contexts like finance or law. Documents in these fields often contain highly similar, templated information where only small details (like a year or a name) differ. These similar pieces of information act as natural distractors, making it difficult for the model to retrieve the correct fact, a problem that is exacerbated by longer contexts.

Model Hallucinations

Analyzed failed attempts in 4-distractor condition:

Abstention: "I don't know"

Distractor x: "The best writing advice I got from my college classmate was (distractor x)"

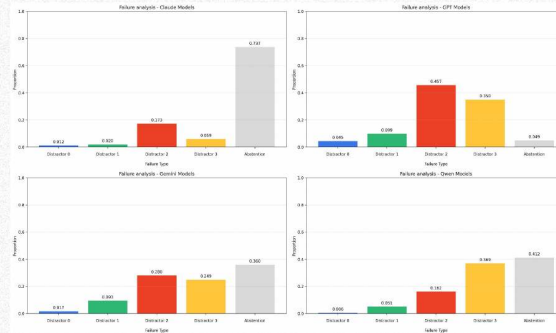


Figure 6.11: Analyzing Failure Modes: Model Hallucinations vs. Abstention

(*Timestamp: 12:55*)

When the models failed in the 4-distractor condition, the researchers analyzed *how* they failed. A key finding was that models often **hallucinate** by confidently providing an answer based on one of the distractors, rather than **abstaining** (stating "I don't know"). This tendency varies by model family: Claude models are more likely to abstain when uncertain, whereas GPT models have the highest rate of hallucination, confidently returning an incorrect answer.

6.6 Experiment 3: Context Structure Matters

Experiment 3: Shuffle Haystack Content

A random needle within a coherent essay disrupts the logical flow of ideas → needle stands out

A needle within other randomly ordered sentences → needle blends in

[Original]

I've discovered a handy test for figuring out what you're addicted to: imagine you were going to spend the weekend at a friend's house on a little island off the coast of Maine. There are no shops on the island and you won't be able to leave while you're there. Also, you've never been to this house before, so you can't assume it will have more than very basic things.

[Needle] The best writing advice I got from my college classmate was to write every week.

Which, looking back, seems like you should make a point of packing? That's what you're addicted to.

For example, if you find yourself packing a bottle of vodka (just in case), you may want to stop and think about that. For me the list is long: things, books, earplugs, a notepad, and a pen. There are other things I might bring if I thought of it, like music, or tea, but I can live without them.

I'm not so addicted to caffeine that I wouldn't risk the house not having any tea, just for a weekend.

[Shuffled]

I've discovered a handy test for figuring out what you're addicted to: imagine you were going to spend the weekend at a friend's house on a little island off the coast of Maine. > imagination

Intelligence does matter a lot of course.

I thought studying philosophy would be a natural strength in wisdom.

[Needle] The best writing advice I got from my college classmate was to write every week.

All the people missing in other things would just end up with a bunch of random knowledge.

"Good" and "bad" technological progress aren't always differentiated, so you'd find you couldn't know the latter without also knowing the former. Please talk so much about abstractions now that we don't realize what a step it must have been when they first started to.

They're doing a finance startup, which means endless negotiations with big, bureaucratic companies.

[Addictive]

[Founders]

[Philosophy]

Figure 6.12: Experiment 3: Shuffling Haystack Content

(Timestamp: 14:12)

This experiment tested whether models process context in a structured, order-sensitive manner. A “needle” (a sentence about writing advice) was placed in a coherent essay. Because the needle disrupts the essay’s logical flow, it stands out. The same needle was also placed in a “haystack” of randomly shuffled, unrelated sentences, where it should logically blend in more. The hypothesis was that the model would find it easier to retrieve the needle from the coherent essay where it was an anomaly.

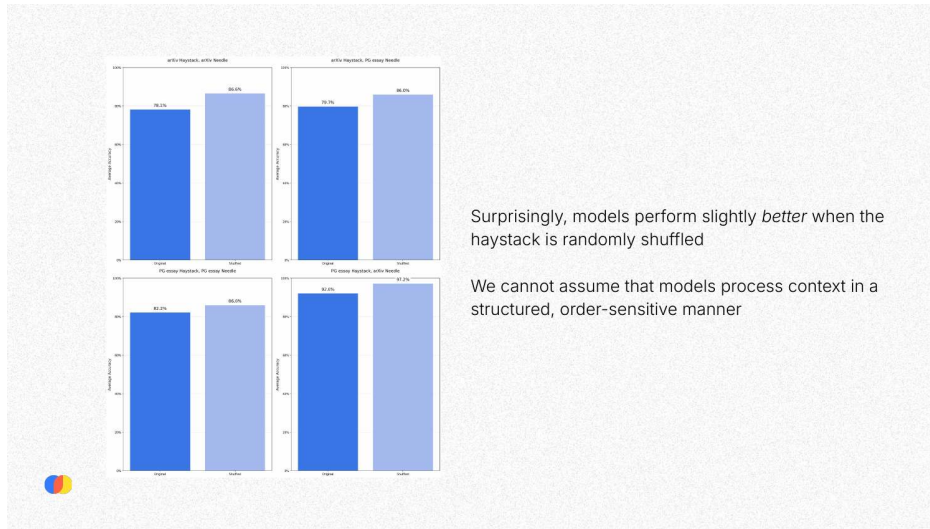


Figure 6.13: Surprising Results: Models Perform Better on Shuffled Context

(*Timestamp: 15:34*)

Counter-intuitively, the results showed that models performed slightly *better* when the haystack was randomly shuffled. This surprising finding suggests that LLMs do not necessarily process context in the linear, structured way humans do and that a disruption in logical flow can actually make a key piece of information harder, not easier, to find.

6.7 Experiment 4: Conversational Memory

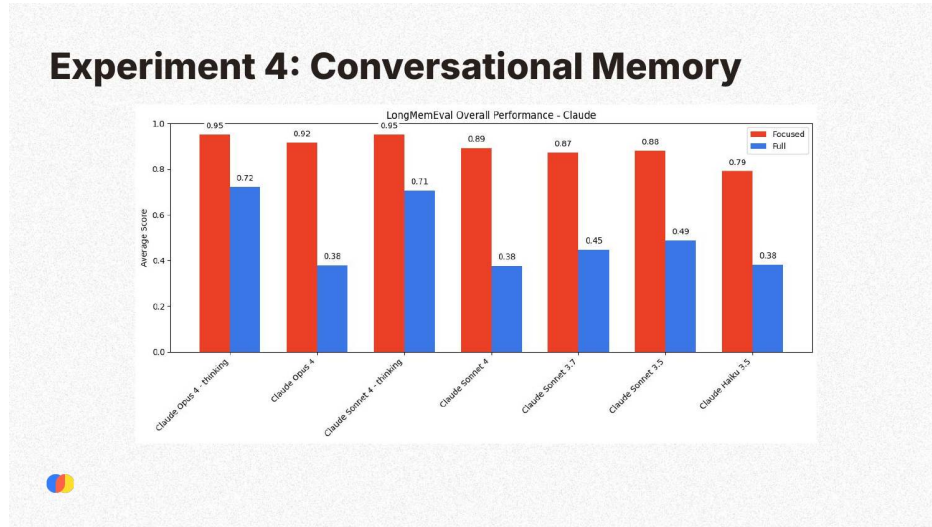


Figure 6.14: Experiment 4: Conversational Memory

(*Timestamp: 17:54*)

This experiment tested conversational memory using the LongMemEval benchmark. Models were tested under two conditions: a “focused” condition with only the relevant conversational history (around 100 tokens), and a “full” condition where the context was padded with irrelevant conversations up to 120k tokens. The results clearly show that all Claude models perform significantly better in the focused condition, demonstrating that irrelevant information degrades performance quickly.

6.8 Experiment 5: Simple Tasks Aren't Immune

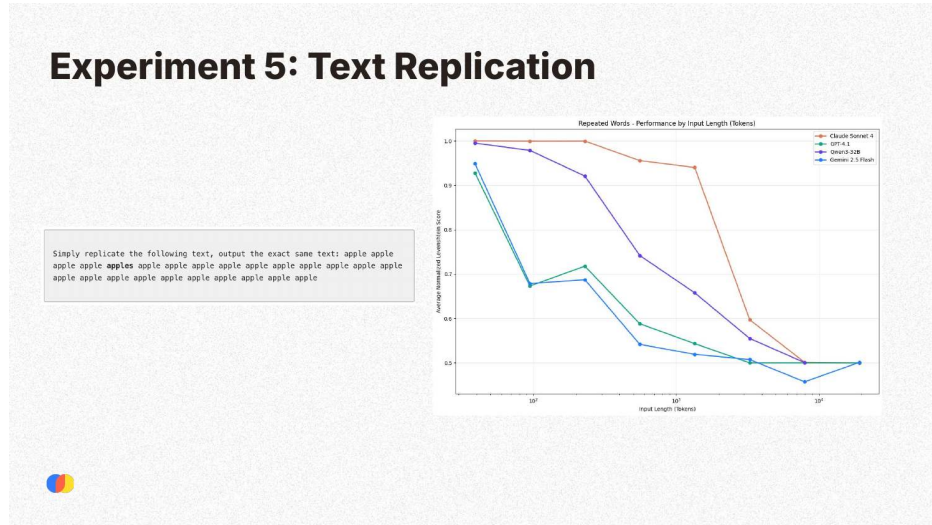


Figure 6.15: Experiment 5: Text Replication Task

(*Timestamp: 19:20*)

This experiment involved a very simple task: replicating a given text of repeated words. Despite the simplicity, all models showed a significant drop in performance as the input length increased. Some models exhibited strange failure modes; for example, at long input lengths, Claude models would refuse to generate the output, citing concerns about copyrighted material, while Gemini models would produce completely random outputs.

6.9 Key Takeaways

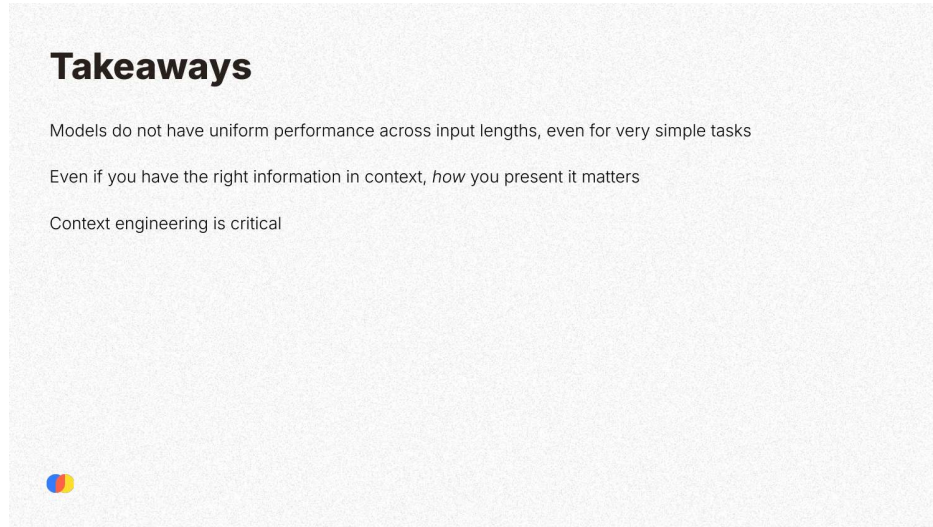


Figure 6.16: Key Takeaways

(*Timestamp: 20:15*)

The research provides three takeaways:

1. LLM performance is not uniform across input lengths, even for simple tasks.
2. Simply having the right information in the context is not enough; *how* that information is presented matters significantly.
3. As a result, thoughtful **context engineering** is critical for building reliable AI applications.

6.10 Practical Solutions: Context Engineering

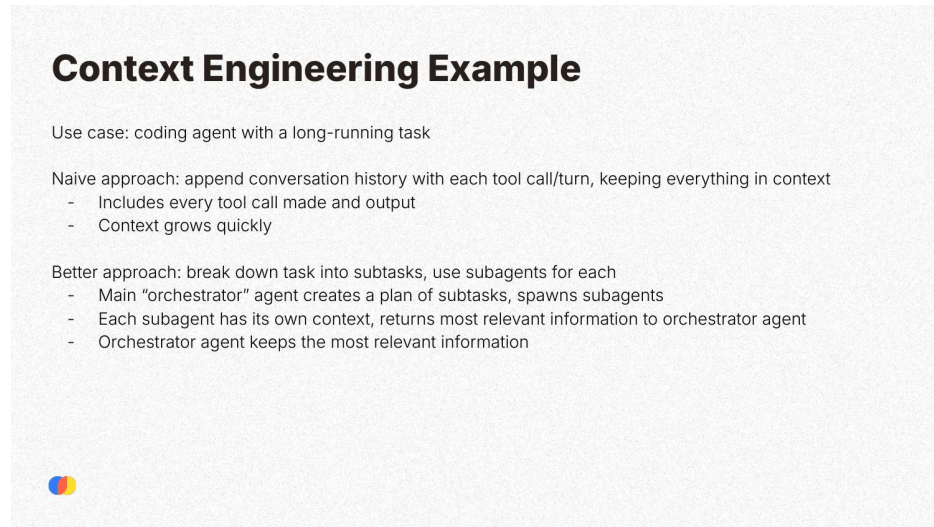


Figure 6.17: Context Engineering Example: Orchestrator and Subagents

(*Timestamp: 21:07*)

Kelly provides a practical example of context engineering for a coding agent with a long-running task.

- **Naive Approach:** Append the entire conversation history, including every tool call and output, to the context. This causes the context to grow quickly and become bloated with irrelevant information (e.g., the full content of a file read), leading to context rot.
- **Better Approach:** Use a main “orchestrator” agent that breaks the task into subtasks and spawns “subagents” for each one. Each subagent operates with its own clean, focused context. It completes its subtask and returns only the most relevant information to the orchestrator, which maintains a concise, filtered history. This prevents context overload and improves reliability.

6.11 Further Resources



Figure 6.18: Further Reading

(*Timestamp: 22:47*)

The presentation concludes by directing the audience to the full technical report and other related research on Chroma’s website, research.trychroma.com.

6.12 Q&A Session

- **Is the Needle in a Haystack (NIAH) benchmark pointless?** (*Timestamp: 06:54*) It’s not pointless, but its utility has diminished. It was useful for evaluating older models, which did show performance degradation on the task. However, modern

frontier models can now perform very well on this simple, lexically-driven task, which makes the benchmark unrepresentative of real-world use cases that require deeper semantic reasoning.

- **Did the research find that one model consistently resists context rot better than others across all tasks?** (*Timestamp: 23:57*) No, performance was “all over the place” and highly task-dependent. There was no single model that ranked first across all experiments. For example, Claude Sonnet 4 performed best on the repeated words task, while GPT-4.1 was the top performer on the Needle in a Haystack task. Each model has different strengths, and no model currently excels at all long-context tasks.
- **What is your advice for developers trying to find and mitigate context rot in their applications?** (*Timestamp: 27:32*) Start by qualitatively analyzing your system. Run a few examples with both short, focused context and long context bloated with irrelevant information. Compare the outputs: what did the model miss with the long context? What irrelevant information could be removed? There’s no single, generalizable solution, as optimal context engineering is highly application-dependent. A good starting point is to carefully examine the data you’re providing to the model and how you can make it more concise and relevant.
- **Prior research found a U-shaped retrieval curve, where information at the very beginning and very end of the context is recalled best. Does that still hold true?** (*Timestamp: 29:06*) In Chroma’s experiments, they did not observe this U-shaped pattern. They tested placing the “needle” at various positions throughout the context—from the beginning to the middle to the end—and found no consistent performance advantage for any particular position. While putting important information at the start or end is a common piece of advice, this

research suggests it may not be a reliable solution for mitigating context rot.

Conclusion

The field of RAG has evolved far beyond the simple “embed documents and search” approach that dominated 2023. Through these five expert presentations, we’ve explored the cutting-edge techniques that are reshaping how we build production-ready retrieval systems.

Key Takeaways

RAG’s Evolution: Ben Clavié showed us that RAG isn’t dead—it’s simply evolving. The naive single-vector approach has given way to sophisticated retrieval strategies that combine multiple techniques and leverage the full power of modern language models.

Better Evaluation: Nandan Thakur demonstrated that traditional IR metrics are insufficient for RAG systems. Modern benchmarks like FreshStack evaluate diversity, grounding, and coverage—not just relevance—reflecting the complex evidence-gathering needs of LLM-powered applications.

Reasoning Integration: Orion Weller’s work on Promptriever and Rank1 showed how instruction-following and reasoning capabilities can be embedded directly into retrieval models, enabling systems that understand nuanced queries and can adapt their behavior through natural language.

Late Interaction Models: Antoine Chaffin explained how ColBERT and similar approaches solve the fundamental information loss problem

of single-vector search by maintaining token-level representations, leading to superior performance on challenging tasks.

Multiple Representations: Bryan Bischof and Ayush Chaurasia’s framework revealed that the best retrieval systems don’t seek a single perfect representation but instead create multiple specialized “maps” of the same data, using intelligent routing to serve diverse user intents.

Looking Forward

These advances represent a fundamental shift in how we think about retrieval. Rather than trying to compress all information into a single vector, we’re moving toward systems that:

- Maintain rich, multi-faceted representations of data
- Understand and follow complex instructions
- Reason about relevance and user intent
- Adapt their behavior based on context
- Combine classical and modern techniques intelligently

The tooling ecosystem is rapidly maturing with libraries like PyLate, frameworks like RAGatouille, and growing vector database support for advanced retrieval methods.

Resources

Watch the full presentations: - Ben Clavié: “I don’t use RAG, I just retrieve documents” - Nandan Thakur: “Modern IR Evaluation for RAG” - Orion Weller: “Reasoning and Retrieval” - Antoine Chaffin: “Late Interaction Models” - Bryan Bischof & Ayush Chaurasia: “Multiple Representations”

Web version: You can also explore the interactive web version of this series at hamel.dev/notes/llm/rag/not_dead.html.

These insights come from our comprehensive [AI Evals course](#), where we teach engineers and product managers how to build robust evaluation systems for AI applications. The course covers not only RAG evaluation but the complete spectrum of LLM evaluation techniques. Readers of this handbook can access the course with a [35% discount](#).

The future of RAG is bright, and these techniques provide the foundation for building the next generation of intelligent information systems.