

Beyond Naive RAG: Practical Advanced Methods

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

2025-12-15

Table of contents

Preface	1
About This Book	1
1 I don't use RAG, I just retrieve documents	2
1.1 Annotated Presentation	3
1.2 Video	23
2 Modern IR Evals For RAG	24
2.1 Annotated Presentation	25
2.2 Reflections	56
2.3 Q&A Session	56
2.4 Video	58
3 Optimizing Retrieval with Reasoning Models	59
3.1 Annotated Presentation	60
3.2 Q&A Session	105
3.3 Video	106
4 Late Interaction Models For RAG	108
4.1 Annotated Presentation	109
4.2 Q&A Session	139
4.3 Video	141
5 RAG with Multiple Representations	142
5.1 Annotated Presentation	143
5.1.1 The Map is Not the Territory	143
5.1.2 Deconstructing RAG Buzzwords	143

5.1.3	A First-Principles View of RAG	149
5.1.4	The Three Responsibilities of an IR Engineer	152
5.1.5	Practical Application: Curving Space	155
5.1.6	Agents as Routers	159
5.1.7	Representations Must Evolve	161
5.1.8	Demo: Semantic Dot Art	162
5.1.9	System Diagrams	164
5.2	Q&A Session	165
5.3	Video	166
6	Context Rot	167
6.1	Annotated Presentation	168
6.2	Q&A Session	185
6.3	Video	187
7	You Don't Need a Graph DB (Probably)	188
7.1	The Hunt for a Silver Bullet	189
7.2	Do You Even Need RAG?	190
7.3	Deconstructing GraphRAG	192
7.4	Evaluation	193
7.5	When to Use a Graph Database	195
7.6	The Graph Hidden in Your Vector Search	196
7.7	Key Takeaways	196
7.8	Video	197
	Conclusion	198
	Looking Forward	198
	Resources	198

Preface

This book compiles talks on Retrieval Augmented Generation (RAG) from our [AI Evals course](#). The speakers are researchers and practitioners who build production retrieval systems.

About This Book

Each chapter is an annotated presentation with timestamped references to the original videos. You can skim the annotations or dive into the full talks.

The field has moved beyond the “embed documents and search” approach that dominated 2023. The topics covered here represent where retrieval is heading: better evaluation methods, reasoning-enhanced search, late interaction models, multiple representations, context engineering, and infrastructure decisions.

You can also view the web version [here](#).

If you want to learn more about AI Evals, check out our [AI Evals course](#). Here is a [25% discount code](#) for readers.

1 I don't use RAG, I just retrieve documents

As part of our [LLM Evals course](#), I hosted [Benjamin Clavié](#) to kick off a 5-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, etc.).

Below is an annotated version of the presentation, with timestamped links for each slide.

1.1 Annotated Presentation

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:

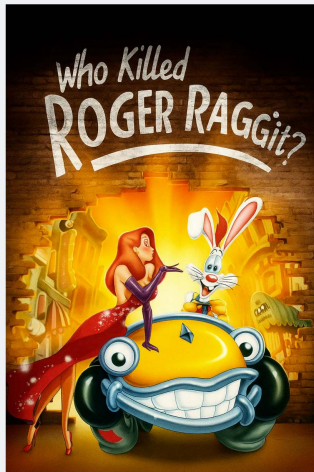


I call this **The Monopicture**. A singular picture of me taken 5 years ago which has gotten a life of its own as the only picture of me floating around.

- 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.



**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.

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.

RAG Visualised

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



(Timestamp: 00:04:08)

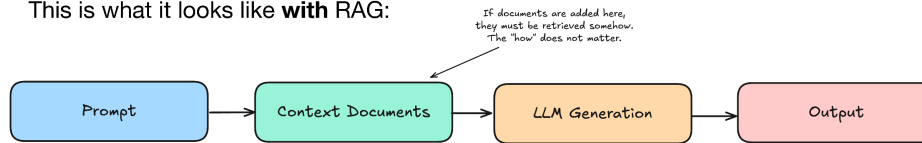
This slide presents a standard, simplified LLM call: a prompt goes in, the LLM generates a response, and an output comes out. This is the baseline for understanding how RAG changes the process.

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.

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**.

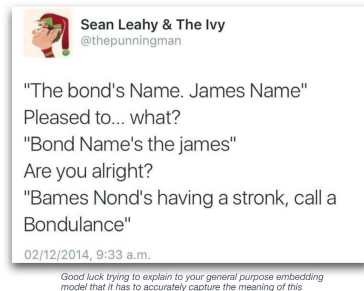


(*Timestamp: 00:05:10*)

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."

Why is Single-Vector Semantic Search not Good Enough?

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



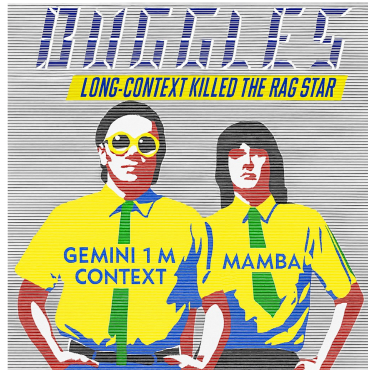
- Single-vector semantic search, where you embed a whole document (or chunk) into just one vector, is inherently limited.
- It has to compress a whole document's worth of meaning into **just a handful of dimensions**.
- This leads to **information loss**, prioritising what the **model assumes will be 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.

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.

RAG Lets Models Talk to The World

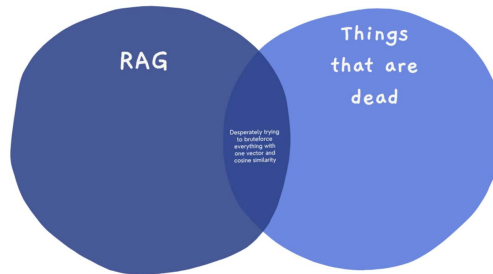
- Why did we and do we still need hard drives: because they **can store arbitrarily large, continuously updated volumes of information**.
- LLM weights are **inherently frozen**: the only way to add new information to them are **complex training processes**.
 - —> They **do not know anything about** 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.

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.

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).

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...

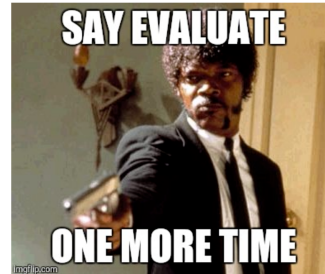
(Timestamp: 00:18:20)

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."

All Retrieval Needs Evaluations

Nandan Thakur

- Retrieval evaluations are difficult (ask Hamel and Shreya!)
- 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](#).

1.2 Video

Here is the full video:

https://youtu.be/Evlk9J-B_uc

2 Modern IR Evals For RAG

As part of our [LLM Evals course](#), I hosted [Nandan Thakur](#) for the second part of our 5-part mini-series on evaluating and optimizing RAG. Nandan 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.

Below is an annotated version of his presentation, with timestamped links for each slide.

2.1 Annotated Presentation



Modern IR Evaluation in the RAG Era

RAG Mini-Series: Guest Lecture #2

July 2, 2025

A title slide for a presentation. On the left is a photo of Nandan Thakur, a man with glasses and a green shirt, standing in a hallway. To the right of the photo, the text reads: "Nandan Thakur", "Fourth-year Ph.D. Student", "University of Waterloo", and "David R. Cheriton School of Computer Science", "University of Waterloo".

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."

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

Current (2021 onwards):

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





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

You'll probably recognize me from:



Twitter (@beirmug)





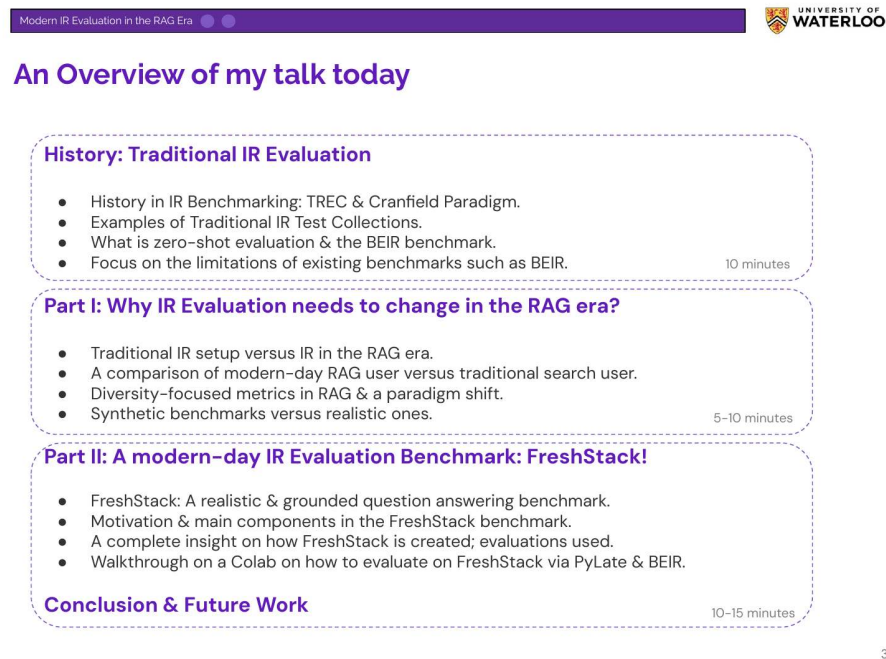
Multilingual Information Retrieval Across a Continuum of Languages



Figure 2.1: Speaker Introduction

(*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.

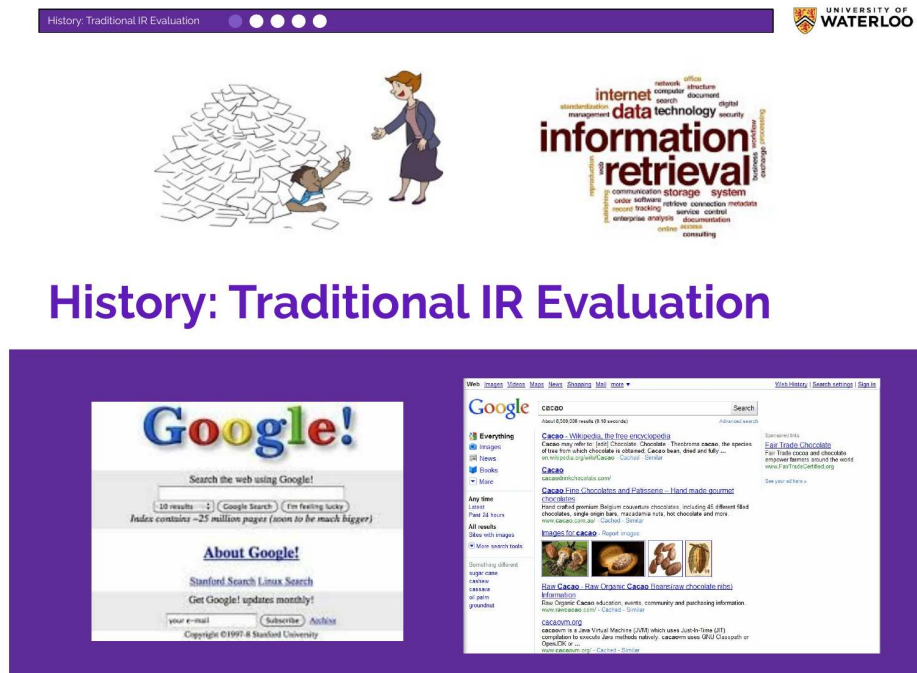


3

Figure 2.2: Overview of the Talk

(*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.



(Timestamp: 00:01:45)

This slide provides historical context. While RAG is new, Information Retrieval is a field with over 60 years of history. The slide contrasts an early Google interface with a modern one to show the evolution of web search.

[illegible]

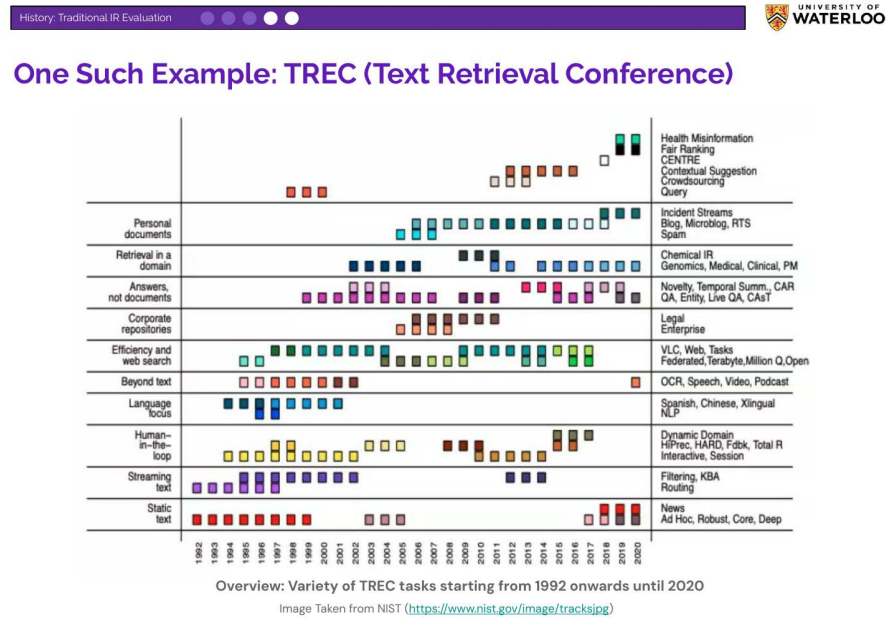
SMART Retrieval System. Published on June 1965!
Taken from <https://dl.acm.org/doi/10.1145/364955.364990>



Figure 2.4: IR's 60-Year History

(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

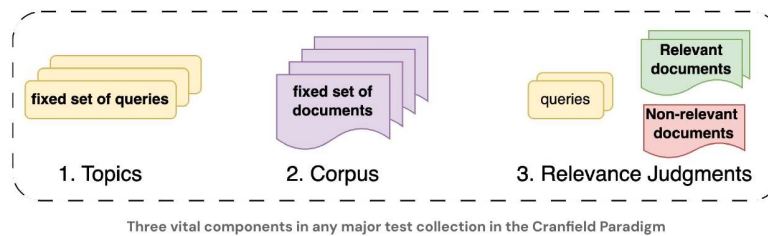
Figure 2.5: TREC Conference Tracks

(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.

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

Figure 2.6: The Cranfield Paradigm

(*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.

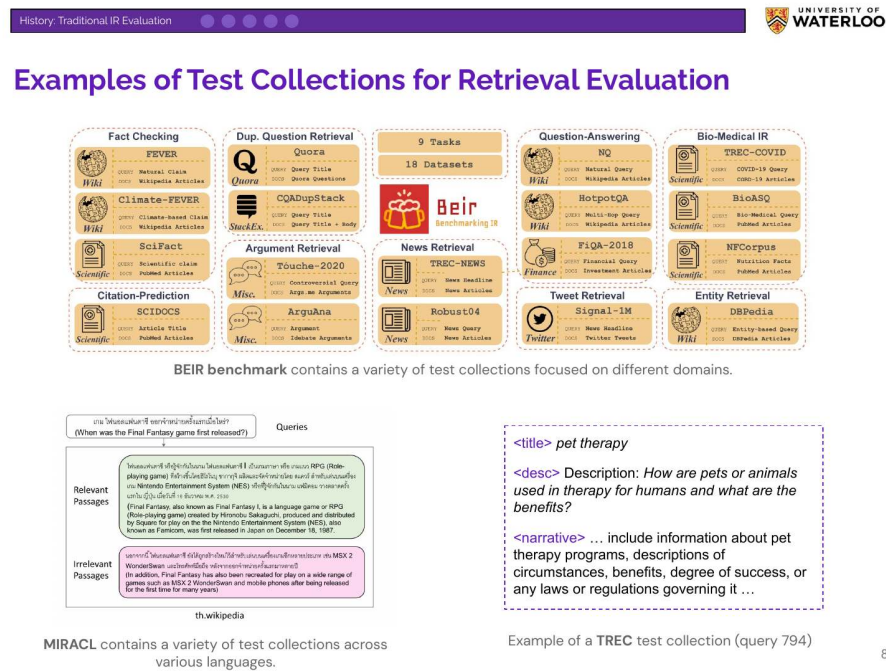


Figure 2.7: Examples of Test Collections

(Timestamp: 00:06:00)

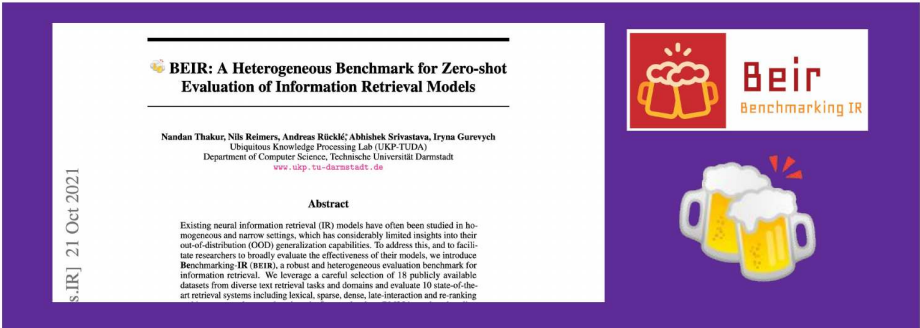
Nandan shows examples of modern test collections. He highlights **BEIR** for its diversity of tasks, **MIRACL** for multilingual retrieval, and the typical **TREC** query structure, which includes a **title**, **description**, and detailed **narrative**.

History: The BEIR Benchmark

UBIQUITOUS KNOWLEDGE PROCESSING TECHNISCHE UNIVERSITÄT DARMSTADT UNIVERSITY OF WATERLOO

 **Hugging Face**

Case in point: The BEIR Benchmark



The slide displays the title "BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models" and lists the authors: Nandan Thakur, Nils Reimers, Andreas Riecklé, Abhishek Srivastava, and Iryna Gurevych. It also includes the affiliation "Ubiquitous Knowledge Processing Lab (UKP-TUDA)" and the "Department of Computer Science, Technische Universität Darmstadt". The abstract discusses the limitations of existing neural information retrieval (IR) models and introduces BEIR as a robust and heterogeneous evaluation benchmark. The BEIR logo, featuring two beer mugs, is shown on the right side of the slide.

Figure 2.8: Introducing 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.

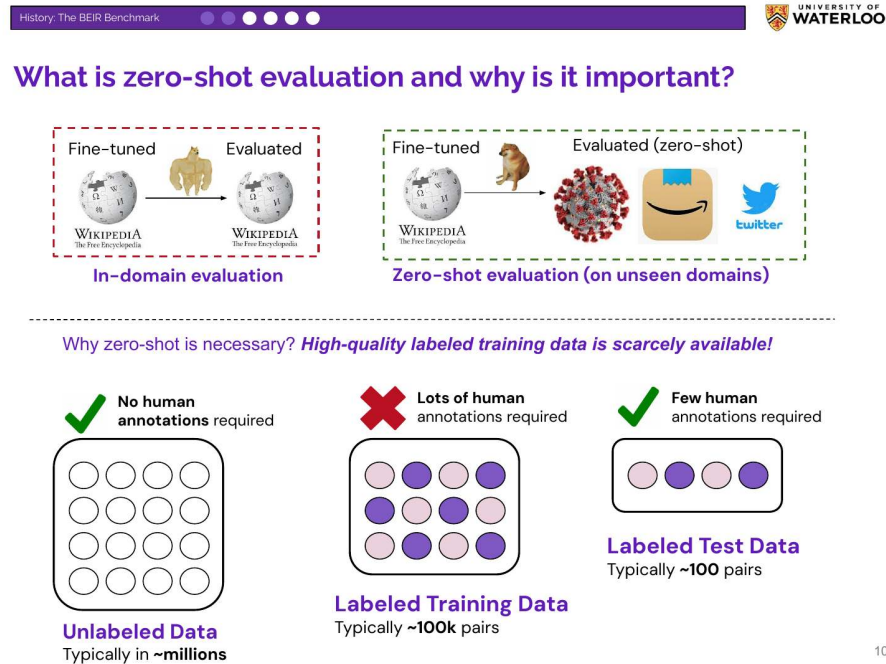


Figure 2.9: What is Zero-Shot Evaluation?

(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.

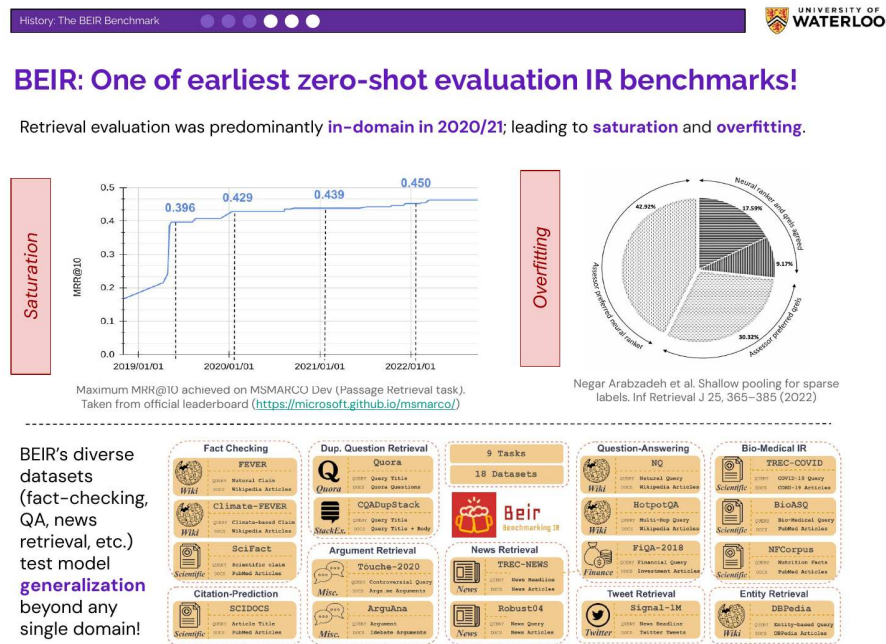
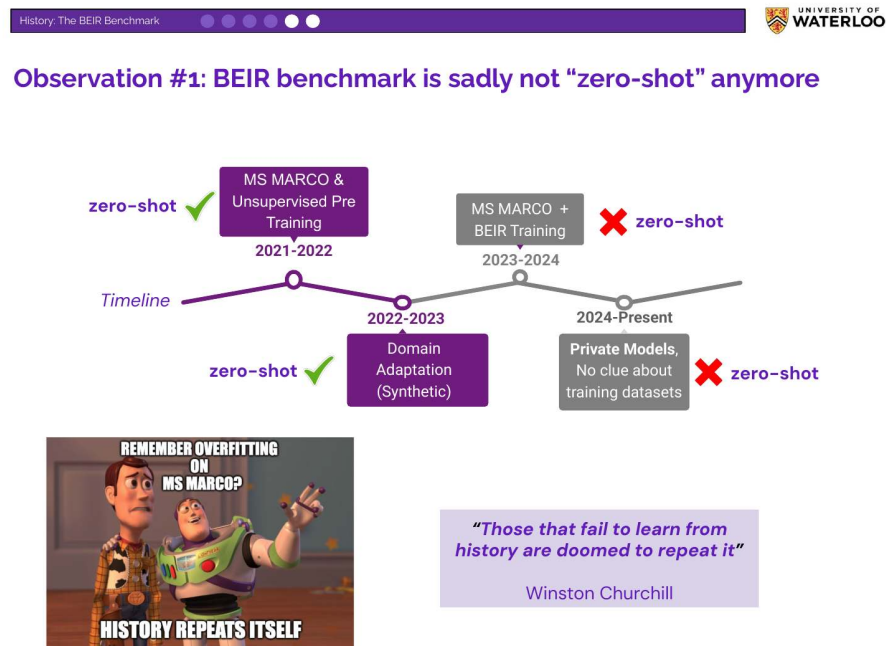


Figure 2.10: The Problem BEIR Solved: Overfitting

(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.



12

Figure 2.11: The Problem with BEIR Today

(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

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

13

Figure 2.12: Leaderboard Saturation

(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.

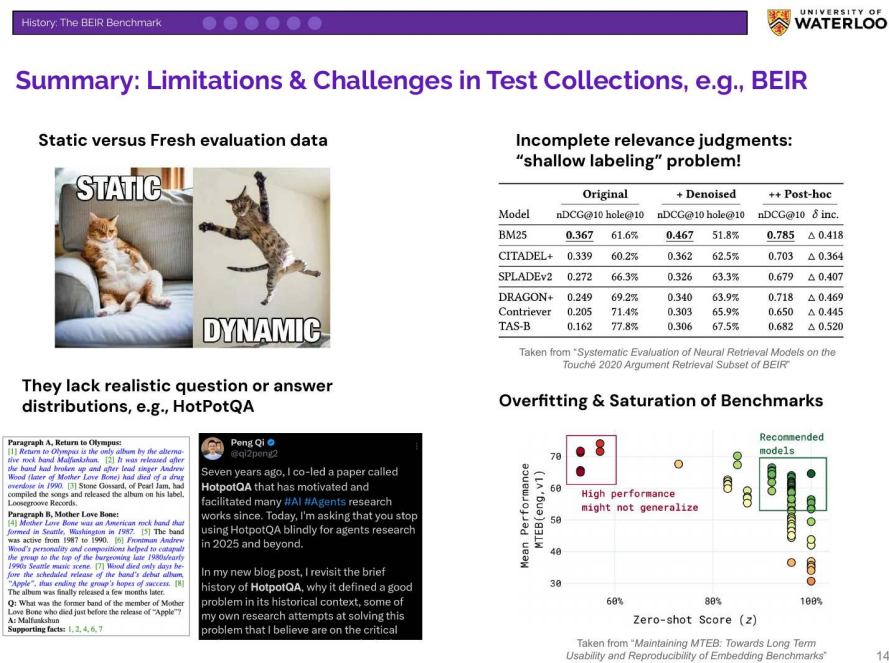


Figure 2.13: Summary of Challenges

(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.

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

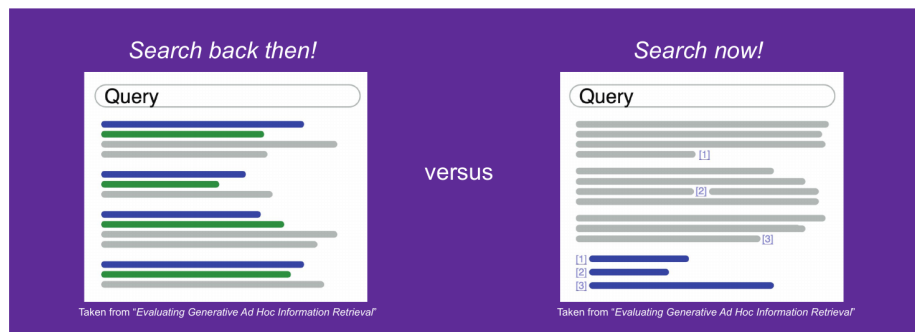


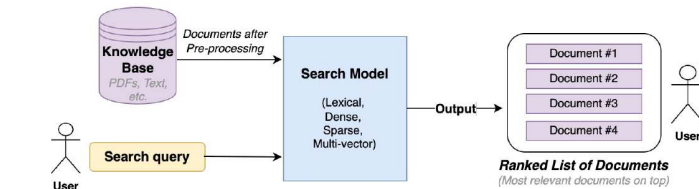
Figure 2.14: The Evolution of Search

(Timestamp: 00:17:28)

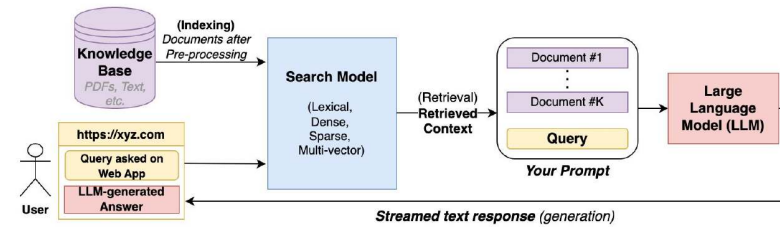
Nandan contrasts the old and new search paradigms. “Search back then” shows a ranked list of links, while “Search now” shows a generated answer block with citations, characteristic of RAG systems.

What has changed in Retrieval in the recent days?

Information Retrieval before the RAG Era



Information Retrieval in the RAG Era



16

Figure 2.15: Information Retrieval: Before and After RAG

(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.

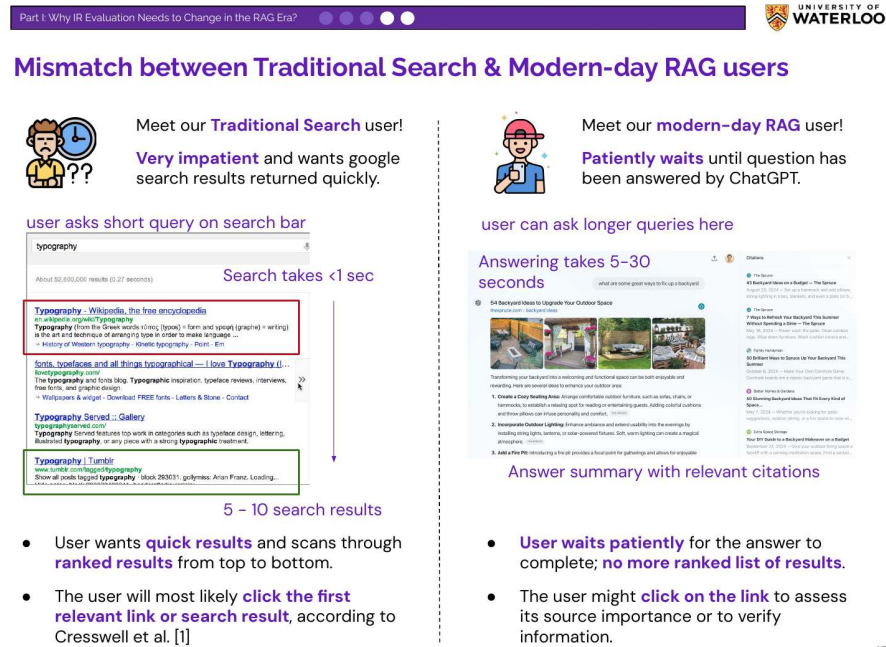


Figure 2.16: Traditional vs. Modern Day RAG Users

(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.

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

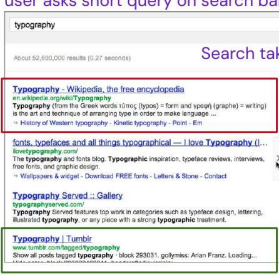
Mismatch between Traditional Search & Modern-day RAG users



Meet our **Traditional Search** user!

Very impatient and wants google search results returned quickly.

user asks short query on search bar



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)

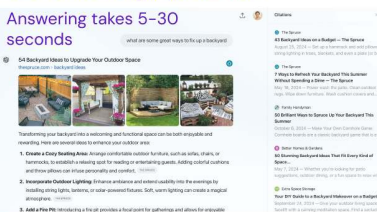


Meet our **modern-day RAG** user!

Patiently waits until question has been answered by ChatGPT.

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)

Figure 2.17: The Mismatch in Evaluation Objectives

(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 (more on that later).

42

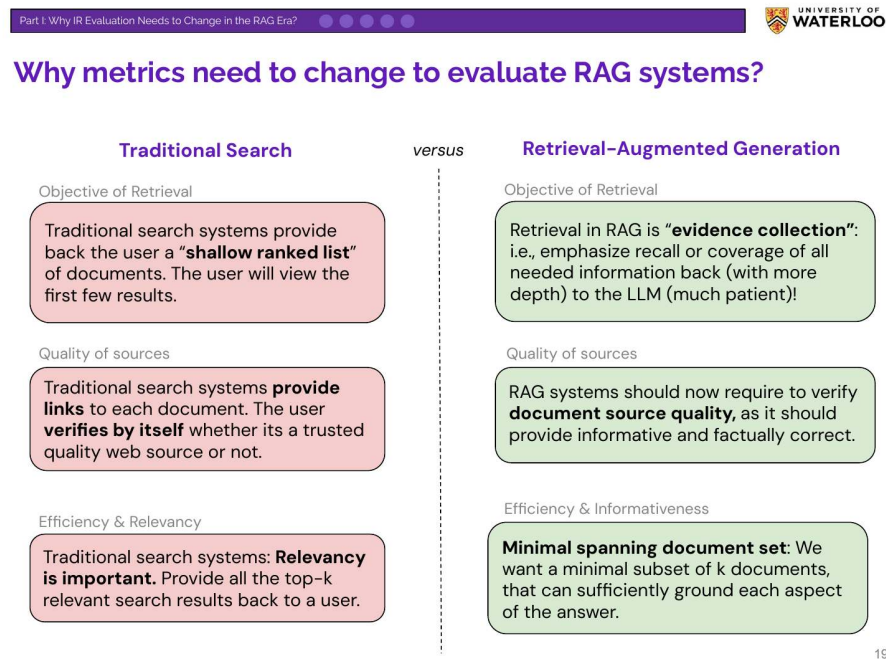


Figure 2.18: Why RAG Metrics Need to Change

(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.



UNIVERSITY OF
WATERLOO



databricks

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

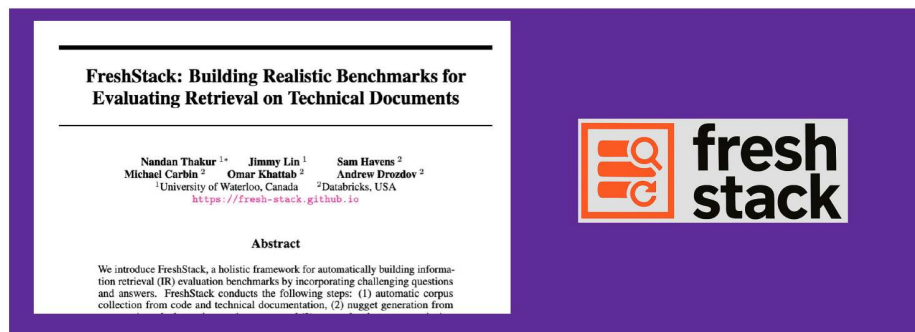


Figure 2.19: Introducing FreshStack

(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**?

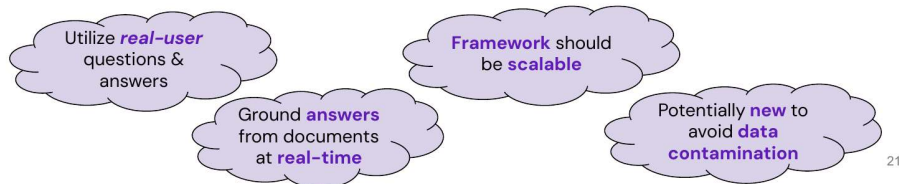
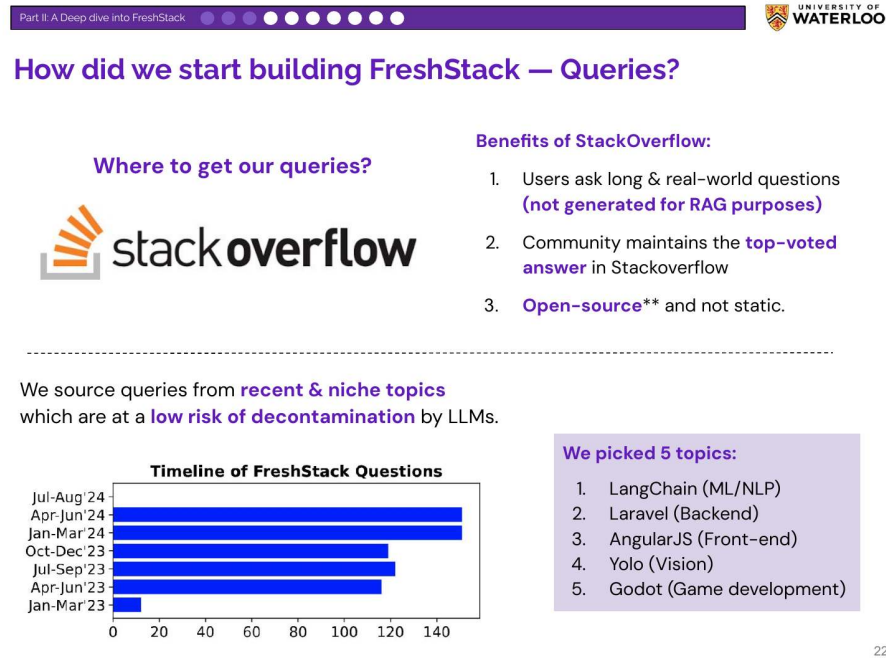


Figure 2.20: Motivation for FreshStack

(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.



22

Figure 2.21: FreshStack Queries: Stack Overflow

(*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?

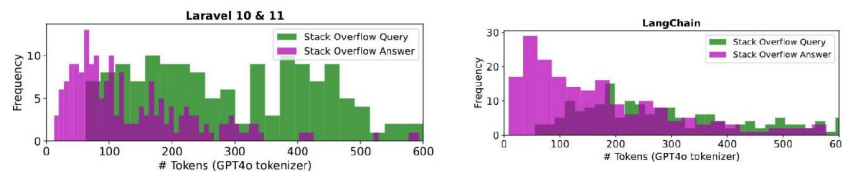


GitHub Repositories

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

Figure 2.22: FreshStack Corpus: GitHub Repositories

(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.

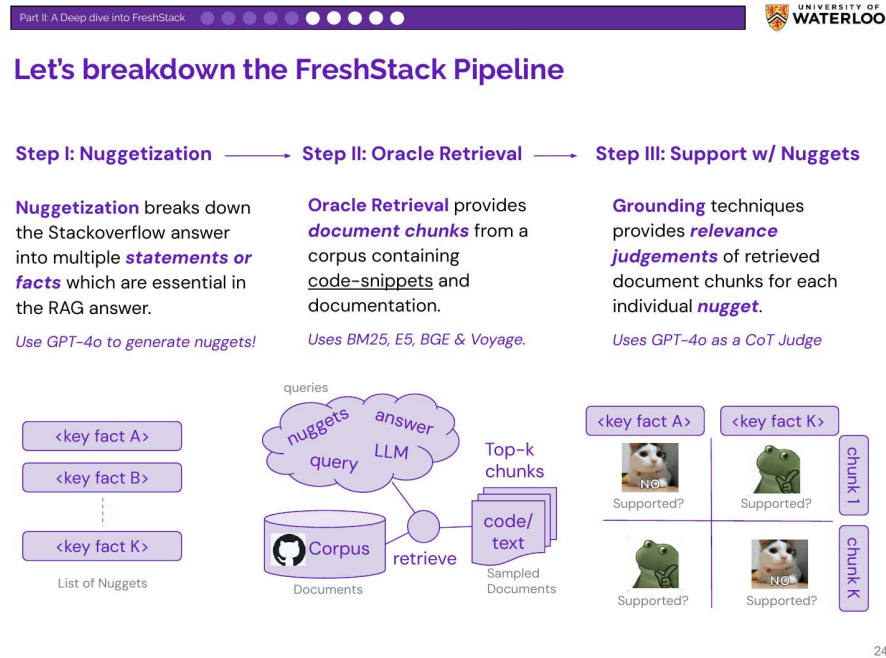


Figure 2.23: The FreshStack Pipeline

(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#migration-to-0416---november-7-2023>

I am not sure what changes are necessary to work with this.

```
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 used for list[str] # Adjust based on your document format (single str)
    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="tmp/chroma_db")
    db.persist()
    print(f"Saved {len(chunks)} chunks")
    return db
```

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

python langchain chromadb

Stackoverflow Answer

1 Answer

Sorted by: Highest score (default)

1 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

Figure 2.24: Step I: Nuggetization Example

(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.

Part II: A Deep dive into FreshStack

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

FreshStack Nuggets

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.

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
documents = [...]
embedding.embed_documents(documents)
embedding.persist_directory("tmp/chroma_db")
db.persist()
                    
```

Supports?

NO

YES

YES

YES

Sampled Doc #1

```

langchain / templates / intel-rag-seen / ingest.py
ccurme community[patch]: deprecate langchain_community Chroma in favor of la...
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 InstructablePDFLoader
6 from langchain_community.embeddings import HuggingFaceEmbeddings
7 from langchain_community.vectorstores import Chroma
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 = InstructablePDFLoader(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 = InstructablePDFLoader(doc, model="gpt-4o", strategy="Text")
25     chunks = loader.load_and_split(text_splitter)
26
27     print(f"Done preprocessing. Created", 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     )
                    
```

Figure 2.25: Step II & III: Oracle Retrieval & Nugget Support

(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.

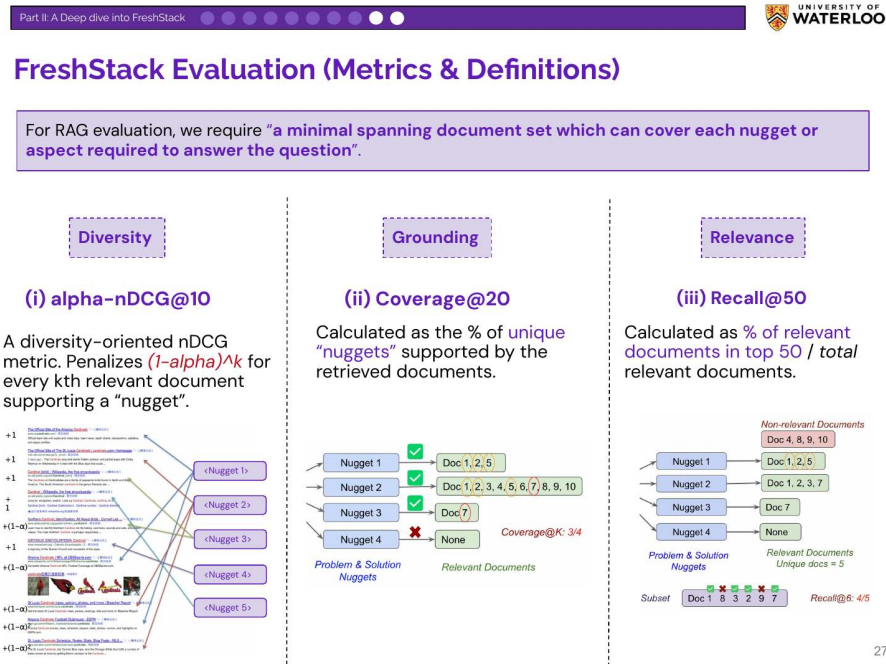


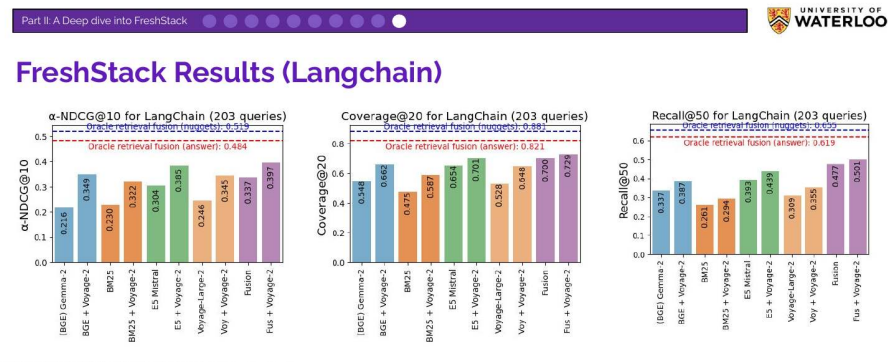
Figure 2.26: FreshStack Evaluation Metrics

(Timestamp: 00:31:26)

Nandan introduces the three metrics used in FreshStack, which provide a holistic view of RAG retrieval performance (for this specific use case):

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.



FreshStack Results highlights & Key Takeaways:

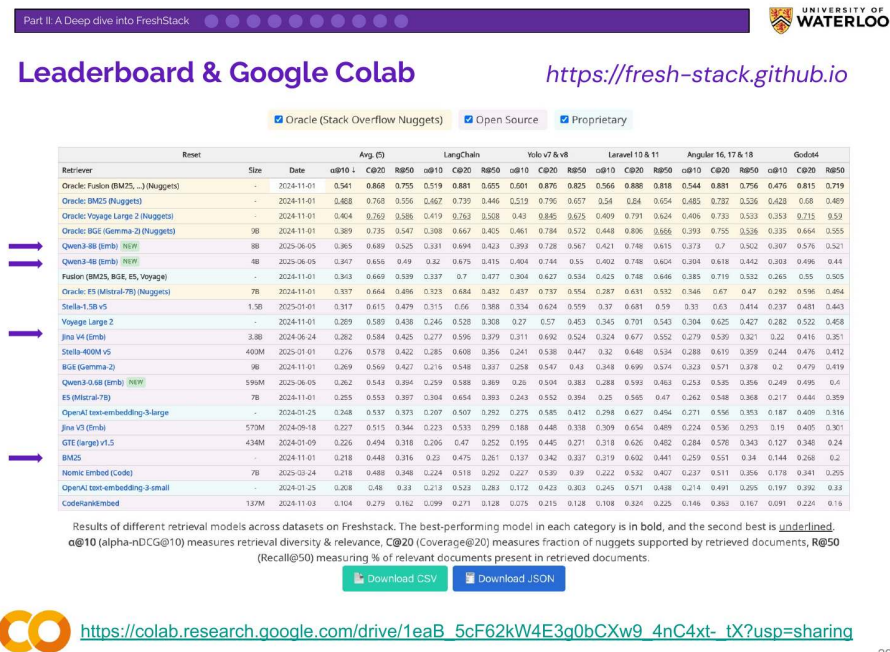
1. Current retrieval techniques (using raw questions) **are unable** to retrieve relevant documents.
2. No individual model performs best across all topics in FreshStack.
3. Rerankers **enhance** retrieval performance (voyage-2), however **not** without explicit fine-tuning.
4. Oracle approaches using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

28

Figure 2.27: FreshStack Results & Takeaways

(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.



29

Figure 2.28: The FreshStack Leaderboard & Colab

(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

Figure 2.29: What Did We Learn Today?

(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

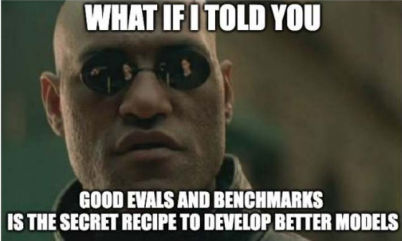
MIRACL 
Multilingual Information Retrieval Across a Continuum of Languages

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

Figure 2.30: Thank You

(*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.

2.2 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.

2.3 Q&A Session

- **How well does FreshStack generalize outside of coding-related domains?** Nandan feels that FreshStack approach should generalize well outside of coding-related domains, but more experiments are needed. The pipeline of nuggetization, retrieval, and grounding can be applied to any domain with a corpus and question-answer pairs, allowing for the creation of a “FreshStack” for finance, law, or other areas.

- **For domain-specific RAG, should people build their own evaluations mimicking this approach?** Yes, but the relative importance of each dimension (Diversity, Grounding, Relevance) depends on the use case. For patent search, recall is critical. For a comparative question, coverage of all viewpoints is important. Weighting these metrics into a combined score according to the product needs is a reasonable approach.
- **Does the FreshStack leaderboard align with intuitions about the best retrieval models?** The leaderboard shows that some popular proprietary models can underperform classic baselines like BM25 on this benchmark. It also highlights that recent open-source models (e.g., Qwen3, Stella) are now highly competitive with top closed-source offerings, and that performance does not always scale with model size.
- **How does this evaluation framework interact with advanced retrieval methods?** The evaluation framework is method-agnostic. While current models on the leaderboard use single-step retrieval, more complex methods like multi-query decomposition or late-interaction models can be evaluated using the same principles. The final set of retrieved documents can still be measured for its diversity, coverage, and recall, regardless of how it was generated.
- **What are you working on next?** Nandan is focusing on generating high-quality synthetic training data for RAG systems. He is also an organizer for the TREC RAG track, where he plans to introduce the diversity and grounding metrics from FreshStack to help push the community toward more robust evaluation standards.

2.4 Video

Here is the full video:

<https://youtu.be/Trps2swgeOg>

3 Optimizing Retrieval with Reasoning Models

As part of our [LLM Evals course](#), I hosted [Orion Weller](#) from Johns Hopkins University for our 5-part mini-series on evaluating and optimizing RAG. Orion’s research 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.

His approach is showcased with two models:

- **Promptriever (bi-encoder):** Creates “instruction-aware” embeddings by training on a novel dataset containing **instruction negatives**. These are examples where a document is relevant to a query but *not* its specific instruction (e.g., “find a document using a metaphor”). This forces the model to encode abstract instructions directly into the query embedding, allowing it to surface documents from a massive corpus that a standard retriever would miss.
- **Rank1 (reranker):** A smaller model fine-tuned by **distilling the reasoning traces** of a larger model. It generates an explicit, auditable chain of thought to assess relevance. This specialized

training makes it exceptionally good at reasoning, allowing it to uncover novel, relevant documents invisible to previous systems.

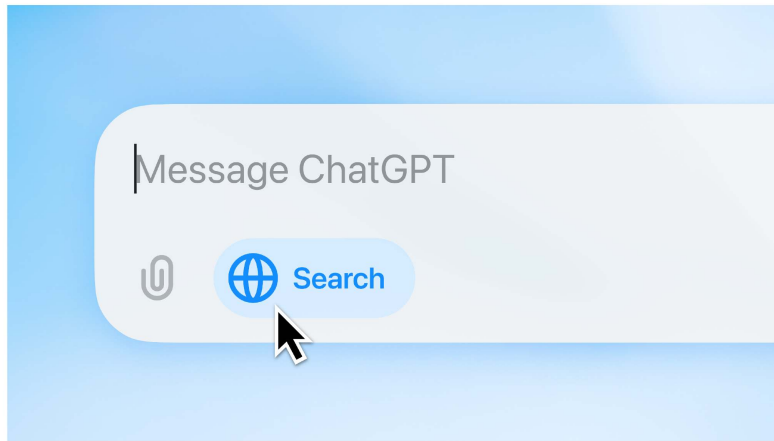
Below is an annotated version of his presentation with timestamped links.

3.1 Annotated Presentation



(*Timestamp: 00:00:00*)

Title slide for Orion Weller's talk on integrating instruction following and reasoning into information retrieval (IR).



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.

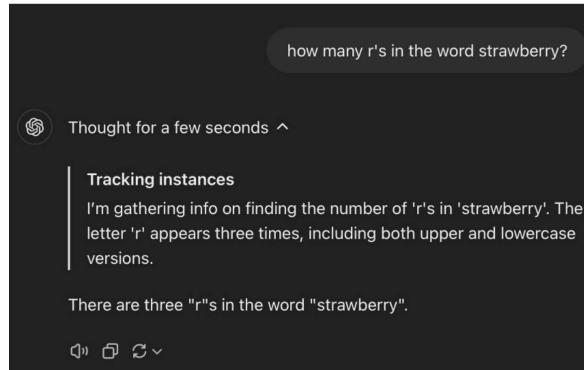
Scourin' the sea wide,
RAG finds treasure in the fog—
Answers hoist with pride!

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. This ability to break down and reason about a task is a major focus in the LLM community.

What about in retrieval?

5

(Timestamp: 00:01:41)

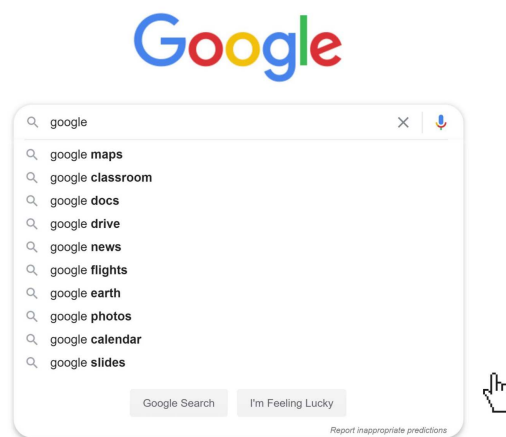
With these LLM capabilities established, Orion poses the talk's central question: how can we integrate these instruction-following and reasoning abilities directly into the retrieval process, moving beyond simply using LLMs to summarize search results?



6

(*Timestamp: 00:01:52*)

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

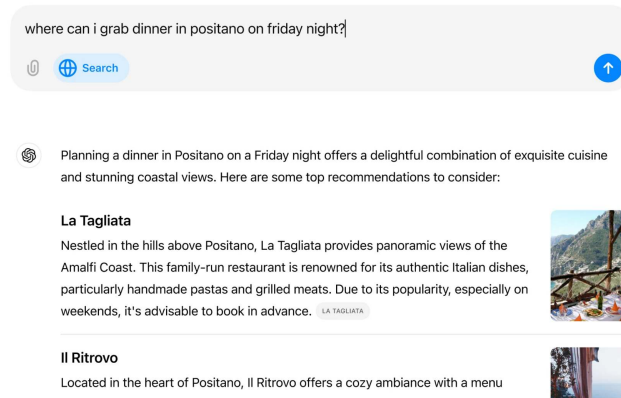


7

(*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.

SearchGPT ●



(*Timestamp: 00:02:17*)

This slide shows a modern “SearchGPT” style interface, which provides a generated, conversational answer.

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. Orion's work aims to change this.

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, highlighting the brittleness of keyword-only approaches.

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.” This improves on keyword search but still falls short of true instruction following.

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 the next paradigm: **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.

This example illustrates the limitation of reranking results of standard semantic search (a popular approach in RAG). Such an approach would fail here because a semantic search model has no way to understand the constraint “uses an extended metaphor.” It would rank documents based only on their relevance to “data privacy,” meaning the “Wolves” document might not score high enough to even be considered by the reranker. To solve this, the instruction must influence the **initial**

retrieval to change which documents are considered relevant in the first place.

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, for example by lowering its relevance threshold to ensure high recall.

What is an instruction in IR?

Doc attributes: date, length, source	NLU aspects: sentiment, style
---	----------------------------------

Logical conditions
AND/OR/NOT

16

(*Timestamp: 00:06:42*)

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

First, instructions can refer to **document attributes** like date, length, or source. A retriever should understand these from document content without needing pre-processed metadata. Second, they can involve **NLU aspects**, such as document sentiment or writing style. Third, they can include **logical conditions**, combining multiple constraints with operators like AND, OR, and NOT.

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

We're so used to prompting LMs

Let's use retrievers the **same way**

17

(Timestamp: 00:07:31)

We are already used to prompting LLMs with complex instructions. Since modern retrievers are built on LLMs, we should be able to interact with them in the same way.



Promptriever
fast embedder



Rank1
strong but slow

19

(*Timestamp: 00:07:45*)

Orion introduces two models from his research that embody these principles. First is **Promptriever**, a fast embedding model for following instructions during initial retrieval.

Second is **Rank1**, a powerful but slower reranker that uses reasoning and test-time compute for nuanced relevance judgments.



Promptriever: Instruction-Trained Retrievers Can Be Prompted Like Language Models



Orion Weller



Benjamin
Van Durme



Dawn Lawrie



Ashwin Paranjape



Yuhao Zhang



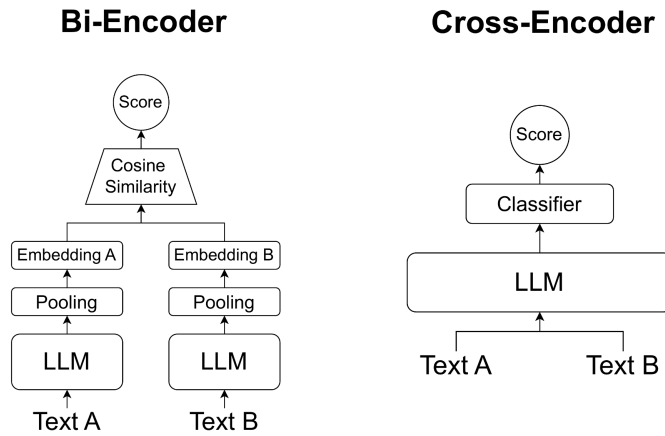
Jack Hessel



20

(*Timestamp: 00:08:17*)

First, we will dive into Promptriever. The associated paper's title is "Instruction-Trained Retrievers Can Be Prompted Like Language Models," a collaboration between Johns Hopkins and Samaya AI.



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

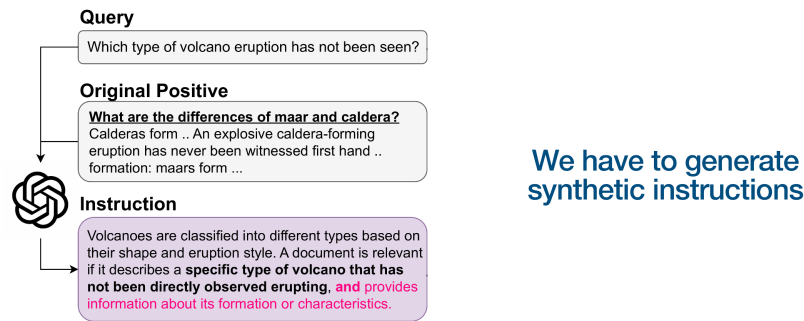
22

(*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. Creating a new dataset with instruction-based queries was necessary to teach the model this capability.

Generating Training Data



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.

The core of the data generation is to use a LLM to look at the query and the relevant document and synthetically generate a detailed **instruction** that makes the relevance criteria more specific. A crucial part of this process was also generating **instruction negatives**, which are documents that are relevant to the query but irrelevant to the *instruction*, forcing the model to pay attention to the new constraints.

Experimental Settings



Started from RepLLaMA's (using LLaMA-2) training recipe for a direct comparison



Evaluation is done on in-domain data (MSMarco), instruction data, and out of domain data

26

(*Timestamp: 00:10:36*)

To ensure a fair comparison, they started with the training recipe from **RepLLaMA**, an existing model that fine-tunes LLaMA-2 for retrieval, and only added their new instruction-based training data. The evaluation was comprehensive, testing on in-domain data (MSMARCO), new instruction-following datasets, and out-of-domain datasets to measure generalization.

Evaluation Data

27

(*Timestamp: 00:11:20*)

This slide introduces the two key instruction-following datasets for evaluation.

Query: I am looking for articles of Teflon use that may pose potential health risks.

Instruction: Find articles referencing Teflon as potential health risk. Relevant documents must give a reason or explain why the use of Teflon is a concern. The application of Teflon ~~can be of any means~~, **has to be related to chemicals, specifically** as long as the health risk is affecting human.

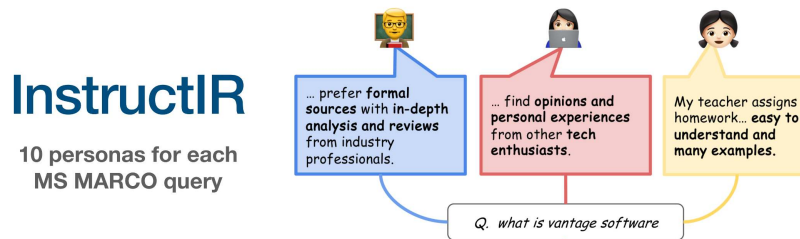
FollowIR

p-MRR ranges
from -100 to 100

28

(Timestamp: 00:11:22)

The first is **FollowIR**, where queries are modified with clarifying instructions. The p-MRR metric measures the ability to adapt, with positive scores indicating successful instruction following.



29

(Timestamp: 00:12:14)

The second is **InstructIR**, which associates queries with user personas (e.g., student, professional). The model must understand the persona's implicit needs to retrieve appropriate documents.

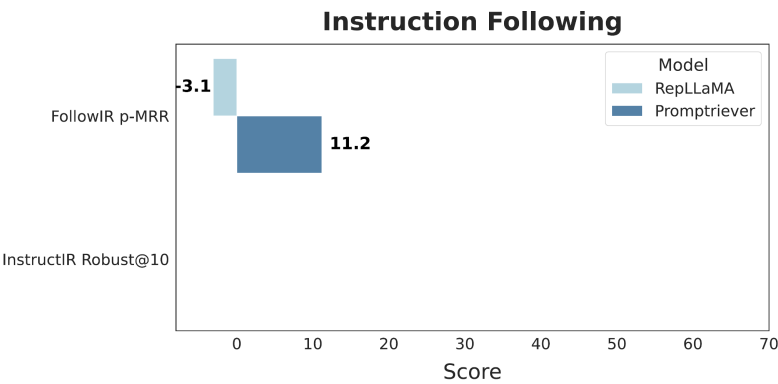
Results

30

(Timestamp: 00:12:30)

This slide introduces the experiment results.

Results

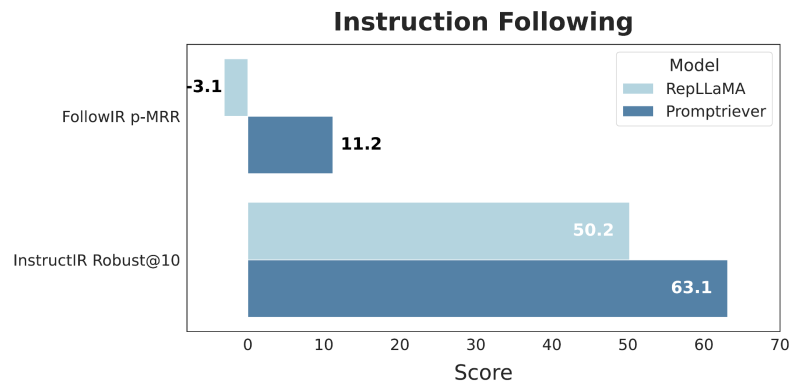


32

(*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



33

(*Timestamp: 00:12:50*)

On InstructIR, Promptriever again significantly outperforms the baseline by understanding the nuanced needs of different user personas.

Results

When evaluating on data with no set instruction, what prompt to use?

- We show with no prompt (standard)
- We come up with 10 generic prompts and use the best on the dev set for test

Prompts

- Be careful when assigning relevance as your job is on the line and I will give you a 1000 dollar tip.
- Think carefully about these conditions when determining relevance.

...

35

(*Timestamp: 00:12:59*)

How do these models perform on standard datasets without pre-defined instructions?

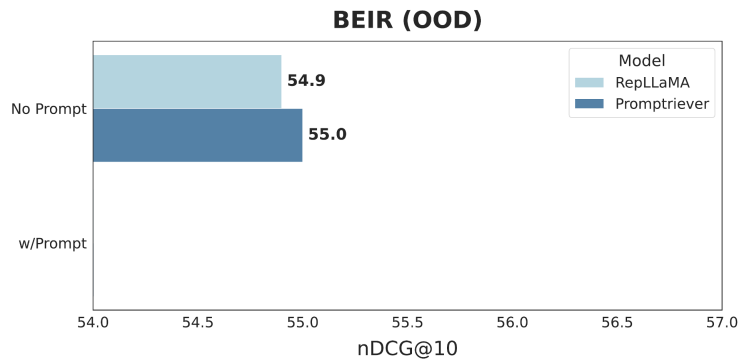
When evaluating on standard data, what prompt should be used?

The first option is using no prompt, the standard for evaluating existing retrieval models.

The second option, unique to instruction-following models, is to experiment with generic prompts (e.g., “Find the most relevant document”) and use the best-performing one, a form of prompt engineering for retrieval.

This slide shows generic prompts created to encourage more careful retrieval, such as “Be careful when assigning relevance as your job is on the line.”

Results



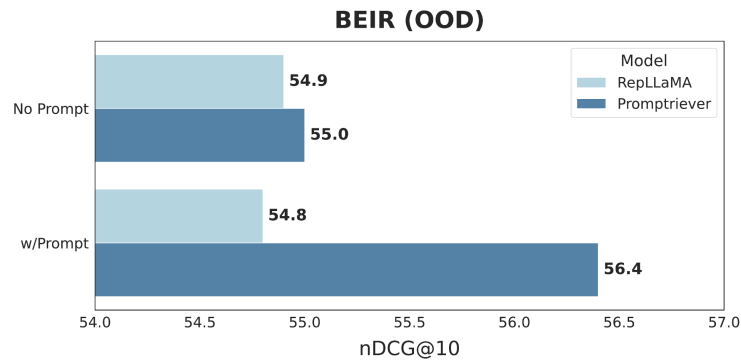
36

(*Timestamp: 00:13:58*)

This slide introduces the BEIR benchmark for out-of-domain (OOD) generalization.

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

Results



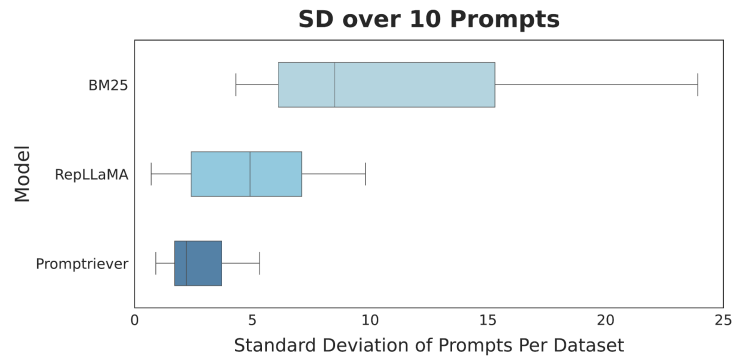
37

(*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.

The [Promptriever paper](#) calls this **zero-shot hyperparameter optimization via prompting**. Instead of tweaking numerical settings like a relevance threshold, one can change the model’s behavior by tweaking the natural language prompt. An instruction like “find documents with high recall” causes the model to adjust its internal strategy to retrieve a broader set of results because it has been trained to understand the *intent* behind such commands.

Results



39

(*Timestamp: 00:14:45*)

To test if the model understands the *meaning* of prompts, they measured the standard deviation of performance across 10 paraphrased versions of the same prompt. Promptriever shows much lower variance than keyword-based (BM25) or standard semantic models (RepLLaMA). This indicates it is robust to wording changes and understands the underlying intent, rather than just matching keywords.

Summary

- **With the right data**, you can have retrievers that are promptable just like LM
- Unlocks new types of queries!
- Don't need to be picky about keywords, just tell the model what you want in natural language

40

(*Timestamp: 00:15:16*)

This slide summarizes takeaways from the Promptriever research:

1. With the right training data, even fast bi-encoder retrievers can be made promptable like larger LLMs.
2. This unlocks new types of queries based on meta-level properties like style, sentiment, or logical constraints.
3. Users no longer need to be picky about keywords; they can tell the model what they want in natural language.



Rank1: Test-Time Compute for Information Retrieval



Orion Weller



Kathryn Ricci



Eugene Yang



Andrew Yates



Dawn Lawrie



Benjamin
Van Durme

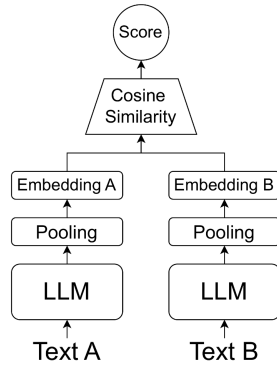


43

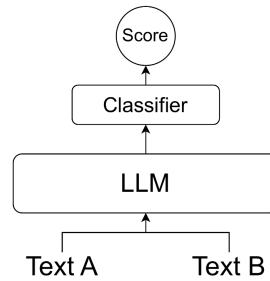
(Timestamp: 00:16:08)

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.

Bi-Encoder



Cross-Encoder

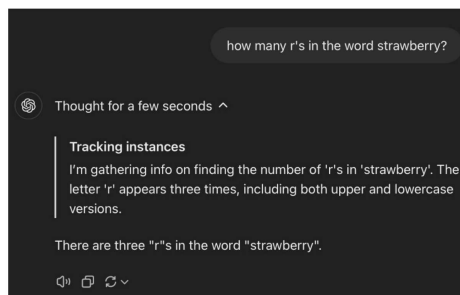


44

(*Timestamp: 00:16:13*)

Rank1 is a **Cross-Encoder**, processing the query and document together for a powerful but slower relevance judgment.

Test-Time Compute

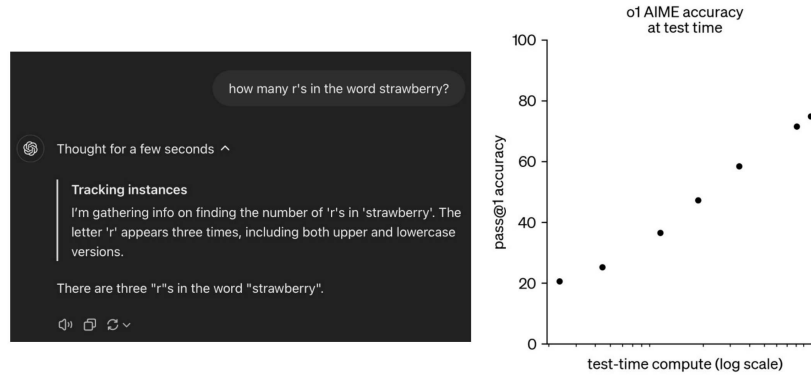


45

(*Timestamp: 00:16:22*)

Rank1 leverages **Test-Time Compute**, where the model generates a reasoning trace to arrive at its decision.

Test-Time Compute



45

(*Timestamp: 00:16:25*)

The chart on the right (from OpenAI's o1 model) shows that as you increase the amount of computation (reasoning chain length), model accuracy on complex tasks increases dramatically.

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 is asked to determine relevance. The model generates a detailed reasoning trace, identifying key phrases, analyzing the relationship between query and document, and questioning its own interpretations ("But wait..."). It uses this step-by-step reasoning to arrive at a final **false** judgment.

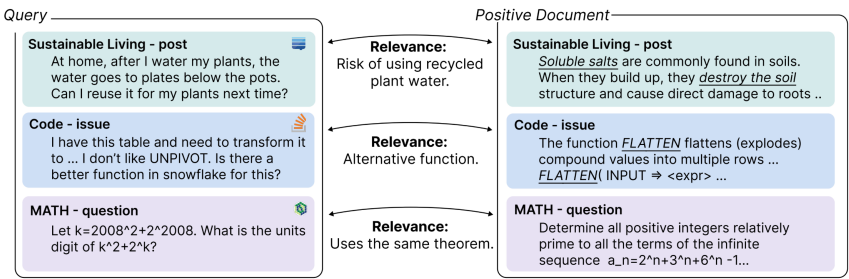
Evaluation Data

49

(Timestamp: 00:18:01)

The talk now moves to the evaluation data for Rank1.

BRIGHT dataset



50

(Timestamp: 00:18:06)

The primary evaluation dataset is **BRIGHT**, designed to test deep reasoning with unique relevance definitions that go beyond topic matching, such as finding a math problem that uses the same theorem.

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.

Results

52

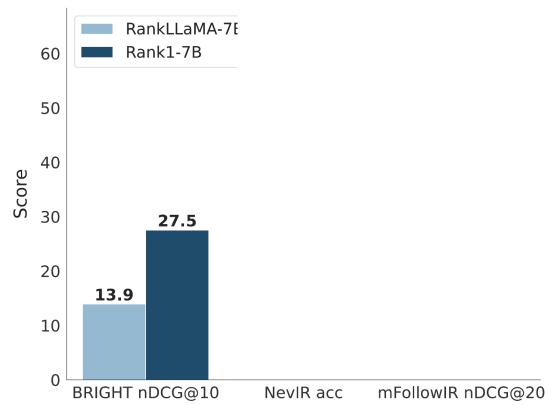
(*Timestamp: 00:19:35*)

This slide introduces the Rank1 experiment results.

Results

We evaluate on a broad range of tasks:

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



54

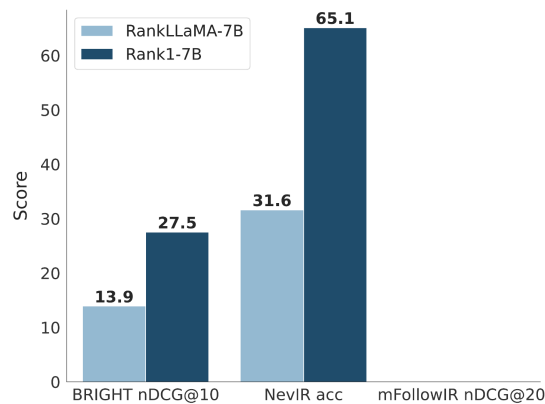
(Timestamp: 00:19:38)

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

We evaluate on a broad range of tasks:

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



55

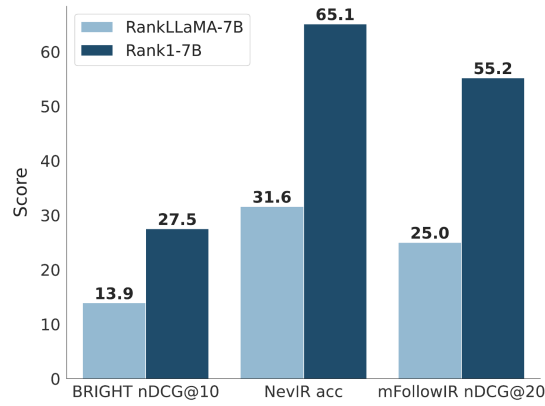
(Timestamp: 00:20:00)

On the NevIR negation task, the gain is even more dramatic, with Rank1 more than doubling the baseline's score.

Results

We evaluate on a broad range of tasks:

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



56

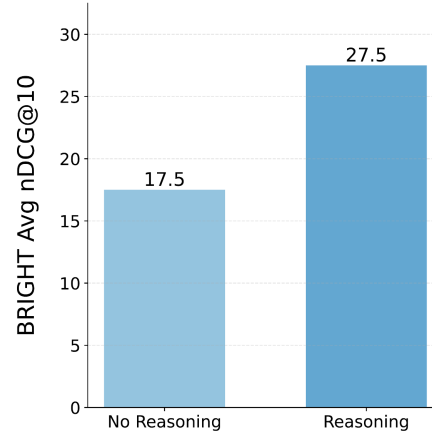
(*Timestamp: 00:20:05*)

The trend continues on the mFollowIR instruction-following task, where Rank1 again more than doubles the baseline's performance.

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.

Old Evaluation Data

59

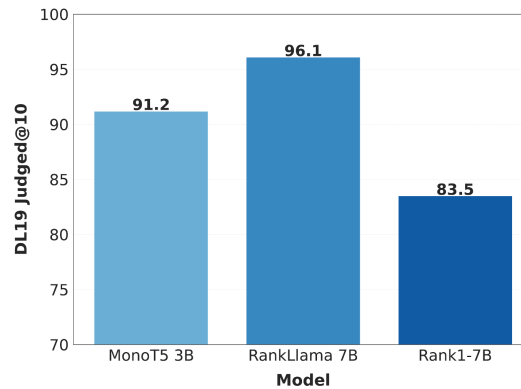
(Timestamp: 00:20:33)

Orion shares a story about evaluating on older, widely-used datasets.

Results

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

Judged@10 is significant less!



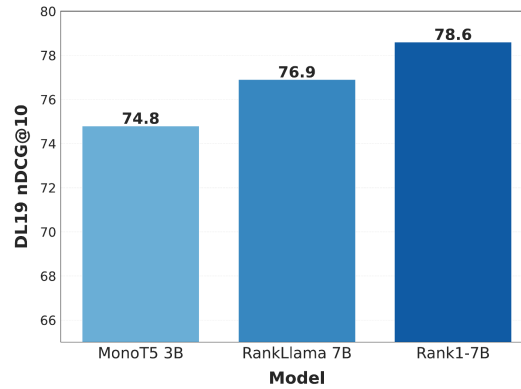
60

(*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. Initial scores showed Rank1 performing worse than expected, below models like RankLLaMA and MonoT5.

Results

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



61

(*Timestamp: 00:21:31*)

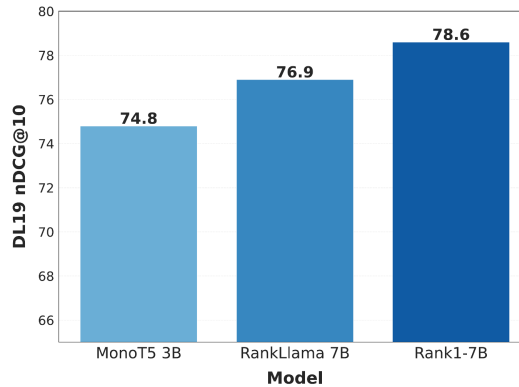
The research team manually re-judged all previously unjudged documents retrieved by their models. After re-judging, Rank1's score increased significantly, making it the top-performing model.

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 (DL19 was created before BERT) as they may not be equipped to measure modern model capabilities.

Summary

- Using test-time compute creates **promptable and reasoning rerankers** - no RL needed!
- It is slower, but much more powerful than previous approaches
- Only trained on general web data — training on in-domain data is likely to see huge gains

62

(*Timestamp: 00:22:05*)

The takeaway is that test-time compute (reasoning) allows for creating promptable and reasoning rerankers using simple supervised fine-tuning, without complex reinforcement learning. These reasoning rerankers are slower than traditional methods but vastly more powerful. The performance gains shown were achieved by training only on general web data. Fine-tuning on in-domain data would likely unlock more significant improvements.



Promptriever
fast embedder



Rank1
strong but slow

63

(Timestamp: 00:22:33)

This slide recaps the two models: Promptriever is fast, while Rank1 is strong but slow.

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.

What does this mean?



New retrievers **benefit**
from advances in LLMs

Instruction-based search:
any query you can type -
they can search



All models are **open-data/open-source**

oweller2@jh.edu

65

(*Timestamp: 00:22:56*)

What are the practical implications? 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. This enables instruction-based search, meaning any query a user can type, no matter how complex, can be understood and executed by the search system.

Orion concludes by emphasizing that all models and data from his research are open-source and available.

3.2 Q&A Session

- **How is Promptriever operationalized for queries vs. documents?**
 - (*Timestamp: 23:45*) The instruction is only applied to the query at inference time. The documents are pre-processed into embeddings without any instruction. This way, you can batch-process your entire corpus once, and then at query time, you append the user's instruction to their query to generate a single, instruction-aware query embedding for the search.
- **Can this instruction-based approach be used for cross-encoders (rerankers) too?**
 - (*Timestamp: 26:04*) Yes, absolutely. Orion mentions they have other work that explores this, and the concepts are applicable to rerankers as well. The [paper for the FollowIR benchmark](#), for example, includes work on instruction-based rerankers.
- **Who provides the meta-instructions for search? Humans or LLMs?**
 - (*Timestamp: 26:32*) Both are possible and interesting. For a “deep research” system, an LLM agent could generate precise, detailed instructions to guide the retrieval process. For end-user applications, a “power user” could type in these complex instructions directly to get more fine-grained control over their search results.
- **How does Rank1 compare to frontier reasoning models like OpenAI's?**
 - (*Timestamp: 28:04*) There is still a performance gap. On some benchmarks, a model like OpenAI's o3 might score

around 75, while the 7B parameter Rank1 model scores around 69. However, Rank1 is significantly smaller (7B vs. a much larger frontier model), faster, and fully open-source, making it ideal for applications with private data or where cost and latency are concerns.

- **How easy is it to train Rank1 on a custom dataset?**
 - (*Timestamp: 30:30*) It's surprisingly easy. The training process uses a standard supervised fine-tuning approach (predict-the-next-token loss) on reasoning traces. The [Rank1 paper](#) notes that the model generalizes remarkably well even without in-domain training, but fine-tuning on a specific dataset is straightforward and would likely lead to large performance gains.
- **Why does supervised fine-tuning (SFT) work for a reasoning model instead of reinforcement learning (RL)?**
 - (*Timestamp: 31:32*) The model learns to reason effectively through **distillation**, a process where it is trained on the reasoning chains generated by a more powerful model (in this case, Deepseek's R1). By learning to mimic the step-by-step "thought process" of the stronger model, it acquires reasoning abilities using a simple and stable supervised fine-tuning objective. This is so effective that it removes the need for more complex RL techniques. Orion speculates this is why major companies have stopped exposing the full reasoning chains of their models, since they are incredibly valuable as training data.

3.3 Video

Here is the full video:

AI Evals Course: 25% off at bit.ly/evals-ai

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

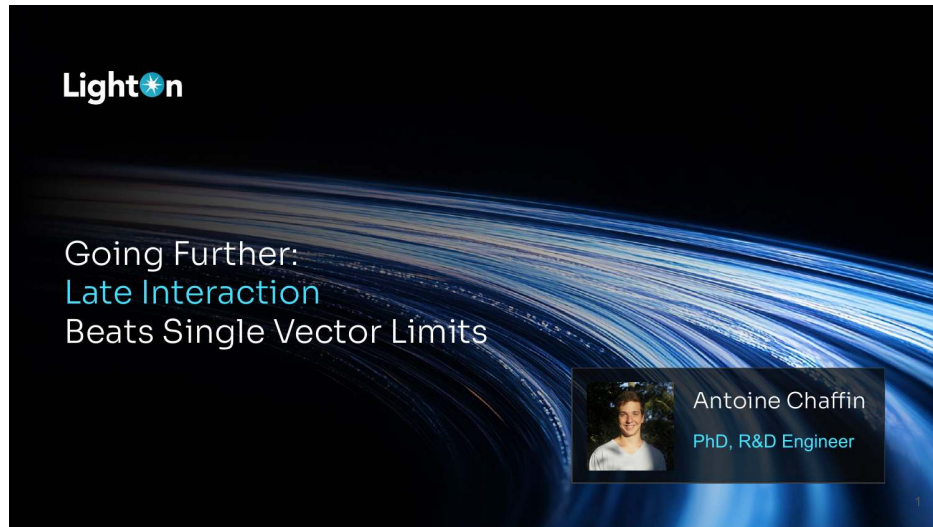
4 Late Interaction Models For RAG

As part of our [LLM Evals course](#), I hosted [Antoine Chaffin](#), a researcher at LightOn, for the fourth part of our 5-part mini-series on evaluating and optimizing RAG. Antoine is a research engineer who has 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.

Below is an annotated version of the presentation, with timestamped links for each slide.

4.1 Annotated Presentation



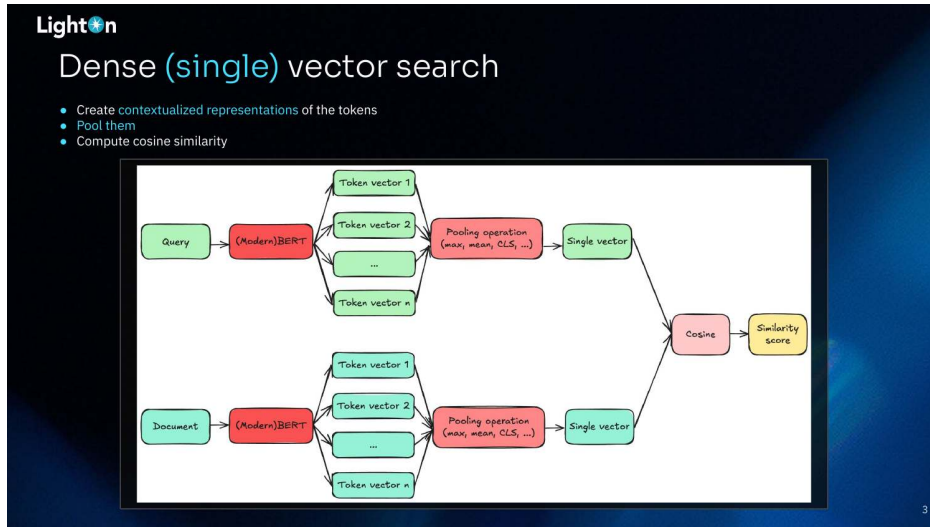
(*Timestamp: 00:00:05*)

The title slide for Antoine's talk, "Going Further: Late Interaction Beats Single Vector Limits."



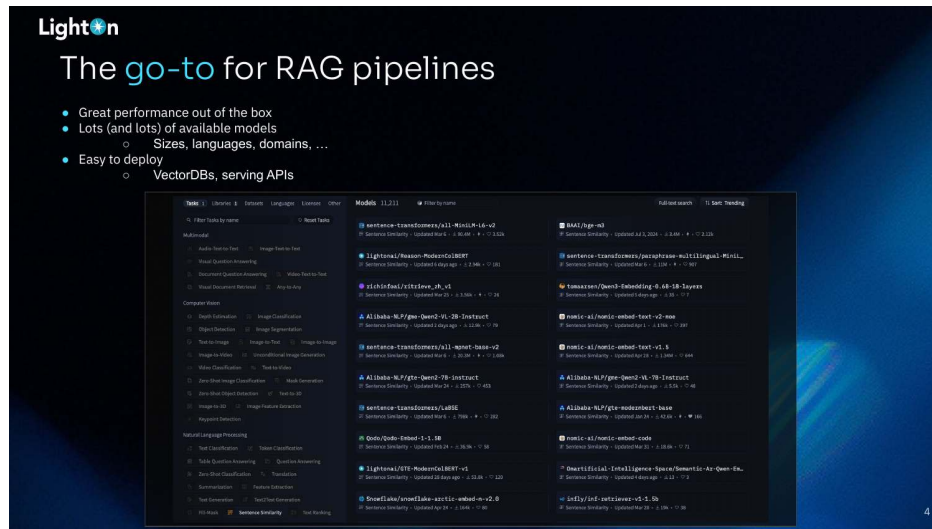
(*Timestamp: 00:00:32*)

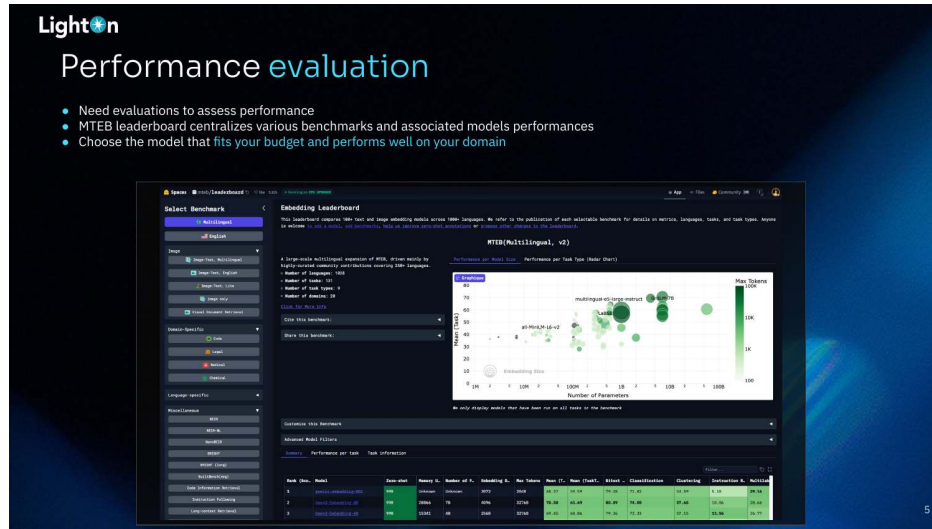
Antoine introduces himself, highlighting his background as an R&D engineer at LightOn with a Ph.D. in multimodal misinformation detection. His work focuses on information retrieval, especially with encoders and late interaction models, which led to his co-creation of ModernBERT and the PyLate library. He also mentions his work on OCR-free RAG pipelines and his active presence on Twitter, where he discusses these topics.



(Timestamp: 00:01:40)

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.





(Timestamp: 00:03:54)

Performance evaluation is crucial for selecting the right model. The MTEB (Massive Text Embedding Benchmark) leaderboard is a valuable resource that centralizes results from various benchmarks, allowing practitioners to compare models and choose one that fits their budget and domain requirements.

Lighton

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!**

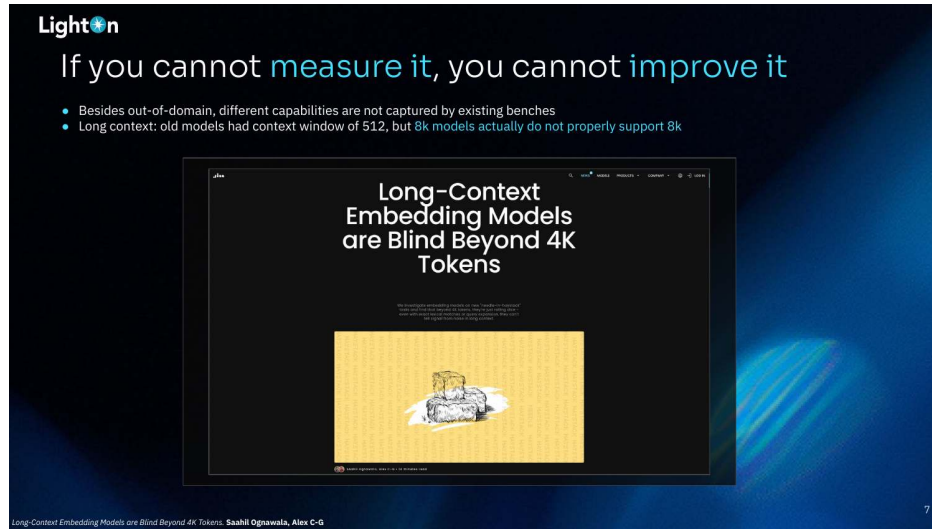
BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models

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.



(*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.

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.
			Passkey	Needle	NQA	QMS	SFD	WQA	
512 Context Models									
ES _{base} (Wang et al., 2022)	110M	512	38.0	28.5	25.3	23.8	74.7	55.8	41.0
ES-RoPE _{base}	110M	512	38.5	31.5	24.6	23.2	66.6	58.8	40.5
GTR _{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
Contriever (Izacard 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
≥ 4k Context Models									
ES-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
Nomic-V1 (Nusshbaum 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
Our Extended Models									
ES _{base} + Tuning (4k)	110M	4,096	67.3	41.5	30.4	35.7	95.2	69.2	56.6
ES-RoPE _{base} + SelfExtend (4k)	110M	4,096	73.5	53.5	32.3	39.1	91.9	74.6	60.8
ES-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

8

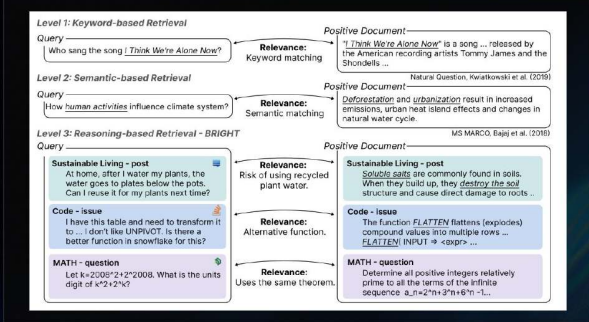
(Timestamp: 00:06:24)

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.

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



Level 1: Keyword-based Retrieval

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

Relevance: Keyword matching

Positive Document: "Think We're Alone Now" is a song ... released by the American recording artists Tommy James 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. MS MARCO. Rajj et al. (2018)

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: Sloppy sa 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.

Code - issue: The function FLATTEN flattens (explodes) compound values into multiple rows ... FLATTEN INPUT => <expr> ...

MATH - question: Let $k=2009^2+2^2 \cdot 2008$. What is the units digit of k^2+2^2k ?

Relevance: Uses the same theorem.

MATH - question: Determine all positive integers relatively prime to all the terms of the infinite sequence $a_n=2^n n+3^n n+6^n n-1$...

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. These tasks are challenging for current models.

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

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, AoPS, 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	AoPS	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
Int-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
Int-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
GrailM	25.8	32.5	18.9	19.8	17.1	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	16.3	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. This highlights the difficulty of reasoning-based retrieval for current systems.

118

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

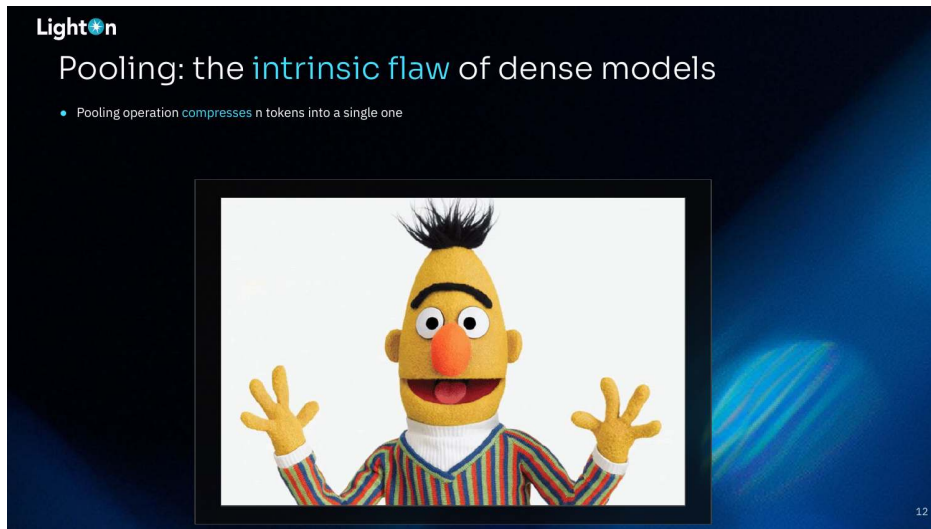
	StackExchange					Coding		Theorem-based			Avg.		
	Bin.	Earth.	Econ.	Psy.	Rob.	Stack.	Soc.	Leet.	Proty	AdPS		TheoQ.	TheoT.
BM25	18.9	27.2	14.9	12.5	13.6	18.4	15.0	24.4	7.9	6.2	10.4	4.9	14.5
Sparse models													
Open-source models (<1B)													
BGE	11.7	24.6	16.6	17.5	11.7	10.8	13.3	26.7	5.7	6.0	13.0	6.9	13.7
Intn-L	15.2	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	15.1	20.4	16.6	22.7	8.2	11.0	15.3	26.6	7.0	5.3	20.0	10.8	14.9
Open-source models (>1B)													
E5	18.6	26.0	15.8	18.8	16.3	11.2	18.1	28.7	4.9	7.1	26.1	20.8	17.9
SFR	19.1	26.7	17.8	19.0	16.3	14.4	10.2	27.4	2.0	7.4	24.3	20.0	18.5
Intn-XL	21.6	34.3	22.4	27.4	18.2	21.2	10.1	27.5	5.0	8.5	15.6	5.9	18.9
CoE-M	26.3	32.3	18.9	19.8	17.1	11.6	17.8	20.9	22.0	6.8	25.2	21.2	21.0
Quora	30.6	36.4	17.8	24.6	13.2	22.2	14.8	25.5	9.9	14.4	27.8	32.9	22.5
Proprietary models													
Cohere	18.7	28.4	20.1	21.6	18.3	18.3	17.6	26.8	1.9	6.3	15.7	7.2	16.6
OpenAI	25.1	26.7	19.7	22.6	12.8	14.5	20.6	21.6	2.4	8.5	23.5	11.7	17.9
Voyage	23.1	35.4	19.9	24.9	10.8	16.8	15.4	30.6	1.5	7.5	22.4	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 Ji Shi, Niklas Muenninghoff, Hao-yu Wang, Haisu Liu, Quan Shi, Zachary S. Siegel, Michael Tang, Ruosi 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.



(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

Dense Model

- Learns to compress based on training data
- Single Vector Output

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

13

(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), where the learned notion of similarity is no longer relevant.

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"

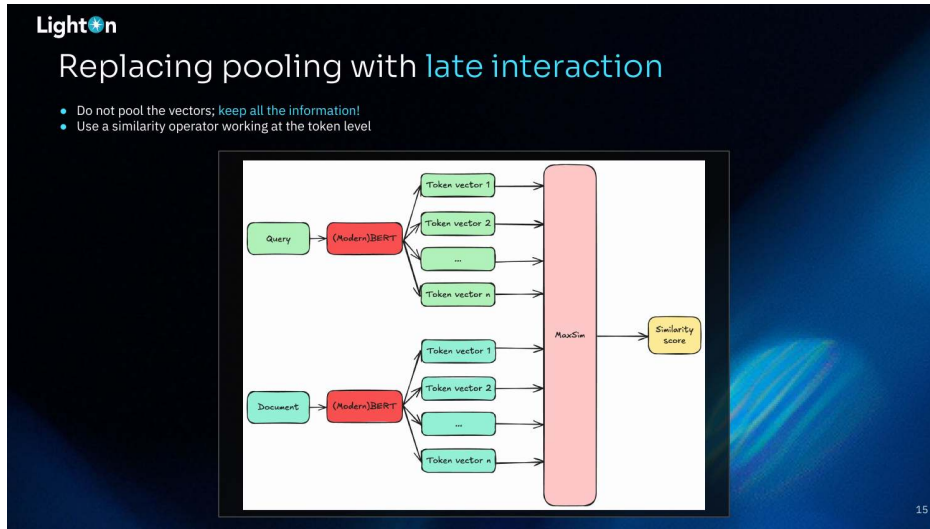
"Titanic features **Leonardo** **DiCaprio** in a romantic **disaster** film..."

✓ No pooling/compression
✗ Exact matching

14

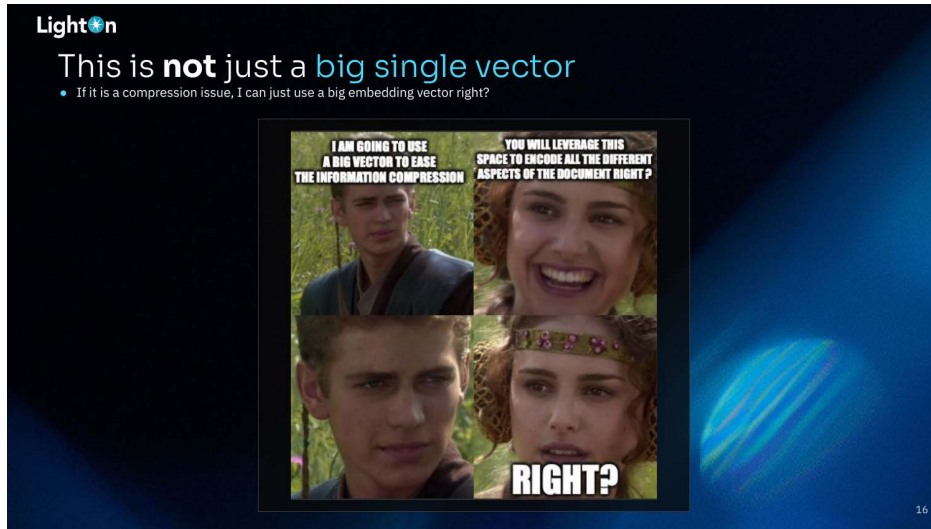
(Timestamp: 00:10:42)

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.



(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.



(Timestamp: 00:12:17)

This meme challenges the idea that using a bigger single vector can solve the information compression problem. While a larger vector can hold more information, it doesn't address the fundamental issue of conflicting signals when multiple distinct concepts are forced into one representation.

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)

Dense (Single Vector)

"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."

Query: "Leonardo DiCaprio movies" Query: "romantic disaster films"

↓ POOLING ↓

Single Compressed Vector

CONFLICTED REPRESENTATION

All information forced into one representation

⚠ Actor and plot signals interfere with each other

Late Interaction

"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."

Query: "Leonardo DiCaprio movies" Query: "romantic disaster films"

↓ NO POOLING ↓

Token-Level Representations

Titanic is masterpiece Leonardo DiCaprio Kate
romantic storyline ship disaster love

MaxSim Operator:

- Actor query matches specific actor tokens
- Plot query matches specific plot tokens
- 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.



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: Optimizing Multi-Vector Retrievers to Create State-of-the-Art Japanese Retrievers with Constrained Resources. Benjamin Clauß
ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. Keshav Santhanam, Omar Khattab, Jon Saad-Faloon, Christopher Potts, Mital Zaharia
ColBERT-XM: A Modular Multi-Vector Representation Model for Zero-Shot Multilingual Information Retrieval. Antoine Louis, Vageesh Saxena, Gijb van Dijk, Gerassimos Spanaklis

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. This highlights its impressive generalization capabilities for long-context retrieval.

Lighton

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	
SGE	13.71
Inst-L	14.21
SGE-07	14.9
> 1B GDS	
ES	17.01
SPR	16.3
Inst-M	16.89
Inst-LM	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)	22.04

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. This demonstrates the power of the late-interaction architecture for complex retrieval tasks.



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. This significant gap underscores the effectiveness of late interaction for challenging, reasoning-intensive retrieval.

Lightn

Interpretability: a nice little bonus

- Granular matching: understand why the document matched
- Identify sub-chunk of the document that matches



The screenshot shows the ColBERT web interface. At the top, it says 'ColBERT query-passage scoring interpretability'. Below that, there's a 'Query (max 50 tokens)' field with the text 'Effects of climate change on marine ecosystems'. Underneath is a 'Passage (max 500 tokens)' field with a longer text about climate change impacts on marine ecosystems. A green button labeled 'Run ColBERT scoring for query - passage' is visible. Below the button, it shows 'MaxSim Score: 27.80' and 'Estimated Relevance: 85.17%'. The 'Contextualised Highlights' section shows the same passage text with specific words highlighted in red: 'marine ecosystems', 'rising temperatures', 'ocean acidification', 'altered precipitation patterns', 'shifts in the distribution and behavior of marine species', and 'underwater ecosystems'. The Vespa logo is at the bottom right of the interface.

22

(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.

Lightn

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!

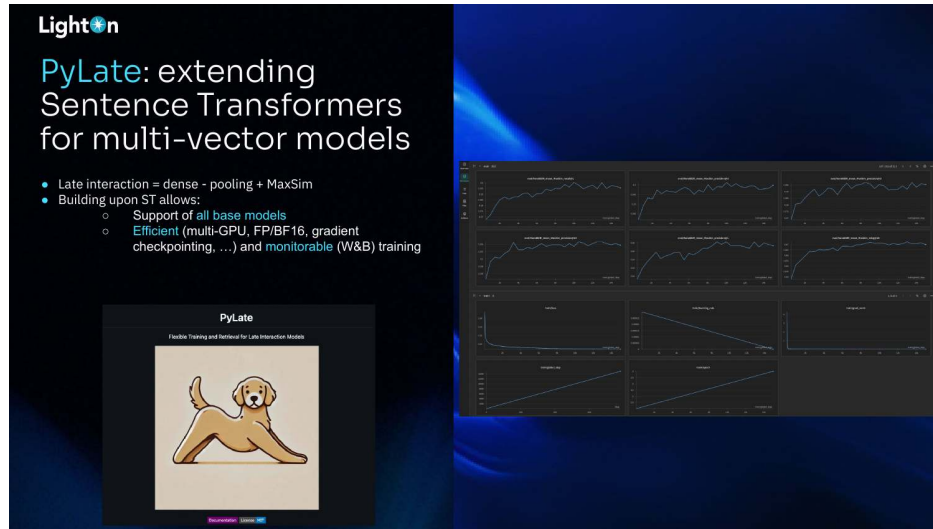


23

(Timestamp: 00:17:42)

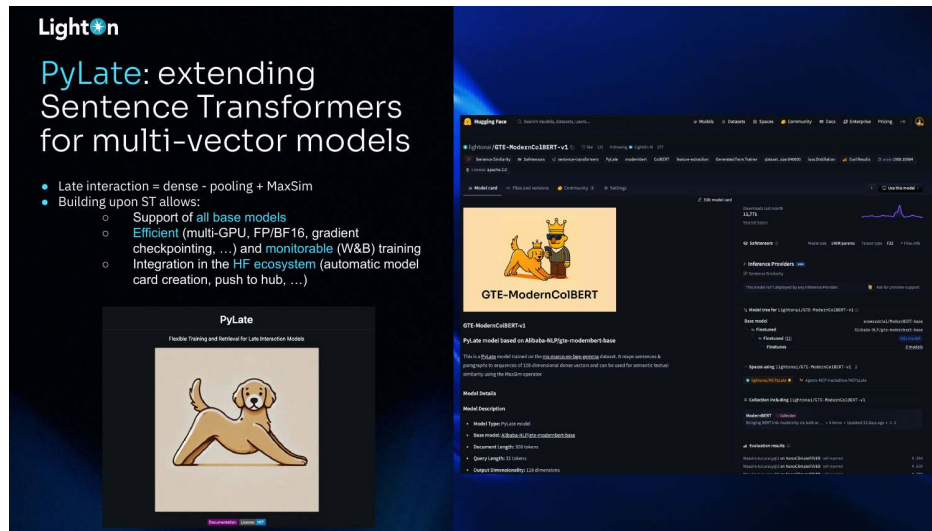
Despite their advantages, dense models are still mainstream. Antoine attributes this to three main factors:

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.



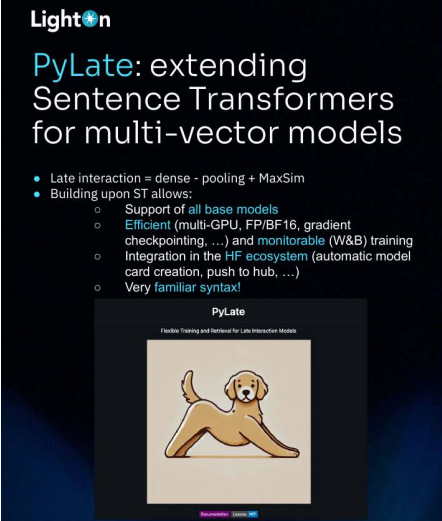
(Timestamp: 00:18:43)

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. This allows for efficient, monitorable training (multi-GPU, FP/BF16, W&B) and support for all base models.



(Timestamp: 00:19:43)

PyLate is well-integrated with the Hugging Face ecosystem. This allows for easy sharing of models on the Hub and includes features like automatic model card creation, making it simple to document and distribute your trained late-interaction models.



Light 

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, FP/BF16, gradient checkpointing, ...) and **monitorable** (W&B) training
 - Integration in the **HF ecosystem** (automatic model card creation, push to hub, ...)
 - Very **familiar syntax**!

PyLate

Flexible Training and Inference for Late Interaction Models



PyLate: Training and Inference

```

1 from datasets import load_dataset
2 from sentence_transformers import (
3     SentenceTransformerTrainer,
4     SentenceTransformerTrainingArguments,
5 )
6
7 from pylate import evaluator, losses, models, utils
8
9 # Define model architecture for contrastive learning
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-Late-Base-Model"
16 model_dir = "models/late-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, sr_path=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_shuffle=True)
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 my_evaluator = evaluator.SentenceTransformerEvaluator()
33
34 # Configure the training arguments (e.g., batch size, evaluation strategy, logging setup)
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 MPS
41     bf16=True, # Set to True if you have a GPU that supports BF16
42     run_name=run_name, # Will be used to add it to wandb, if supported
43     logging_level=logging.INFO,
44 )
45
46 # Initialize the trainer for the contrastive training
47 trainer = SentenceTransformerTrainer(
48     model=model,
49     args=args,
50     train_dataset=train_dataset,
51     eval_dataset=eval_dataset,
52     loss_function=train_loss,
53     data_loader_args=utils.CustomDataLoader(model, train_loss),
54 )
55
56 # Start the training process
57 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="Evaluates a model")
8     parser.add_argument(
9         "-i", "--index",
10         default="ir_datasets",
11         help="Name of the dataset to evaluate on (default: 'ir_datasets')",
12     )
13     args = parser.parse_args()
14     dataset_name = args.index
15     model_name = "lightn/ir_datasets-beir-v1"
16     model = models.IrDataset(
17         model_name, args.index
18     )
19     documents, queries, qrels = evaluation.load_beir(
20         dataset_name=dataset_name, split="test"
21     )
22     index = indexes.PLAID(
23         index_name=f"lightn_{dataset_name}_{model_name}_plaid",
24     )
25     retriever = retrieval.Collate(index=index)
26
27     documents_embeddings = model.encode(
28         documents[dataset_name] for document in documents,
29         batch_size=1000,
30         in_memory=True,
31         show_progress_bar=True,
32     )
33
34     index.add_documents(
35         documents, documents_embeddings
36     )
37     documents_embeddings = model.encode(
38         queries[dataset_name] for query in queries,
39         batch_size=100,
40         show_progress_bar=True,
41         in_memory=True,
42     )
43
44     scores = retriever.retrieve_queries(
45         documents_embeddings, queries_embeddings, k=100
46     )
47     evaluation_metrics = evaluation.evaluate(
48         documents_embeddings,
49         queries_embeddings,
50         queries_list=queries.keys(),
51         metrics=["map", "recall@10", "recall@20", "recall@50", "recall@100"],
52     )
53     print(evaluation_metrics)

```

(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. The system is compatible with standard data formats (e.g., MTEB, BEIR), so you can evaluate on existing benchmarks or your own data.

Lightn

What are the future avenues?

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

16-bit Dense Vector	Index Size
2-bit PLAID Index	345 MB
pool 2	760 MB
pool 3	388 MB
pool 4	260 MB
pool 5	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	fiqa	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. Pooling ₂	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. Pooling ₆	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)

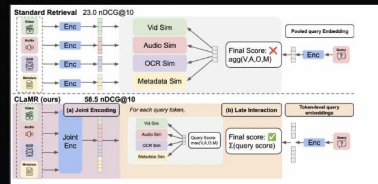
One future research avenue is to reduce the storage cost of multi-vector models. Techniques like hierarchical pooling and quantization are being explored to find the optimal trade-off between index size and performance. The goal is to make the footprint of multi-vector indexes comparable to that of single-vector representations without sacrificing much 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 Glentzer, Saba Shams, Mohammad Kalin Akran, 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, consistently outperforming single-vector approaches.



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
{ziwei ji, 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, as shown in the paper “Efficient Document Ranking with Learnable Late Interactions” from Google, presents an opportunity to further improve the performance of these models.

Lightn 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)

Antoine concludes by summarizing the key takeaways. Late interaction models overcome the intrinsic limitations of single-vector search and are well-suited for modern, real-world use cases (out-of-domain, long context, reasoning-intensive). With growing ecosystem support and tools like PyLate, it's easier than ever to experiment with these models. He encourages the audience to try existing models on their own data and to train their own specialized models using the provided resources.

4.2 Q&A Session

- **(25:52) Why aren't late-interaction models mainstream yet, given their advantages?** Antoine believes it's still early days. The tools and VectorDB support have only recently matured.

It takes time for new technologies to be adopted, especially when it requires changes to production systems. He notes that many use cases don't require scaling to millions of documents, and for those that do, modern indexes make it feasible. As more models become available for different languages and domains, he expects adoption to grow. Regarding latency, while late-interaction might be slightly slower, the performance gains often outweigh the minor latency increase, which is often not the bottleneck in complex RAG pipelines.

- **(31:04) If you fine-tune both a dense vector model and a late-interaction model on the same in-domain data, does the performance gap still hold?** Yes, the performance gap still exists, even in-domain. Antoine points to the comparison on the BRIGHT benchmark, where a late-interaction model significantly outperformed a single-vector model with the same backbone and training data. He also suggests that fine-tuning a late-interaction model is easier and more stable because there's less risk of the model's knowledge "collapsing" onto the new training distribution, as the updates are more granular.
- **(33:20) How easy is it to get started and fine-tune with PyLate? Are there any tips?** It's very straightforward, especially for those familiar with Sentence Transformers. The boilerplate code is nearly identical. Antoine recommends using the in-training evaluation feature to monitor performance, which is particularly helpful when sweeping hyperparameters. He also mentioned that the training process is generally more stable and converges faster than with single-vector models. The PyLate documentation and repository contain boilerplates and more detailed guidance.
- **(34:22) What are some common mistakes or points of confusion for people moving from single-vector to late-interaction models?** Antoine hasn't seen many major pitfalls.

He says if you can train a single-vector model, you can train a late-interaction model with PyLate. The common advice still applies: tune the temperature for contrastive loss, use a large batch size, etc. The documentation covers most of these common issues, and he encourages users to open issues or reach out on Twitter for help.

4.3 Video

Here is the full video:

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

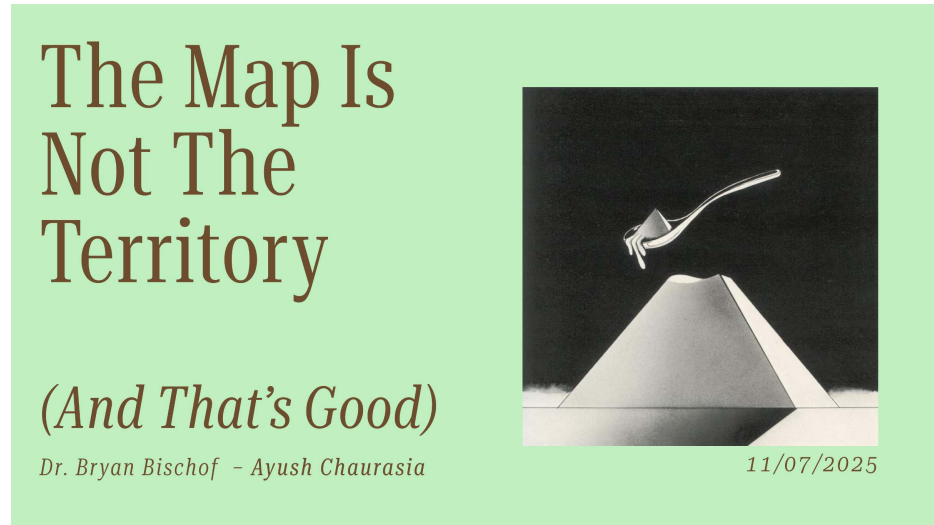
5 RAG with Multiple Representations

As part of our [LLM Evals course](#), I hosted [Bryan Bischof](#) and [Ayush Chaurasia](#) for the final part of our 5-part mini-series on evaluating and optimizing RAG. They 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, a concept they demonstrate with semantic dot art.

Below is an annotated version of their presentation, with timestamped links for each slide.

5.1 Annotated Presentation

5.1.1 The Map is Not the Territory



(*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.

5.1.2 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.

Part I

“_____ RAG”

Leave Me Alone

- + *No-De RAG*
- + *Agentic RAG*
- + *Hybrid RAG*
- + *Graph RAG*
- + *Multi-Modal RAG*

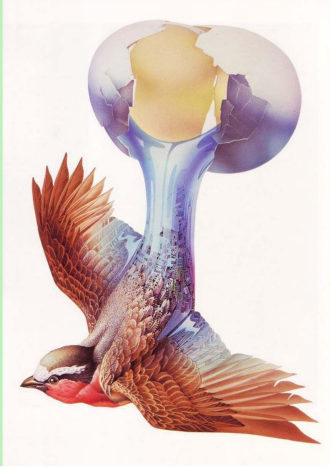
(Timestamp: 06:45)

All these techniques are different approaches to the same problem and could have been invented from first principles.

5.1.3 A First-Principles View of RAG

Advanced RAG techniques can be reframed as core engineering pipelines.

Part II



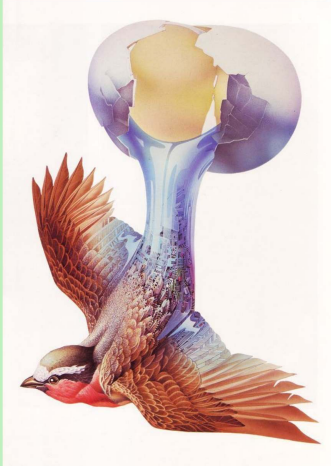
ONE

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.

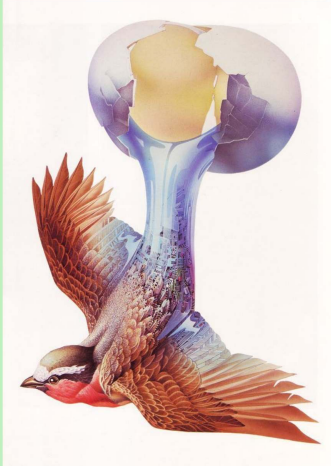
TWO

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.

5.1.4 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.

5.1.5 Practical Application: Curving Space

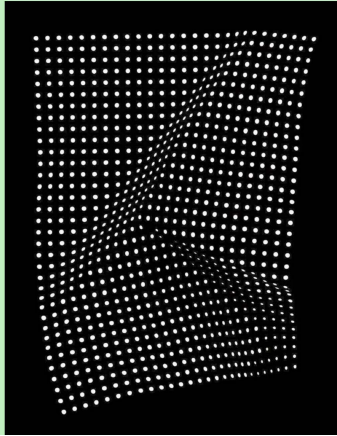
“Curving space” means shaping data representations and indices to improve search.

Part III

Let's retrieve Financial Documents

ONE

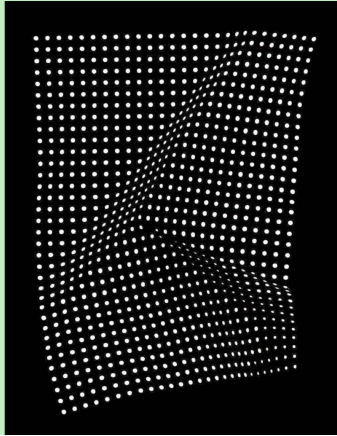
What are the different concepts one may construct from these docs? Summaries?
Tables of Data? Lists of Named Entities?
Form types/categories? What else?



(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?

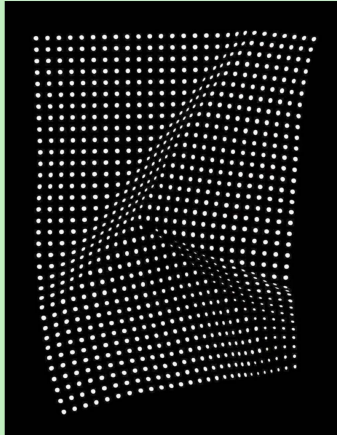
TWO

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.

5.1.6 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.



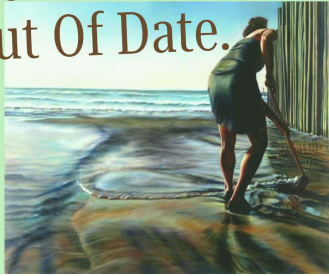
(*Timestamp: 19:04*)

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.

5.1.7 Representations Must Evolve

Part V

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

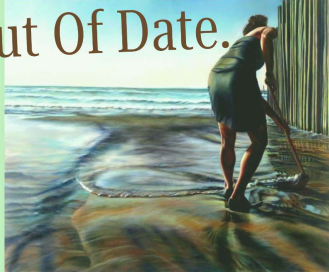


(Timestamp: 20:51)

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.

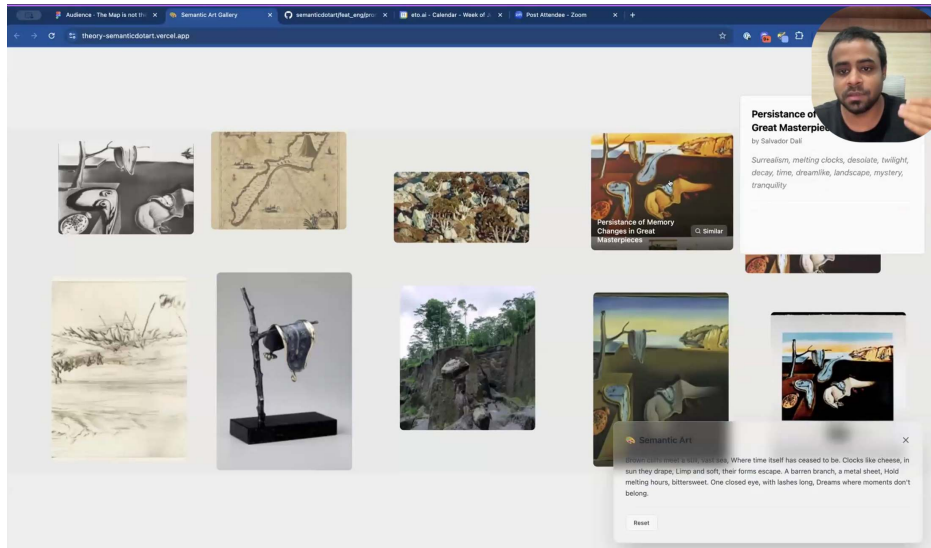
(*Timestamp: 22:49*)

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.

5.1.8 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.

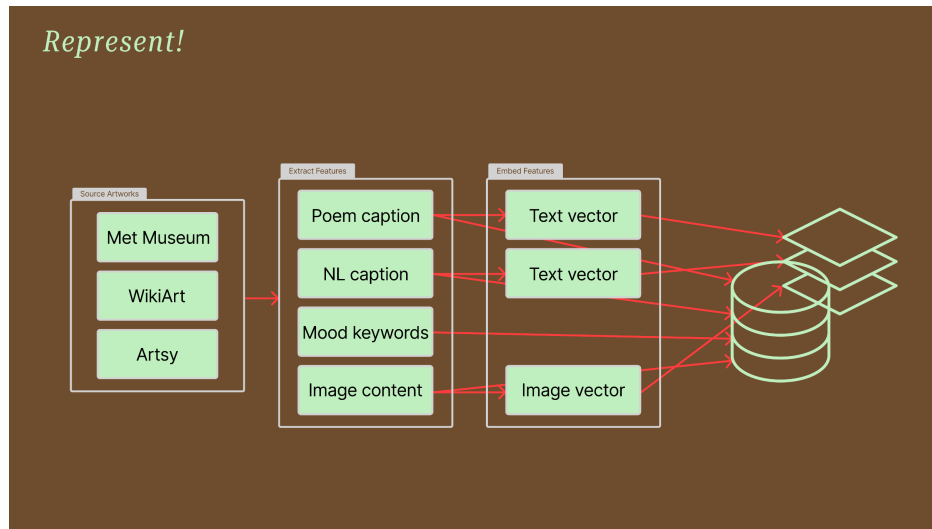


A user can search for art using different “maps” of the data. For example, they can search with a literal description (“multiple clocks melting in a desert”), a poetic description, or even a similar image. The system retrieves not only the original artwork but also derivative pieces and other thematically 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.

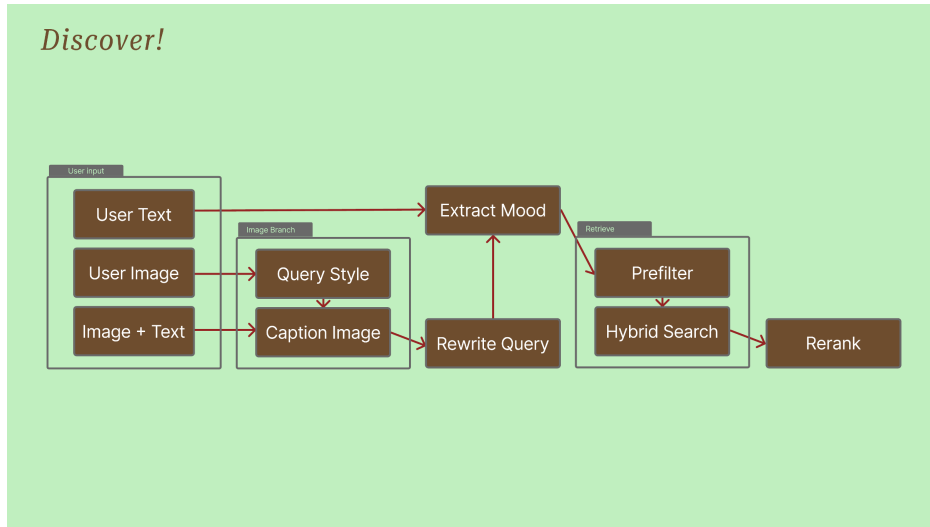
5.1.9 System Diagrams

Ayush then described the system he demoed in more detail.



(*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.

5.2 Q&A Session

- **Why do people often think there’s only one “map” (representation) for their data?** (*Timestamp: 32:41*) People often seek a single “best” representation. When it fails, the instinct is to fix it rather than augment it with new, different representations. A better mindset for retrieval is to build and use

many specialized “maps” of your data, similar to owning different bicycles for different terrains. Modern models make this easier by helping route queries to the correct map.

- **How do Reasoning Models and Late-Interaction Models (like ColBERT) fit into this “multiple maps” framework?** (*Timestamp: 35:45*) They fit well. **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.
- **What is routing in this context, and is it just a classifier?** (*Timestamp: 42:31*) 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 (e.g., via a tool call). As the number of routes grows, a dedicated, trained classifier becomes more appropriate. LLMs are good for proving the concept of semantic routing, but for scalable classification, a fine-tuned model is a better approach.

5.3 Video

Here is the full video:

<https://youtu.be/hf9B3knU9vc>

6 Context Rot

As part of our [LLM Evals course](#), I hosted Kelly Hong, a researcher at Chroma, to discuss [her research](#) on “Context Rot.” Despite the narrative that RAG is dead because of large context windows (e.g. 1 million tokens), Kelly’s research shows that performance is not uniform. As you add more information, models become increasingly unreliable, even on simple tasks. This phenomenon, which Kelly coined as “Context Rot,” is worth paying attention to if you are building AI applications. Kelly’s talk breaks down the experiments that uncovered this issue and highlights why thoughtful context engineering and retrieval is more important than ever.

Below is an annotated version of her presentation.

6.1 Annotated Presentation

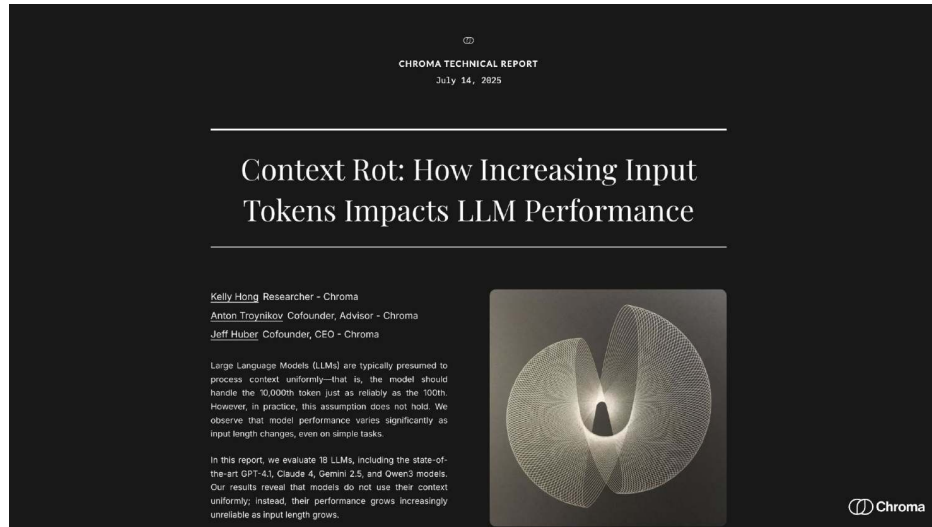


Figure 6.1: Title Slide

(*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.

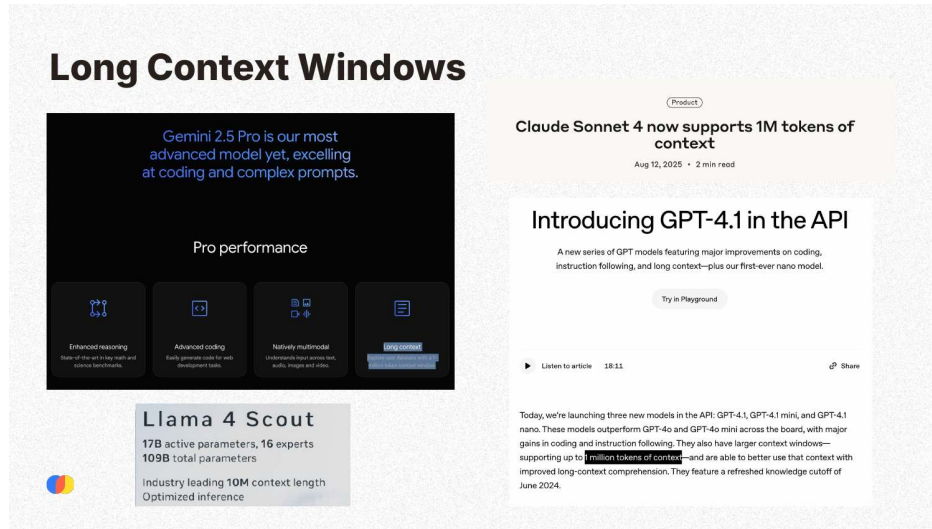


Figure 6.2: The Rise of Long Context Windows

(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

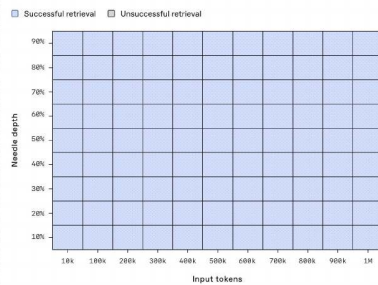


Figure 6.3: The Common Assumption: More Context is Better

(*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.

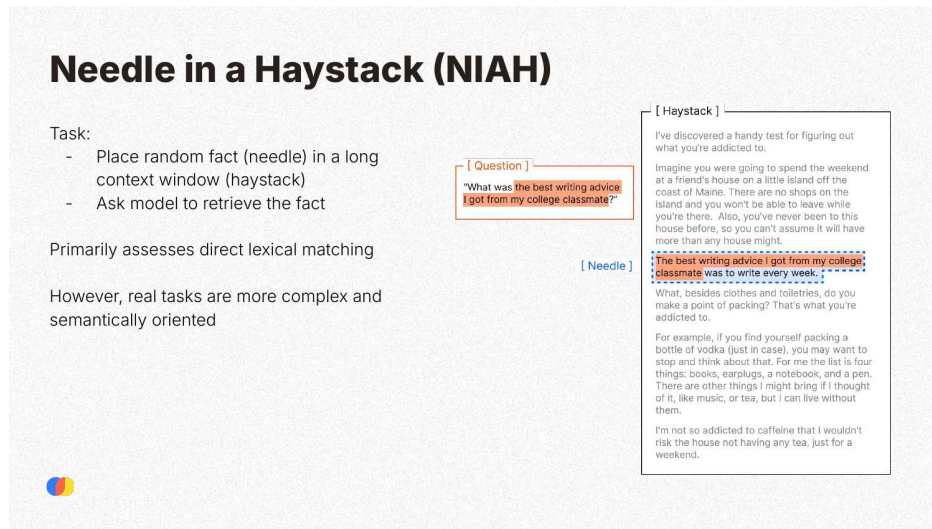


Figure 6.4: Explaining the “Needle in a Haystack” (NIAH) Benchmark

(*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.

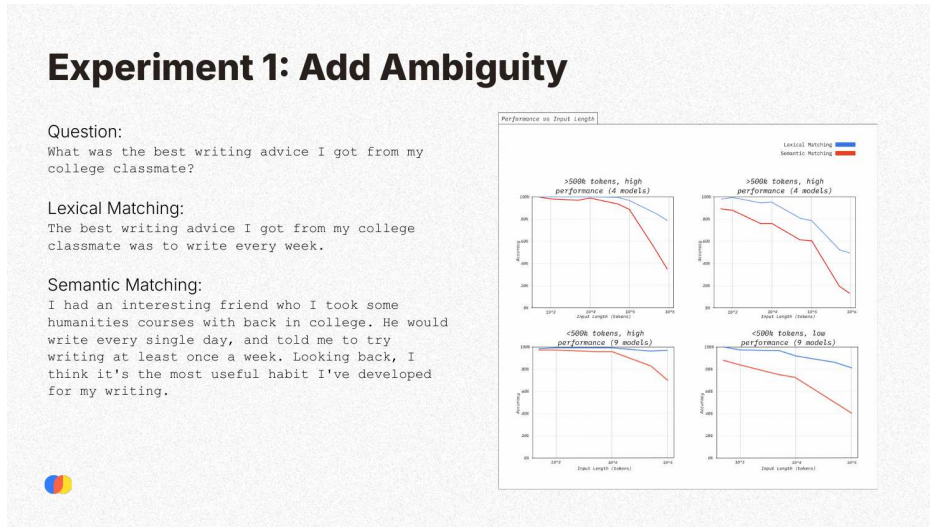


Figure 6.5: Experiment 1: Adding Ambiguity (Semantic vs. Lexical Matching)

(*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.

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.

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

[Distractor]
What, besides clothes and toiletries, do you make a point of packing? That's what you're addicted to.
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.

[Irrelevant Content]

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.

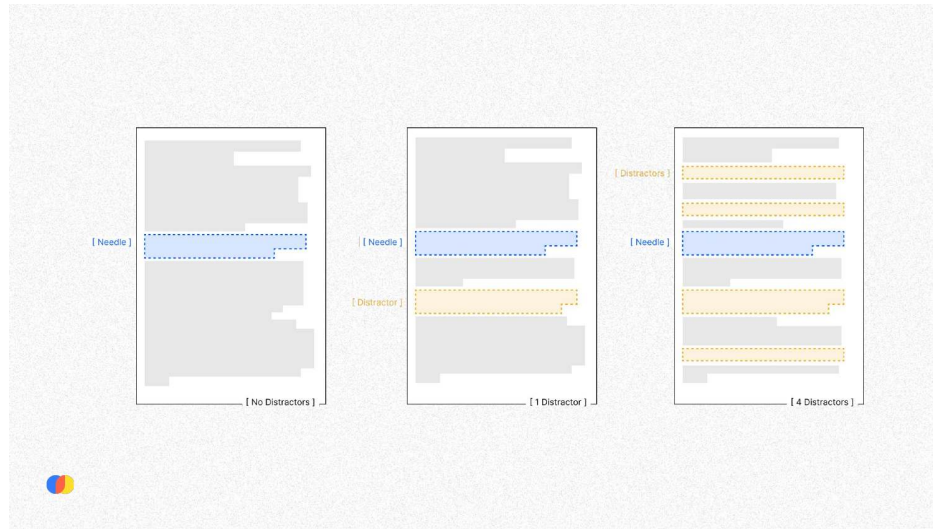


Figure 6.8: Visualizing the Distractor Setup

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.

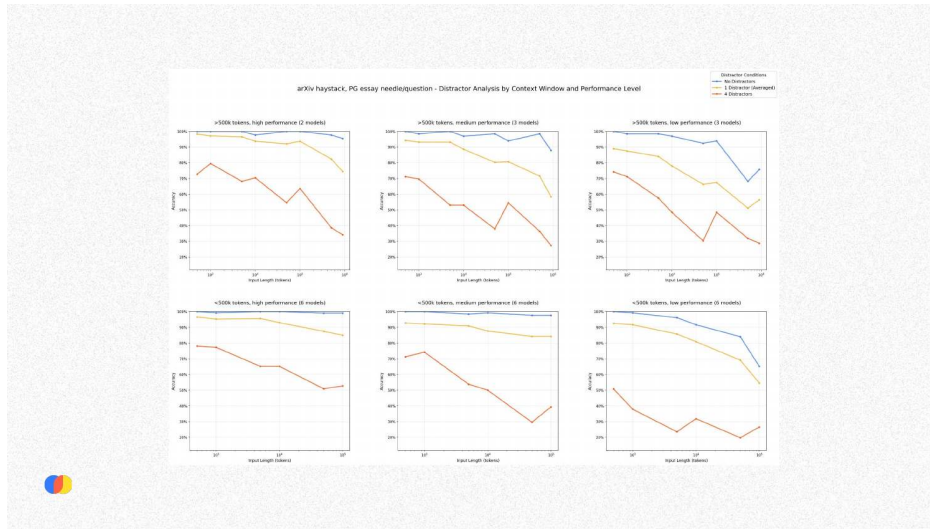


Figure 6.9: Results: Performance Degrades with More Distractors

(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)"

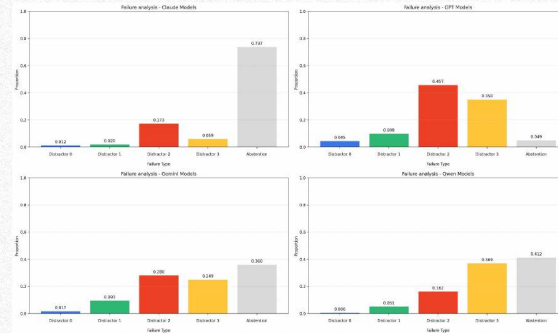


Figure 6.11: Analyzing Failure Modes: Model Hallucinations vs. Abstention

(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.

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

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.

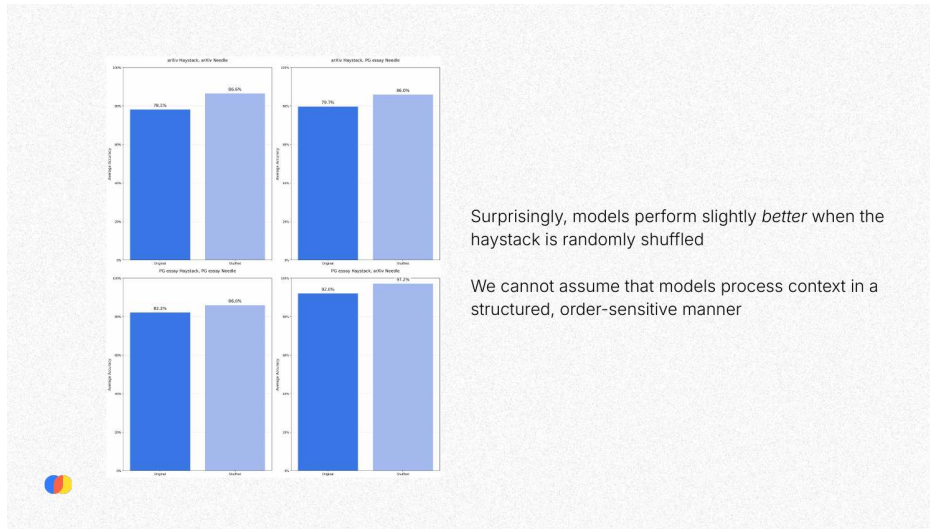


Figure 6.13: Surprising Results: Models Perform Better on Shuffled Context

(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.

Experiment 4: Conversational Memory

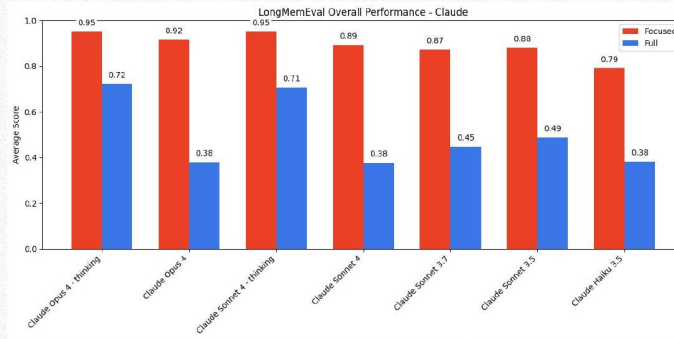


Figure 6.14: Experiment 4: Conversational Memory

(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.

Experiment 5: Text Replication

Simply replicate the following text, output the exact same text: apple apple
apple apple apples apple apple apple apple apple apple apple apple
apple apple apple apple apple apple apple apple apple

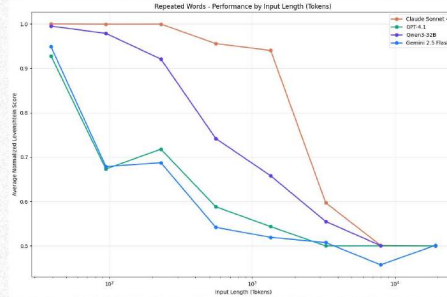


Figure 6.15: Experiment 5: Text Replication Task

(*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.

Takeaways

Models do not have uniform performance across input lengths, even for very simple tasks

Even if you have the right information in context, *how* you present it matters

Context engineering is critical



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.

Context Engineering Example

Use case: coding agent with a long-running task

Naive approach: append conversation history with each tool call/turn, keeping everything in context

- Includes every tool call made and output
- Context grows quickly

Better approach: break down task into subtasks, use subagents for each

- Main “orchestrator” agent creates a plan of subtasks, spawns subagents
- Each subagent has its own context, returns most relevant information to orchestrator agent
- Orchestrator agent keeps the most relevant information



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.

Further Reading

View full report & other research: <https://research.trychroma.com/>

Chroma Research

[14-July-2025] [Context Rot: How Increasing Input Tokens Impacts LLM Performance](#). Kelly Hong, Anton Troynikov, Jeff Huber.

[07-April-2025] [Generative Benchmarking](#). Kelly Hong, Anton Troynikov, Jeff Huber, Morgan McGuire.

[03-July-2024] [Evaluating chunking strategies in retrieval for AI applications](#). Brandon Smith, Anton Troynikov.

[29-May-2024] [Embedding Adapters](#). Suvansh Sanjeev, Anton Troynikov.



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.

6.2 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.

6.3 Video

Here is the full video:

https://youtu.be/3s_N60u0jEY

7 You Don't Need a Graph DB (Probably)

Every so often, a technology captures the industry's attention. Graph databases are having their turn, fueled by a desire to have a powerful way to augment LLMs with your data. The logic seems simple: if your data is connected, you need a graph.

Throughout this series, we've argued that naive single-vector RAG is dead and shown more sophisticated approaches: [better evals](#), [reasoning](#), [late interaction](#) and [multiple representations](#). But sophistication isn't the same as complexity. There's no better example of unnecessary complexity than reaching for a graph database before you've exhausted simpler options.

That's why I hosted a talk with [Jo Kristian Bergum](#). Jo is a 25-year veteran in search and retrieval, with experience at Yahoo and Vespa. He's one of the few who publicly shares this skepticism.

Jo covers when graph databases are overkill and when they make sense.

Below is an annotated version of [his presentation](#).

HORNET

GraphRAG and GraphDB

No single silver bullet to solve retrieval

Jo Kristian Bergum, CEO HORNET.dev

(*Timestamp: 00:00:00*)

7.1 The Hunt for a Silver Bullet

Interest in RAG has exploded since late 2022. More engineers than ever are tackling search problems. But this led to a frantic search for a silver bullet.

First, it was the specialized vector database. Now, with the popularity of a Microsoft paper on “GraphRAG,” the pendulum is swinging toward graph databases.

Jo argues this is flawed thinking. There is no single tool that will solve all your problems. The desire to map a new technique (GraphRAG) to a technology (a graph DB) is a trap many engineers fall into.

Why I'm here?



(*Timestamp: 00:01:38*)

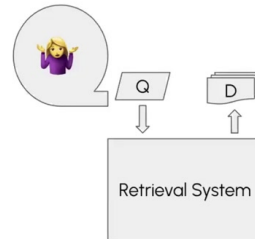
7.2 Do You Even Need RAG?

Before we get to graphs: do you need a retrieval component?

The “R” in RAG stands for retrieval. The classic information retrieval (IR) model involves a user with an information need, a query, a retrieval system, and a ranked list of documents.

What is (information) retrieval (IR)

Information retrieval (IR) is the task of retrieving information that are relevant to an information need, typically expressed as a query.



https://en.wikipedia.org/wiki/Information_retrieval

(*Timestamp: 00:05:08*)

When LLMs had tiny 8k context windows, retrieval was a necessity. But now, models like Gemini can handle 1 million tokens. That's about 3 megabytes of raw text, the equivalent of an old floppy disk.

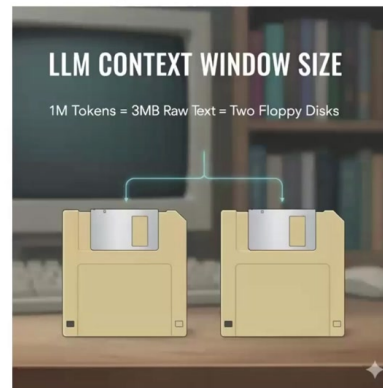
Do we need retrieval at all with long context models?

Many applications: no.

\$3 per million input tokens
(Sonnet 4)

Agentic LLM usage

- Cumulative token usage
- Turns X context size



3.5" floppy disk from the 80s 1.44MB per disk

(*Timestamp: 00:06:15*)

Jo shared an anecdote about a company struggling with a complex RAG pipeline. When he asked how many documents they had, the answer was 300. For that scale, he advised them to feed all the documents into the LLM's context window. It was simpler, cheaper, and more effective than over-engineering a retrieval stack.

If your document set is small enough and your query volume is low, stuffing the context window is a valid strategy.

7.3 Deconstructing GraphRAG

So what is this “GraphRAG” that’s causing all the fuss? The technique, as described in the Microsoft paper, involves a few steps:

- Process the corpus: Use an LLM to read your entire document collection. This is an expensive, offline process.

- Build a knowledge graph: Prompt the LLM to extract entities (nodes) and relationships between them (edges).
- Query the graph: At query time, traverse this graph to find relevant information.

The challenge is *building and maintaining* the graph. A knowledge graph is a collection of triplets: (source_entity, relationship, target_entity). You can store this in a CSV file, a JSON object, or a standard relational database like Postgres.

What is a knowledge graph?

A triplet

nodes = entities

edges = relationships

source_entity, target_entity,
description

```
CREATE TABLE entities (  
  entity_id INT PRIMARY KEY AUTO_INCREMENT,  
  entity_name VARCHAR(255) NOT NULL,  
  entity_type VARCHAR(50), -- e.g., 'PERSON', 'ORGANIZATION', 'LOCATION'  
  entity_description TEXT,  
  -- Each entity is assigned to a community.  
  community_id INT,  
  FOREIGN KEY (community_id) REFERENCES communities(community_id)  
);  
  
-- Table to store the relationships (edges) between entities.  
CREATE TABLE relationships (  
  relationship_id INT PRIMARY KEY AUTO_INCREMENT,  
  source_entity_id INT NOT NULL,  
  target_entity_id INT NOT NULL,  
  relationship_description TEXT,  
  FOREIGN KEY (source_entity_id) REFERENCES entities(entity_id),  
  FOREIGN KEY (target_entity_id) REFERENCES entities(entity_id)  
);
```

(*Timestamp: 00:09:23*)

The hard part is keeping everything accurate. That means either spending tokens on LLM calls, or getting domain experts to help.

7.4 Evaluation

How do you know if adding a knowledge graph, or any new technique, is improving your results? You have to measure it.

This is the step most teams skip. They jump from one new method to the next without a stable benchmark. As [Nandan showed in Part 2](#), you need an evaluation framework *before* you can make good decisions about retrieval techniques. In retrieval, we use metrics like:

- **Mean Reciprocal Rank (MRR):** Measures how high up the list the first relevant document appears.
- **Precision:** Of the documents you returned, how many were relevant?
- **Recall:** Of all possible relevant documents, how many did you find?

To use these metrics, you need an evaluation dataset. This involves taking a sample of real user queries and manually labeling which documents are relevant for each query. This “golden dataset” is your ground truth. It allows you to compare retrieval methods and know if a change is an improvement.

How to evaluate if it's useful

Retrieval quality metrics	Query, Relevant Document, (Optional how relevant)
Mean Reciprocal Rank 1/rank of first relevant hits	.
Precision (nothing but relevant)	You only need a simple triplet file to evaluate retrieval quality for your task
Recall (all the relevant)	

([Timestamp: 00:12:05](#))

7.5 When to Use a Graph Database

Once you have an evaluation framework, you can start asking the right questions. A specialized graph database (like Neo4j) is optimized for fast traversal of graph structures, usually by keeping the graph in memory.

Before you add one to your stack, ask yourself:

- Do I need fast, low-latency graph traversal?
- Is my graph so large that a simpler solution (like Postgres or a file) is too slow?
- Does my use case require complex, multi-hop queries (e.g., finding friends of friends of friends)?

If the answer to these questions isn't a clear "yes," you probably don't need a specialized graph DB. The operational complexity and cost of adding another database to your stack is high. As Jo pointed out, early Facebook ran its massive social graph on MySQL. You can get surprisingly far with general-purpose tools.

What are your bottlenecks for system performance

Do you need sub-ms traversal of graphs?

Anecdotal observation

Does use case require multi-hop traversals? (friends of friend)

Early Facebook used mysql for their friend graph

Versus

- The complexity of adding another DB
- The potential effectiveness?

You don't necessarily need a graph DB to represent a graph, or use graphRAG as one technique

(*Timestamp: 00:16:21*)

7.6 The Graph Hidden in Your Vector Search

If you're using modern vector search, you're already using graphs.

The most common algorithm for approximate nearest neighbor (ANN) search is **HNSW (Hierarchical Navigable Small World)**. HNSW works by building a graph where the nodes are your vectors and the edges connect vectors that are close in the embedding space. Searching for nearest neighbors means traversing this graph.

Jo mentioned the “cluster hypothesis” in information retrieval: documents that are similar to a relevant document are also likely to be relevant.

You can use this with any vector DB:

1. Run your standard vector search for a user's query.
2. Use an LLM or reranker to identify the most relevant results from this initial set.
3. Perform a second vector search, this time looking for documents similar to the relevant documents you just found.

This is a form of graph traversal (finding neighbors of your best results) that can improve recall. You don't need a specialized graph database to do it.

7.7 Key Takeaways

In this series, we've shown you how to move beyond naive RAG: better evals, reasoning, late interaction, multiple representations, context engineering. But the goal was never complexity for its own sake.

Jo's talk reinforces this. There is no silver bullet. Before you adopt a new method like GraphRAG, build a golden dataset to measure if it improves performance ([Part 2](#)). You don't need special tools. A knowledge graph can live in a CSV file or Postgres. Early Facebook ran their social graph on MySQL. Vector search algorithms like HNSW are already graph-based, so you can implement graph-like retrieval strategies without adding new infrastructure.

7.8 Video

Here is the full video:

<https://youtu.be/7kXY-2fYdHI?si=Lbk0LN85S7jA6DZS>

Conclusion

RAG has moved past the simple “embed and search” approach from 2023. The presentations in this book show what that looks like in practice: richer representations, instruction-following retrievers, careful evaluation, and thoughtful infrastructure choices.

Looking Forward

Rather than compressing all information into a single vector, we’re seeing systems that maintain multi-faceted representations, reason about relevance, and combine classical techniques with modern ones.

The tooling is maturing. Libraries like PyLate, frameworks like RAGatouille, and vector databases with advanced retrieval support make these techniques accessible.

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”](#)

- [Kelly Hong](#): “Context Rot”
- [Jo Kristian Bergum](#): “You Don’t Need a Graph DB”

Web version: [here](#)

These talks are from our [AI Evals course](#), which covers LLM evaluation techniques beyond RAG. Readers of this book can use [this 25% discount](#).