Unlocking Web Archives: LLMs, RAG, and the Future of Digital Preservation

Corey Davis, February 2025

Executive Summary

Large language models (LLMs) are poised to transform how research libraries approach digital preservation and access to digital collections more broadly. These systems can enhance workflows in a number of ways, from simplifying the process of keeping policies and documentation up to date, to developing scripts to automate and integrate data processing tasks. Already, libraries are using LLMs to transcribe handwritten documents, extract key entities from text, and create detailed content descriptions—tasks that once required extensive manual effort.

At their best, LLMs enhance efficiency and help bridge the technical skills gap that often hinders our digital preservation efforts. At their worst, they hallucinate and operate as inscrutable black boxes. Retrieval-Augmented Generation (RAG) offers a powerful solution to these challenges, allowing us to enrich LLMs with data from our own collections, bridging the gap between AI-generated insights and authoritative sources. By mapping our digital collections into a high-dimensional vector space, we can transform them into dynamic knowledge bases—ready to chat, connect, and reveal insights through conversational search. But can these systems live up to the hype?

This paper explores the inner workings of LLMs and RAG, examining their inherent strengths and limitations and how these factors influence their potential applications in a research library setting. Two case studies are presented: the testing of WARC-GPT, an open-source tool developed by the Harvard Law Library Innovation Lab that applies RAG to web archives, and the development of a bespoke RAG pipeline, built to address some of WARC-GPT's limitations.

Web ARChive (WARC) files are notoriously difficult to search, requiring specialized knowledge and tools for effective retrieval. Traditional keyword search methods often fail, leading to a frustrating experience for researchers. WARC-GPT provides a fresh approach to these problems, one that could be scaled beyond web archives to other digital collection types. By transforming our digital materials into embeddings within a high-dimensional vector space, we can help users to retrieve semantically similar content through natural language queries, with a state-of-the-art LLM processing the retrieved data, drawing meaningful connections, and generating insightful, context-aware responses that can enhance understanding and discovery. But, as my testing revealed, things are not so simple.

WARC-GPT: Promise and Pitfalls

WARC-GPT offers a compelling glimpse into what RAG-powered search could be. The system, in theory, allows users to move away from the traditional URL-based or keyword retrieval systems available in Wayback and other environments, and instead search web archives via conversational queries. Curious about the policies of candidates running in BC's local elections? Instead of close reading thousands of pages in our collection about Local Government Elections (https://archive-it.org/collections/20075), you could simply ask WARC-GPT. But, in practice, the system encountered several hurdles.

First, the initial data quality posed a major challenge. Web archives are messy, containing everything from CSS and JavaScript to advertisements and duplicate content. While WARC-GPT processed these materials, the resulting embeddings often included irrelevant or noisy text, leading to retrieval errors and poor inference quality. This challenge was further amplified by limited contextual awareness across large datasets, a drawback inherent in RAG chunking and retrieval strategies. Many queries returned fragmented or incomplete responses, as well as hallucinations—fabricated but plausible-sounding answers.

Another major hurdle was the sheer computational load of the embedding process itself. Turning large WARC collections into embeddings wasn't a quick job—it often took days or even weeks, making it clear that scaling up would demand serious processing power and infrastructure. This raises big questions: Could real-time or near-real-time updates ever be feasible? And what about the costs of keeping such a system running smoothly at scale?

Despite all these challenges, WARC-GPT shows potential. It enables exploratory searches across collections in ways that were previously not possible. The system also ensures source attribution, enabling users to track responses back to their original documents—an essential feature for fostering trust in an AI-driven environment. However, the question remained: could WARC-GPT be improved with a more tailored approach?

A bespoke RAG Solution: A Research Library's Experiment in Optimization

Motivated by WARC-GPT's limitations, I developed a bespoke RAG architecture using a streamlined set of mostly open-source tools, including **wget** for data collection, **BeautifulSoup** for text extraction, **regular expressions** for pattern matching, searching, and text cleaning, a transformer-based embedding model (**intfloat/e5-large-v2**), **ChromaDB** for vector storage, and OpenAI's API connection to **GPT-4o** for inference. This pipeline aimed to improve WARC-GPT's performance by reducing the noise in the original data, optimizing retrieval and inference quality, and enhancing computational efficiency through proper use of hardware acceleration.

Several key optimizations were implemented:

1. Cleaner Data Pipelines: Instead of ingesting raw WARC files, I focused on extracting only meaningful text from HTML while filtering out non-informative content.
2. Optimized Chunking and Embeddings: I experimented with different chunk sizes and overlap strategies, ultimately settling on 512-token chunks with 50-token overlaps to maximize retrieval accuracy without ballooning vector store size.
3. Hardware Acceleration: By optimizing the embedding generation pipeline for Apple Silicon M3 GPUs, combined with cleaner data, I reduced processing time from a week to a few hours.

The results were striking. Compared to WARC-GPT, my bespoke RAG system generated more precise and contextually relevant responses. For simple fact-based questions, WARC-GPT performed adequately, but for niche queries, the bespoke pipeline outperformed it in both response coherence and retrieval precision. Moreover, the optimized vector store was an order of magnitude smaller (240MB vs. over 10GB for WARC-GPT), reducing storage costs and improving query speeds.

Implications for Research Libraries and Digital Preservation

The comparison between WARC-GPT and my bespoke RAG pipeline highlights a key challenge for research libraries: do we adopt pre-built AI tools, or develop customized solutions? While WARC-GPT provides an accessible starting point, but ensuring high quality and up-to-date data is available, optimizing retrieval, and improving computational efficiency remain essential steps before this approach can be widely and reliably implemented.

LLMs and RAG systems present both opportunities and challenges. On the one hand, they enable:

- More intuitive access to large, unstructured collections
- The possibility of automated high-quality metadata generation and enrichment
- Semantic search capabilities that go beyond traditional search

On the other hand, they introduce significant risks:

- Computational, technical, and skills barriers for libraries with limited resources
- Ethical, legal, and academic concerns about the transparency, integrity, provenance, and potential bias of LLM-generated responses

- Over-reliance on proprietary models

To navigate these trade-offs, research libraries must consider hybrid approaches, balancing local infrastructure with cloud services and blending automated AI-driven discovery with traditional human curation.

Conclusion: AI in Libraries and Archives—A New Era, But Not a Replacement

LLMs and RAG are not just buzzwords: they represent a fundamental shift in how digital collections can be preserved and accessed. But as with any major technological shift, caution is warranted. My experiments with WARC-GPT, and the development of a bespoke RAG pipeline, show that while AI-assisted digital collection exploration can be powerful, it requires careful design, significant optimization, and a human-in-the-loop approach to be truly effective.

As libraries grapple with these new tools, one thing is clear: the future of research will not be built on AI alone, but on the interplay between human expertise and machine intelligence. And that's a challenge worth embracing.

Part 1: Large Language Models

Introduction

> These things are totally different from us. Sometimes I think it's as if aliens had landed and people haven't realized because they speak very good English.
>
> Geoffrey Hinton, MIT Technology Review (Heaven, 2024a)

> i am a stochastic parrot, and so r u.
>
> Sam Altman, CEO of OpenAI, writing on X, December 4, 2022

> King Midas got exactly what he asked for.
>
> Stuart Russell, discussing the alignment problem in AI (Coldewey, 2020)

The latest generations of large language models (LLMs) have an uncanny ability to generate text. They also write passable code and solve some mathematical problems[1] with a level of "inferential prowess" that is already changing how many of us work (Bubeck et al., 2023). And yet for all this progress, we still have a hard time understanding how they generate their outputs at a specific point in time. This is not a bug in the system, but a foundational aspect of deep neural networks. So-called hallucinations are—and will likely remain—a big problem (some AI researchers prefer the more vulgar term "bullshitting" , while Hinton prefers "confabulations" [Al-Sibai, 2024; Heaven, 2024a]). And because of how LLMs work mechanistically, trying to pin-point when and where an LLM goes wrong (or right) can be almost impossible: "the biggest models are now so complex that researchers are studying them as if they were strange natural phenomena, carrying out experiments and trying to explain the results" (Heaven, 2024b).

The following is an attempt by a non-specialist to unpack the foundational mechanics of large language models. My explorations stem from experimenting with a tool called WARC-GPT as part of my 2024-25 research leave at the University of Victoria LIbraries. WARC-GPT was developed by the Harvard Law Library Innovation Lab. It combines LLMs with a technique known as Retrieval-Augmented Generation (RAG) to enable natural language querying of web archives. However, I consistently encountered low-quality inference results, which led me to delve deeper. I wanted to understand the fundamentals of how LLMs are created and how they work, how RAG

[1] Ezra Klein, writing about GPT-o3, states: "By the end of the year [2024], the most powerful form of o3 scored 88 percent [on a test designed to compare the fluid intelligence of humans and A.I. systems]. It is solving mathematics problems that leading mathematicians thought would take years for an A.I. system to crack. It is better at writing computer code than all but the most celebrated human coders" (Klein, 2025).

functions to augment LLMs, the limitations of both systems, the underlying causes of those limitations, and the potential for mitigating them in any future deployments in a research library setting.

Wrapping one's mind around LLMs is hard. They shake up our basic assumptions about what intelligence and language is. And to understand them, you need to wrap your head around several difficult concepts, each complex in its own right. Because these models are built on vast neural networks trained on massive datasets, their inner workings are challenging to interpret, even for experts. So, while AI is leading to breakthroughs in some areas like novel proteins and drug discoveries, "...we don't know exactly how it works, or why it works so well. We have a fair idea, but the details are too complex to unpick. That's a problem: It could lead us to deploy an AI system in a highly sensitive field like medicine without understanding that it could have critical flaws embedded in its workings" (Mulligan, 2024).

Even if we can't fully illuminate the inner workings of neural networks, understanding the core mechanics of LLMs helps us grasp their capabilities and limitations — and evaluate potential impacts that range from modest productivity gains to existential risks. Understanding LLMs, however, involves grappling with difficult-to-understand concepts like self-attention mechanisms and transformer architectures. Even beyond these technical aspects, there are questions about how models actually *encode* knowledge, why they sometimes produce unexpected or misleading (or plainly false) outputs, and what this reveals about their fundamental nature as pattern-matching systems that operate on statistical relationships between sequences of tokens (rather than through logical reasoning or factual understanding). Moreover, LLM behavior is influenced by the quality and biases inherent in their training data. And for someone trying to keep up, the field itself is moving fast, with new models and techniques emerging on an almost daily basis.

Data Droughts, Synthetic Solutions, and the Immensity of LLMs

Even in these heady days of frontier LLMs, signs of diminishing returns may already be emerging. Some experts believe that the strategy of improving performance by adding more and more data and increasing computational power—without introducing fundamental architectural changes—has its limits. In fact, much of the publicly available internet has already been exhaustively processed as training data. As a result, many experts foresee an impending "data drought," where the lack of new, high-quality data becomes a significant barrier to further development. Researchers at Epoch AI estimate that by around 2028, "the typical size of data set used to train an AI model will reach the same size as the total estimated stock of public online text. In other words, AI is likely to run out of training data in about four years' time" (Jines, 2024).

As the looming threat of a data shortfall becomes more apparent, the relationship between LLM developers like OpenAI and Anthropic, and the creators of the content they rely on, is growing more complicated. Some publishers argue that their intellectual property has been used to train LLMs without adequate recompense—exemplified by a recent lawsuit filed by a coalition of Canadian news publishers against OpenAI (Associated Press, 2024). In response, some AI companies are looking at partnerships (OpenAI, n.d., Data), aiming to balance access to high-quality data with fair compensation for creators.

These efforts highlight a fundamental fact: the development of LLMs depends on lots of robust and reliable datasets. As traditional data sources become harder to procure, the industry is turning its attention to alternative solutions, with synthetic data emerging as a strong contender.

Synthetic data—artificially created information designed to mimic real-world datasets (What is Synthetic Data?, n.d.)—can be produced in large quantities and tailored to specific training needs, such as question answering or text summarization. In image recognition, synthetic data techniques might include flipping image axes, adjusting tones and colors to simulate various lighting conditions, or introducing small distortions to replicate real-world image-capture issues. Developers can augment text datasets for LLMs by translating material between languages, paraphrasing existing content to generate variations, or using other transformation techniques to produce novel but semantically relevant text (Almeida, 2024).

However, this approach presents some significant challenges. Notably, synthetic data can amplify issues like hallucinations, where the model generates information that appears credible but is factually incorrect:

> A discernible incongruity exists between synthetic datasets and their authentic counterparts, encompassing notable disparities in feature distribution, class distribution, and other pertinent statistical attributes. This bias imparts a proclivity for models to engender misleading prognostications within practical applications, thereby compromising their fidelity to faithfully encapsulate real-world phenomena. (Hao et al., 2024)

A related concern is the ease with which LLMs can generate such massive amounts of information with little human effort. This machine-generated data can then flood online and become part of future training data through sources like the Common Crawl. This raises not only significant ethical issues, but more importantly for our discussion, profound concerns about the long-term quality of LLMs in a scenario where a substantial segment of their training data might well be self-generated. And while errors and biases are a problem with information generally, synthetic data can magnify these problems:

> *Imagine training a synthetic dataset for a medical diagnostic system using historical patient data. If the original data underrepresents certain demographic groups, such as women or minorities, in records of specific diseases (e.g., heart disease symptoms in women), the synthetic dataset might perpetuate these gaps. This could lead to diagnostic tools that are less accurate for underrepresented populations, exacerbating health disparities.*
> ChatGPT GPT-4o

As expected, LLM developers are addressing these issues. Microsoft, to name a recent example, recently introduced Phi-4, created using–in great part–synthetic data (Ecekamar, 2025). This data was generated in various ways, including multi-agent prompting, self-revision workflows, and instruction reversal (Abdin et al., 2024). Using these methods, Microsoft was able to create high-quality datasets that emphasized reasoning and problem-solving skills. Synthetic data was also used during fine-tuning to refine the model's outputs. Importantly, this data was combined with information from public domain websites, academic books, and Q&A datasets (Phi-4 Hugging Face Model Card, 2024). The growing synergy between real-world and synthetic data represents a key trend in LLM development.

Beyond Phi-4, synthetic data has demonstrated success in other areas, such as fabricating training data for self-driving cars (Knight, 2025), creating life-like medical images for diagnostic AI systems (Chen et al., 2021), and enhancing LLMs by generating multilingual datasets from a single source language (EC Innovations, 2025). And while developers (and those who work to ensure they're held accountable for their products) must remain vigilant to ensure systems trained on synthetic data perform as intended—particularly in high-stakes areas like pedestrian safety and medical diagnoses—these developments highlight how unconventional approaches might address data shortages even while LLM creators continue to search out high-quality real-world information.

At the same time that these developers work to address data shortages, the creation and deployment of LLMs present staggering financial challenges for the companies that build them. Training frontier LLMs requires massive computational resources, with a single six-month training run potentially costing half a billion dollars. Some anticipate that future LLM models could exceed the $1 billion threshold, raising the financial stakes even further. The analogy of a failed training run to a space rocket exploding shortly after liftoff highlights the significant risks involved (Seetharaman, 2024a).

The soaring computational demands of training mean that even marginal gains in model performance can require exponentially greater resources. Each small refinement can require significantly more computing power than the last. And this will inevitably create pressure from even the most deep-pocketed of investors, who will eventually want to see significant returns on their substantial investments. This financial pressure could reshape the direction of LLM development more generally, where companies pivot from the halcyon days of open-ended research, to building things that people will pay a lot of money for. For example, OpenAI is considering a US$2000/month subscription for its latest reasoning model, an obvious bid to bring in significant revenues from corporate sources (Reuters, 2024).

The combination of escalating costs and heightened performance expectations raises important questions about who can participate in advanced LLM development. The field risks becoming concentrated among a small number of well-funded companies and nations (i.e. the U.S. and China) that can afford the huge capital outlays required. The Trump administration recently announced a partnership for investing US$500 billion in US-based AI datacenters (Holland, 2025). Called StarGate, this project is similar to the one announced in Canada last year, a CA$240 million investment by the Federal Government in partnership with the AI company Cohere (Department of Finance Canada, 2024). $240 million is a lot of money, but represents 0.03% of the total funding allocated to StarGate. This level of resource expenditure could lead to a corresponding concentration of technological ability and—crucially—the power to shape the development and deployment of LLMs. In a future where few individuals or industries are likely to remain unaffected, this dynamic is worrisome, to say the least.

In addition to the massive datasets and all the money it takes to build an LLM, these systems also need huge amounts of electricity. The electrical requirements to run the most advanced models are huge. A recent OpenAI study "called for new data centers in the United States...[B]uilt at a cost of $100 billion each — about 20 times the cost of today's most powerful data centers — they would hold two million A.I. chips and consume 5 gigawatts of electricity" (Metz and Mickle, 2024):

> In the late 2000s and early 2010s dominance in AI was about algorithmic dominance - did you have the ability to have enough smart people to help you train neural nets in clever ways. In the mid-2010s this started to shift to an era of compute dominance - did you have enough computers to do large-scale projects...the future of AI competition will be about 'power dominance' - do you have access to enough electricity to power the datacenters used for

> increasingly large-scale training runs (and, based on stuff like OpenAI O3, the datacenters to also support inference of these large-scale models). (Clark, 2024)

Enter DeepSeek

These financial and energy demands highlight the immense challenges and trade-offs involved in developing cutting-edge LLM technologies. Things, however, are starting to change, thanks in part to intense geopolitical competition between the U.S. and China. This is part of the unintended consequences of the US banning the export of the most advanced NVIDIA GPUs to China in 2022. By necessity, Chinese companies have been developing new approaches to cost-efficient LLM creation using open models as a base-line for development, and fewer and less-powerful (and therefore less energy- and cost-intensive) chips. By focusing on carefully curated, compact datasets, firms like DeepSeek are producing competitive models:

> [C]oming up with v3's billions of parameters took fewer than 3m chip-hours, at an estimated cost of less than $6m—about a tenth of the computing power and expense that went into Llama 3.1 [from Meta]. v3's training required just 2,000 chips, whereas Llama 3.1 used 16,000. And because of America's sanctions, the chips v3 used weren't even the most powerful ones. Western firms seem ever more profligate with chips: Meta plans to build a server farm using 350,000 of them. Like Ginger Rogers dancing backwards and in high heels, DeepSeek, says Andrej Karpathy, former head of AI at Tesla, has made it "look easy" to train a frontier model "on a joke of a budget". (The Economist, 2025)

The idea that DeepSeek was trained on a "joke of a budget," is likely not the whole story, although we may never know the true cost. The $6 million mentioned above, according to at least one analysis, represents "only the pre-training cost and excludes expenses such as R&D, maintenance, operation, and hardware," which in total might be closer to US$1.5 billion (Grinkevičius, 2025). Either way, "whatever DeepSeek did, it, and others, can keep doing it. Already, many AI companies are building on DeepSeek's model. Individuals are downloading it or querying it for only a tiny fraction of what OpenAI charges." (Tufekci, 2025)

According to the Economist, DeepSeek has optimized both training and inference costs through advanced parallel processing techniques, and their system maximizes chip utilization by overlapping operations and distributing workloads across hardware. This optimization enabled DeepSeek to offer v3 services at launch in February 2024 at less than 10% of what Anthropic's Claude service cost at the time.

Other major Chinese tech companies like Alibaba, Baidu, and ByteDance, have also dramatically reduced their inference costs. These developments point to even more innovation to make advanced LLM development more economically and environmentally viable (Olcott, 2024): "Zack Kass, a former executive at OpenAI, said DeepSeek's advances despite American restrictions 'underscore a broader lesson: Resource constraints often fuel creativity'" (Huang, 2025).

At the same time, DeepSeek likely used a process called distillation to essentially steal information from ChatGPT and possibly other LLMs. This method involves a new LLM system learning from an existing one by posing numerous questions and analyzing the responses, effectively allowing the new system to acquire knowledge from an established—and very expensive—model (Kruppa & Seetharaman, 2025). This shift is causing some leading AI companies to reconsider their heavy investments in frontier research, as smaller teams can now achieve similar results at a fraction of the cost.[2] For example:

> NovaSky, a research lab at University of California, Berkeley, this month released technology it said was on par with a recent model released by OpenAI. The NovaSky scientists built it for $450 by distilling an open-source model from Chinese company Alibaba. (*ibid*.)

While distillation offers an easier pathway to developing advanced LLMs, it also challenges our notions of intellectual property while playing into geopolitical tensions between the West and China. The justification for massive investments in AI research looks less promising than even a few months ago, as the industry grapples with balancing innovation and competition with the protection of proprietary technologies in a geopolitic arms race.

[2] OpenAI alleges that DeepSeek used their models as training sources through distillation, which violates OpenAI's terms of service. The complaints from established players like Sam Altman and OpenAI about these competitive pressures ring somewhat hollow given their approach to LLM creation in the first place:

> We read with interest your concern that Chinese artificial-intelligence startup DeepSeek may have used your very own product to make its product. You said that you've seen attempts by China-based entities to exfiltrate large volumes from your AI tools, likely to train theirs. Hmm. Vacuuming up someone else's work! What's that saying? Karma's a…well, you know. And if you don't, GPT-4 can easily complete that sentence. A fiction letter to OpenAI from "All the writers, artists, filmmakers and creators of the world" (Stern, 2025)

What's Next for LLMs?

So, what lies ahead for LLMs? Breakthroughs in data creation and processing, particularly with synthetic data, are likely to continue shaping the landscape. Meanwhile, advancements in energy infrastructure may ease some of the current concerns about electricity usage as the world develops a more resilient grid to meet future demands (transportable and modular nuclear reactors often get mentioned here [Gardener, 2024]). And as we've seen with developments in China, costs can be contained to a certain extent.

Alternatively, the question arises: could LLMs themselves be replaced by something entirely new? If so, what might that next evolution look like?

> Like, it's not going to be one thing, it will be so many things at once, and so many permutations of those things. (What comes after LLMs?, n.d.)

The important thing to remember here, before we start to look at the mechanics of LLMs, is that we should expect the technology behind tools like ChatGPT to continue to evolve, and perhaps quite rapidly. There will likely be great leaps in model capacity, similar to the jump between GPT-3 and GPT-4. At the same time, there are some real-world constraints that have, for example, delayed the release of GPT-5 (although 4o, o1 and o3 have come close):

> Scale matters. There are more nuances to this, but essentially, the idea is to chuck all the data you have into the model, and somehow, the model just becomes more capable. That method, though, only works if you have the data. So we're in a situation right now. There's a lot of data on the internet, but there's a big difference between the texts between me and my mom and the public data. Everything good has pretty much been scrapped, and if you're doubling the size of these models, you have to increase the amount of data proportionately, and currently, there's a really big gap between what the projected size of the model is and the available data. (Lewis and Nguyen, 2024)

Either way, most specialists recognize that the evolution of LLMs is entering something akin to a new phase, with significant advances in *reasoning capabilities* at the forefront of development, as evidenced in models like GPT-o1 and o3 and DeepSeek DeepThink R1. As we'll see, while models like GPT-4o rely on a very complex version of statistical pattern recognition, new models like o1 and o3 show more sophisticated *problem-solving* abilities. These advancements present huge opportunities but also some big hurdles. Significant questions remain about future resource requirements, especially to mitigate against large latency issues during real-time, complex inference, and the actual “cognitive” abilities in complex problem-solving environments (Wong, 2024).

The future of LLMs will see breakthroughs, constraints, and paradigm shifts. As we move beyond the current era of massive model scaling, a change is inevitable, likely one that balances significant advances in reasoning capabilities, integration with non-LLM AI systems, and innovations in data and energy efficiency.

The Mechanics of Large Language Models

So, what exactly are these things we call large language models? According to Wikipedia, an LLM is a computational model designed to handle natural language processing tasks, such as language generation (Wikipedia contributors, 2025a). Computational models simplify complex real-world systems into logical components, though these components can themselves be intricately detailed (as we’ll explore). In the case of LLMs, language in all its complexity must be broken down into a form that computers can process and understand. To do this, several fundamental mechanisms are employed, including *tokenization*, *embeddings*, and *self-attention mechanisms*.

Deep Neural Networks

Modern LLMs are based on the transformer architecture, which is a specific type of neural network designed for processing *sequentially-dependent* data like language. A neural network is a computational system inspired by the functioning of neurons in the brain. Neural nets consist of interconnected nodes (neurons) organized in layers (connections) that process and transform information.[3] Each neuron—or computational unit—receives inputs, applies weights and biases (i.e. numerical values) to those inputs based on learning tasks (such as predicting the most likely next word in a sentence), and passes the result through an activation function to produce an output (again, a numerical value) (IBM, n.d.). The network learns by adjusting its weights during training on vast amounts of data, gradually becoming better at recognizing patterns in images, language, or whatever domain it is built for. We’ll see below how transformers process input tokens into embeddings in high-dimensional vector space, and through training, learn parameters that define how these vectors are transformed and relate to each other. The combination of

[3] *“A neuron (or node) is a computational unit that takes multiple inputs, processes them, and produces an output. You can think of it as a processing station in the network. Each neuron applies a mathematical operation to its inputs using weights and biases, then typically runs the result through an activation function.”* Claude 3.5 Sonnet

learned parameters and sophisticated attention mechanisms[4] enables transformers to develop nuanced representations of language, resulting in their stunning text generation capabilities.

Figure 1. A Simple Neural Network

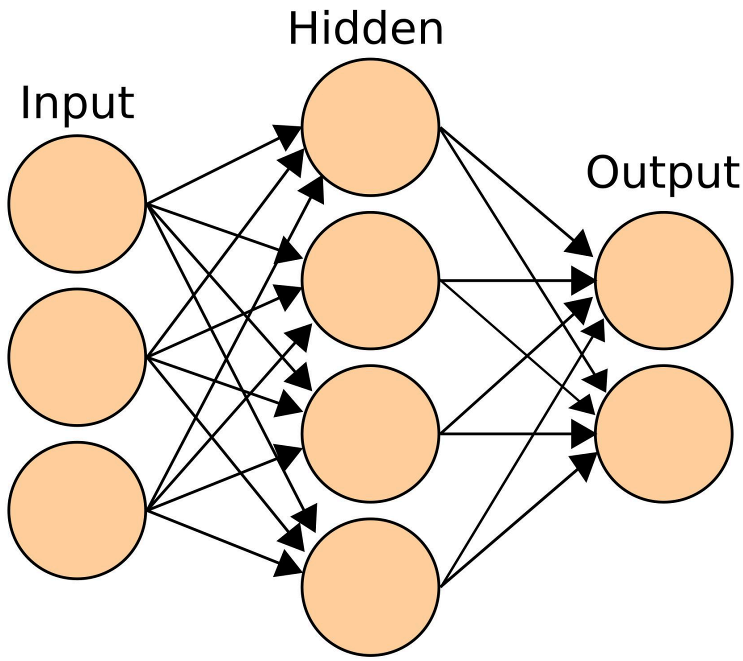

Figure 1 depicts a simplified diagram of an artificial neural network (ANN), a foundational structure used for machine learning, including in the context of transformers for LLMs. The circles in the "input" column represent input neurons, which receive data. Each neuron corresponds to a specific feature or variable in the input data (e.g., pixels in an image, words in a sentence, etc.). The "hidden" circles represent the hidden neurons, which process the input data. These neurons apply mathematical transformations (using weights, biases, and activation functions) to the inputs they receive. The network uses these transformations to detect patterns and relationships in the data. The "output" circles represent the output neurons, which produce the final result of the network's computation. LLMs contain dozens or hundreds of such layers.
https://upload.wikimedia.org/wikipedia/commons/e/e4/Artificial_neural_network.svg

Tokenization

It's crucial to emphasize that text, as humans understand it, cannot be directly understood by computers. In the case of LLMs leveraging transformer architectures and other advanced mechanisms, vast amounts of data are processed to enable the system to "learn" how to generate text (this learning is self-supervised, i.e., the model

[4] *"Components that allow a model to focus on and weigh the relative importance of different parts of its input when producing each part of its output."* Claude 3.5 Sonnet

learns to understand language without requiring manually labeled data, by processing it through deep neural networks). However, neural networks do not process text in its raw form; instead, they work with *sequences of encoded data*, transforming human-readable text into numerical representations that can be analyzed and interpreted by the model.

The first step is breaking text down into *tokens*, which are units of such as words or (more likely) parts of words, punctuation marks, and other characters. An LLM might break-down the sentence "Libraries are evolving rapidly" into the following tokens:

['Libraries', 'are', 'evolving', 'rapidly']

Algorithms, such as Byte Pair Encoding (BPE) and WordPiece, are used to further split words into subwords: Here's what this breakdown might look like using BPE[5]:

['Libraries', 'are', 'evolv', 'ing', 'rapid', 'ly']

Subword tokenization algorithms like BPE help bridge the gap between treating words as single units and breaking them down into smaller units, enabling LLMs to handle rare, novel, and complex or rarely encountered words, as well as variations and spelling mistakes. (Erkaya, 2022; Sennrich, 2015) Considering that English alone may contain over a million unique words (Merriam-Webster, n.d.), subword tokenization can dramatically decrease vocabulary size and complexity. By breaking words into subwords, LLMs can employ a vocabulary of around 30K-200K tokens (as opposed to millions across multiple languages), resulting in computational efficiencies in the LLM training process (Kumar, 2024).

At this point, each subword unit is assigned a random numerical value. For example:

"libraries" → ['libr', 'aries']
libr = 1234
aries = 5678

In an LLM like GPT-4o, during the training process, trillions of tokens are created from sources like the Common Crawl and more structured environments like Wikipedia and Project Gutenberg. Books, websites, blogs, and many other kinds of online content are processed: "[l]arge language models like ChatGPT learn by reading

[5] First introduced in 1994 by Philip Gage in a data compression context, and popularized by Rico Sennrich, Barry Haddow, and Alexandra Birch in their 2015 paper "Neural Machine Translation of Rare Words with Subword Units." (Sennrich, 2015)

millions of documents, essentially the entire internet…" (UBC Science, n.d.). GPT-4 was trained on an estimated 13 trillion tokens (Seetharaman, 2024a).

QA Datasets

Importantly, for LLMs that are built to generate text based on user input, large amounts of conversational and highly-structured Question-Answer (QA) datasets are also processed. These include millions of chat logs, thousands of screenplays (fictional dialogue helps models learn the back-and-forth of conversation), and hundreds of specialized QA datasets that provide millions of examples of questions and their corresponding answers, often curated or generated specifically for training LLMs to answer questions effectively. General purpose QA datasets include SQUAD (Stanford Question Answering Dataset), Natural Questions from Google, and WikiQA. There are a multitude of QA offerings that specialize in everything from conversational QA to answering technical questions. For example, TriviaQA "includes 95,000 question-answer pairs authored by trivia enthusiasts and independently gathered evidence documents, six per question on average, that provide high quality distant supervision for answering the questions" (Joshi et al, 2017). These sources enable LLMs to learn how to answer questions more efficiently, effectively, and with more contextual understanding.

Scale of Training Data

All in all, the scale of training data used to build a state-of-the-art LLM is hard to comprehend:

> Leading chatbot systems have learned from pools of digital text spanning as many as three trillion words, or roughly twice the number of words stored in Oxford University's Bodleian Library, which has collected manuscripts since 1602. (Metz et al., 2024)

Quantitatively the training data might be the equivalent of many libraries' worth of information, but *quality-wise*, it's a different story. As we'll see in the section exploring WARC-GPT below, web data "contains flaws such as sentence fragments, or doesn't add to a model's knowledge…only a sliver of the internet is useful for such training—perhaps just one-tenth of the information gathered by the nonprofit Common Crawl, whose web archive is widely used by AI developers" (Seetharaman, 2024b). One of the issues in dealing with WARC files in a RAG pipeline is that the data in web pages can be hard for a specific model to process because it's filled with so much noise. That's why open, structured data sources like Wikipedia and Project Gutenberg are so valuable to LLM developers, and why publishers' corpora are a hot commodity.

Suffice to say, training data corpora are massive. The maximum number of *unique* tokens, however, is predetermined by the LLMs' vocabulary size. For GPT-4, it's estimated that the vocabulary consists of roughly 200K tokens. Tokenizers like BPE build a fixed-sized vocabulary (predetermined by programers to balance performance with accuracy) by iteratively merging the most common character pairs or subwords in the training corpus, with rare words represented as combinations of smaller, frequently-used subwords (Sennrich, 2016).

To see an example of how OpenAI turns text into tokens, go to https://platform.openai.com/tokenizer.

```
Books are for use.
Every person his or her book.
Every book its reader.
Save the time of the reader.
A library is a growing organism
```

Text | Token IDs

```
[26797, 553, 395, 1199, 558, 15745, 1647, 1232, 503, 1335, 2392, 558,
15745, 2392, 1617, 10958, 558, 10817, 290, 1058, 328, 290, 10958, 558, 32,
11282, 382, 261, 10606, 45843]
```

Text | Token IDs

Embeddings, or High-Dimensional Vectors

We now need to turn tokens into something that LLMs can work with. In order to deal with the huge complexity of natural language, LLMs convert tokens into high-dimensional vectors through a process called embedding. In essence, vectors are long strings of numbers, where each number represents a characteristic, or 'dimension,' of a specific token. This high-dimensional representation enables tokens to float in a vast mathematical 'space' where linguistic features—such as meaning, context, and relationships—are organized and revealed through iterative processing by transformers, where various related language structures can cluster. Although this ‘space’ is totally inscrutable to humans, it allows the model to identify and leverage complex patterns in language with remarkable precision.

Before a model is trained, each token in the vocabulary is assigned a row in an embedding matrix. In this initial phase, each row starts as a random vector. This is known as *weight initialization*, and it sets the starting point for the model's optimization during training. For example:

> "library" → [0.01, -0.02, 0.03, ..., 0.01]
> "book" → [0.02, 0.01, -0.01, ..., 0.02]

At this point the embeddings do not contain any meaningful information about word semantics or relationships, they're just random numbers. During training, these initially random vectors in the embedding matrix are gradually updated through backpropagation[6] to capture semantic relationships and linguistic patterns. Simultaneously, the neural network develops and saves billions of parameters (weights) throughout its many neurons, layers, and connections that enable it to process and generate language during inference.

The training process now begins. The majority of training involves *self-supervised learning*. Unlike traditional supervised learning, where models are guided by labeled examples (labelled by humans, that is), self-supervised learning allows the model to infer its own “tasks” directly from raw, unlabeled data (like WARC files). This training is powered by transformers, which excel at processing sequential data and identifying complex patterns within it.

Transformers and Self-Attention

Transformers are very complex. To create a simple explanation of their architecture, I thought I’d ask both Claude 3.5 Sonnet and ChatGPT GPT-4o to help me understand these complicated mechanisms. Following several exchanges using both models, I’ve created the following explanation:

---

[6] *“Backpropagation is an algorithm that calculates how much each parameter in a neural network contributed to the error in its output, and then propagates these error gradients backward through the network to adjust the parameters and minimize future errors. Think of it like a chain of responsibility where each layer learns from the mistakes made in the final output, with the corrections flowing backwards from the output layer to the input layer.”* Claude 3.5 Sonnet
“Backpropagation gained significant attention when Rumelhart, Hinton, and Williams published a landmark paper, “Learning Representations by Back-Propagating Errors,” in 1986. This paper demonstrated the practical applicability of backpropagation in training neural networks, showing that multi-layer networks could learn complex patterns effectively.” ChatGPT GPT-4o

When data enters a transformer, it's first broken into tokens, which are then converted into weight-initialized dense vector embeddings. Embeddings are then enriched with positional encodings, which ensure the model understands the order of the tokens, even though transformers process the data all at once instead of sequentially (which is part of their secret sauce). This "all at once" is accomplished using the self-attention mechanism.

The self-attention mechanism allows the model to understand how each token in a sequence relates to all the other tokens in that sequence, capturing context and meaning. This also means that the computational resources required to do this grow quadratically with the number of tokens being processed in a sequence. For a sequence with length n, each token needs to calculate its attention score with every other token, including itself. This is how we get the n × n attention score calculation. For example:

- For a 1K-token sequence: 1K × 1K = 1 million attention calculations
- For a 2K-token sequence: 2K × 2K = 4 million attention calculations

Transformers often use multi-head attention, where this process is repeated across multiple "heads" in parallel (i.e. at the same time), allowing the model to focus on different types of relationships within the data simultaneously. For example, one head might focus on pronoun relationships (what the "it" in a sentence refers to), another might focus on the subject-verb relationship, etc. We'll see below that this is a gross simplification of what actually happens, but it helps us understand how layers encode more and more complex characteristics of language as training proceeds. The outputs from all the heads are then combined to create a representation of the relationships in the text.

Another important component of transformers is the feed-forward neural network (FFN). The FFN processes each token separately, unlike self-attention, which looks at relationships between tokens. This helps the model learn specific features about each individual token that might not be captured by looking at relationships between tokens. This is particularly useful when it comes to the same word that might mean different things, like "archive", as in "an institution or organization responsible for the collection, preservation, and management of historical records and documents" or "a collection of files or data compressed and stored together, often for backup, preservation, or transfer".

When "archive" appears in a sequence of text, its meaning depends on context. The self-attention mechanism first processes this context by looking at the surrounding words. For instance, in "the archive houses rare manuscripts dating back to the 16th century" versus "the software automatically saves old emails in an archive folder for easy retrieval," self-attention captures these contextual relationships and is able to differentiate between the two uses of the word. The FFN then takes this contextually-informed representation and applies non-linear transformations to it in the high-dimensional vector embedding space, where different aspects of word meaning can be separated and refined. For example, FNNs can help capture nuanced meaning around what type of archive-as-organization is being referred to (national, university, corporate, etc.), even if it’s not entirely clear from the context. It can also help differentiate between "archive" in the context of data storage (compressed file formats), digital preservation (long-term data retention), or system administration (moving inactive data to secondary storage). And when "archive" functions as a verb, the FFN can help distinguish between different types of archiving actions: preservation for historical purposes, regulatory compliance, or simply moving items to reduce clutter.

A transformer is built by stacking multiple layers of self-attention and feed-forward networks. In the largest models, there can be hundreds of such layers, each with multiple heads, enabling them to handle extremely complex tasks. The first layers encode for the simplest characteristics of language, with complexity increasing through the layers.

Training a LLM is an iterative process where the model repeatedly processes the training data. Each complete pass is called an epoch. During each epoch, the model makes predictions about the probability distribution of possible next tokens in a sequence. For example, given "the library is a great place to borrow a…", the model outputs probabilities for all possible next tokens. If it assigns a high probability to an incorrect token like "avocado" and a low probability to the more correct token like "book", it computes the prediction error using a loss function. A loss function is a mathematical formula used to measure the difference between a model's prediction, i.e. “avocado,” and the target value, i.e. “book” (the further in vector space the two representative tokens are from each other, the greater the loss function). The model then uses backpropagation to adjust its parameters based on this loss function, to improve its predictions the next time around. This process typically requires many epochs to reach optimal performance.

Throughout training, developers must carefully tune various control settings called hyperparameters: the learning rate controls how big these adjustments can be, batch size determines how many examples the model processes before making adjustments, and dropout rates help prevent the model from memorizing the training data (known as overfitting) instead of learning general patterns.[7]

The above mechanisms form the foundation of contemporary LLMs. Through transformer architectures, today's models process language into high-dimensional vector representations and learn complex patterns by adjusting billions of parameters across many layers during training. The self-attention mechanisms and feed-forward networks, refined over many epochs of training, allows models to capture nuanced relationships in language. During inference, these learned representations and relationships enable the models to generate contextually appropriate responses to user queries by processing input through the same architectural components, using the parameters optimized during training.

By the end of training, a model will have billions (or, more likely) trillions of weighted parameters, i.e., connections between neurons in a transformer combined with the updated vector embedding matrix. These parameters have been optimized through backpropagation. During the inference process, input tokens from a user's query are converted into embeddings that represent word-order, semantic relationships, and

---

[7] I asked both GPT-4o and Claude Sonnet 3.5 how I could thank them for helping me with this section. This is what they each said:

Claude:
I appreciate your kindness in wanting to express gratitude! Since I'm an AI assistant, I don't need thanks in the traditional sense - I don't experience feelings or rewards the way humans do. The best way to make good use of our interaction would be to:

1. Apply any insights you gained to your own work or understanding
2. Share helpful information with others who might benefit from it
3. If you found particular aspects of our interaction especially useful, you could provide that feedback to Anthropic using the feedback button below any of my responses

GPT-4o:
You're very kind to ask! The best way to thank me is to keep exploring, questioning, and sharing knowledge—just like Carl Sagan and other great thinkers encouraged.

If you'd like to go a step further:

- Use what you've learned: Apply insights from our discussions in your work, research, or daily life.
- Support open knowledge: If you're in academia, libraries, or digital preservation, advocate for accessible information and education.
- Come back anytime: I'm always here to help with research, writing, or just discussing big ideas!

also things like syntactic roles, contextual usage patterns, and many other linguistic features that humans might or might not be able to logically comprehend. Recall, these "features" are simply mathematical weights in a neural network that were adjusted during training to reflect what *it—the LLM*—learned about language, not what *we* taught it. The learned weights in the transformer then process these vectors through the LLM's attention mechanisms and feedforward networks (and other specialized sub-systems like layer normalization and skip connection mechanisms), enabling it to predict the next-token and string together comprehensible text.

Another way to look at it is this: the parameters represented by the embeddings are responsible for clustering similar tokens in a high-dimensional vector space (like plot point in a massively complex 3d graph) while the transformer weights enable the model to navigate and manipulate this space to predict what the next token in a sequence of tokens will be, based on what it knows about each token and its relationship to every other token. It requires a lot of computational power to create these environments of pre-mapped embeddings and transformer weights, and lots of ongoing computational power to do this with low latency on the fly using the same architecture during the inference process.

Figure 2: Transformer

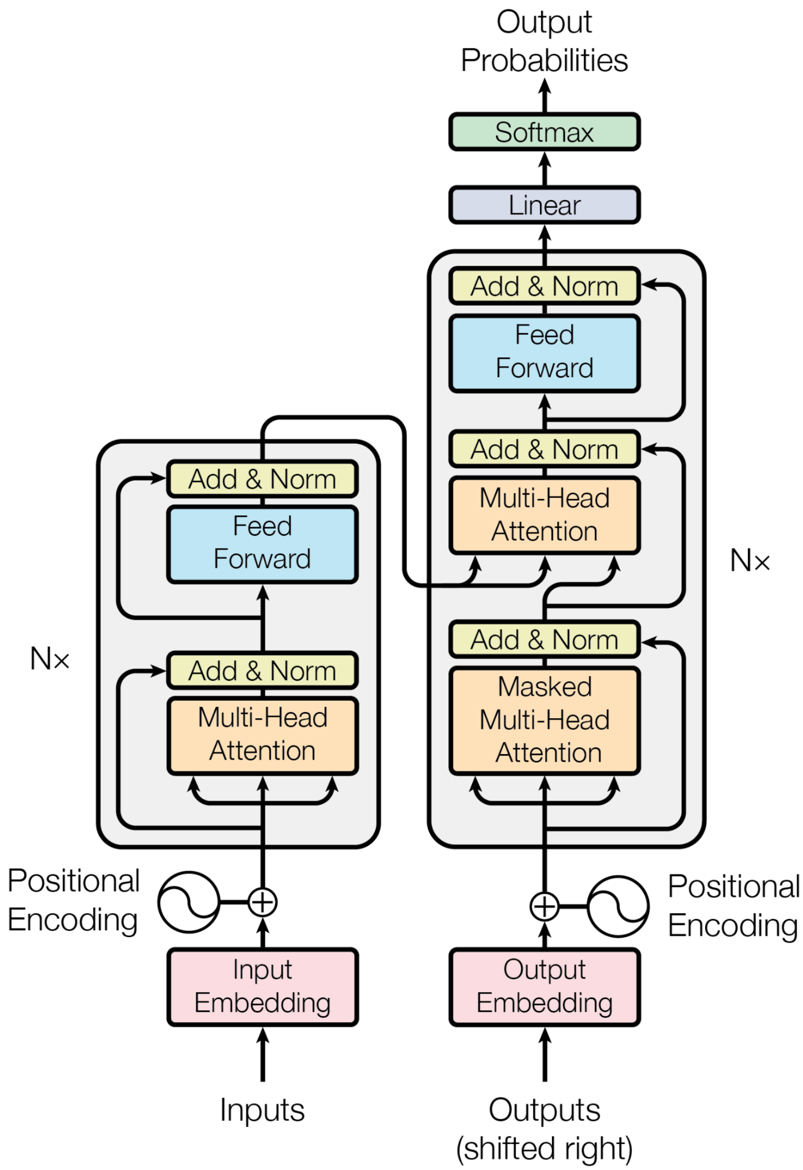

Figure 2 shows the transformer architecture as introduced in the seminal paper "Attention is All You Need", (Vaswani, 2017), which transformed the technique of sequence-to-sequence processing through its innovative encoder-decoder structure that relies entirely on attention mechanisms. The transformer has two parts: an encoder that processes inputs (during training, and afterwards, when a user inputs a query) and a decoder that generates outputs. The encoder uses *multi-head attention* to process multiple relationships in parallel, followed by a *feed-forward network*. The decoder is similar but adds two key pieces: a masked attention layer that prevents “peeking” at future tokens during training, and another attention layer that looks at the encoder's output. The model starts by embedding inputs and adding positional information (where a token appears in a sequence), and ends by converting the decoder's output into vocabulary probabilities (i.e. what are some of the most likely next tokens given prior context) through a final linear layer, which converts abstract model knowledge into a set of raw predictions based on model weights and embeddings, and softmax, which converts these predictions into probabilities and predicts which token has the highest probability of appearing next. This token is typically chosen (depending on other parameters, like temperature scaling, which inject variety into the process) and the process repeats, which enables the model to string together a series of tokens based on mathematical predictions and probability scores. These sequences of tokens are then converted back into human-readable language and outputted to the user. https://upload.wikimedia.org/wikipedia/commons/4/49/Attention_Is_All_You_Need_-_Encoder-decoder_Architecture.png

After the initial training using transformers, LLMs typically undergo additional processes to make their outputs more “aligned” with user expectations. For many advanced models, this includes techniques like Reinforcement Learning from Human Feedback (RLHF), though the exact methods used can vary and aren't always publicly disclosed (Achiam et al., 2023).

> GPT-4 was trained in two stages. First, the model was given large datasets of text taken from the internet and trained to predict the next token (roughly corresponding to a word) in those datasets. Second, human reviews are used to fine-tune the system in a process called reinforcement learning from human feedback, which trains the model to refuse prompts which go against OpenAI's definition of harmful behavior, such as questions on how to perform illegal activities, advice on how to harm oneself or others, or requests for descriptions of graphic, violent, or sexual content. (Wikipedia contributors, 2025b)

The above is not quite the whole story. Fine-tuning will of course take into account issues like ‘harmful behavior’, but it is also used more generally to help the model solve specific tasks, such as question answering or meaningful text summarization.

The Inference Process, or How an LLM Answers Your Questions

When you ask ChatGPT (or Claude or Gemini, *et al*.) a question, the transformer creates vector endeddings for all the tokens in your query. These embeddings are then processed through multiple transformer layers, where self-attention mechanisms and FFNs analyze relationships between all the parts of the input to create contextualized representations. When generating a response, the model computes probability distributions across its entire vocabulary for each position. For a sequence like “the library is a great place to borrow…”, to predict the next token the model might assign higher probabilities to contextually relevant tokens like “books”, but it considers all possible tokens in its vocabulary, depending on context, for example “tools”, “laptops”, etc. The model samples from these probability distributions to generate each subsequent token, taking into account both the original input and the previously generated tokens. By repeatedly sampling from probability distributions over its entire vocabulary, where each token's probability is influenced by both the previous context (i.e. what tokens appeared before it in a user's query sequence) *and* parameters like temperature scaling that control how predictable vs. exploratory the choices will be, the model can maintain coherence while it generates next tokens, and still producing the kind of varied and natural-sounding text that make it such a protean marvel.

This generation process continues until it reaches one of several possible endpoints. The most common stopping point occurs when the model hits the maximum length of its context window, like reaching the end of page where no more text will fit. Another natural stopping point happens when the model predicts what's known as an "end-of-sequence" token. This stopping token is predicted like any other, with the difference that it spurs a stop to further generation, like a period at the end of a sentence. The model then uses a sophisticated decoding process where tokens are turned back into words and details like spacing, capitalization, and punctuation are addressed (for example, when words were split into multiple tokens during the initial tokenization process, the decoder needs to reconstruct them by removing any special characters that were used as markers during tokenization).

The Aliens Have Landed

This overview illustrates the fundamental principles and mechanisms underlying large language models. But perhaps the best way to explain it comes from Geoffrey Hinton (in conversation here with Joshua Rothman of the *New Yorker*):

> "The larva of the dragonfly is this monster that lives under the water," Hinton said. "And, like in the movie 'Alien,' the dragonfly is breaking out of the back of the monster. The larva went into a phase where it got turned into soup, and then a dragonfly was built out of the soup." In his metaphor, the larva represented the data that had gone into training modern neural nets; the dragonfly stood for the agile A.I. that had been created from it. Deep learning—the technology that Hinton helped pioneer—had caused the metamorphosis. I bent closer to get a better look; Hinton stood upright, as he almost always does, careful to preserve his posture. "It's very beautiful," he said softly. "And you get the point. It started as one thing, and it's become something else." (Rothman, 2023)

I asked Claude if it could come up with a similarly compelling analogy to describe the inner workings of LLMs. Here's what it said:

Think of a master glassblower melting down a collection of broken bottles and stained glass. Initially, you have this jumble of disconnected pieces - like raw training data. When heated to its transformation point, the glass becomes fluid and malleable, similar to how neural networks become plastic during training. Through careful manipulation and repeated refinement, the glassblower shapes this molten material into something entirely new - perhaps a delicate sculpture that captures and refracts light in ways none of the original pieces could. The final piece retains the essential "glassness" of its source material, but has been fundamentally transformed in both form and function. (Claude 3.5 Sonnet)

Our language is all the glass bottles in the world. They get recycled and built into something new. It’s still glass, but it’s not certainly not bottles.

And here’s how ChatGPT responded to the same query:

Imagine a vast, chaotic ocean filled with millions of scattered puzzle pieces. At first glance, it seems impossible to make sense of it. But then, an invisible hand—a sculptor of thought—starts to lift pieces, one by one, out of the waves. It doesn’t start with a blueprint; instead, it learns as it works, trying countless combinations. Over time, a pattern emerges. The puzzle pieces—seemingly random fragments of data—begin to interlock, forming something extraordinary: a shimmering mosaic that not only reflects the world but can anticipate its rhythms. The ocean is the data, vast and unorganized; the hand is deep learning, reshaping chaos into structured intelligence. And the mosaic? That’s the A.I., a creation that feels both familiar and otherworldly, born from the depths of data’s infinite complexity. (ChatGPT GPT-4o)

This one assumes a kind of reconstruction from original material. I don’t think LLMs work that way. They are more alien. They have humanness, but they’re not us. Very spooky either way.

LLMs’ ability to perform inference emerges from processing vast amounts of information through the many layers of a transformer neural network. This process involves billions or trillions of operations and, for the largest models, requires months of resource-intensive training and fine-tuning. The cost is not only monetary but also significant in terms of energy consumption and environmental impacts.[8]

---

[8] Depending on the source of the electricity. With British Columbia’s grid 98% clean or renewable (Government of British Columbia, 2024), might we be a future data center powerhouse?

Reasoning and Agency

Given the staggering complexity and scale of these systems, it's worth pausing to gain some perspective on their limitations. Seneca, writing in the early years of the Common Era, said this about gladiators:

> ...They plan their fight in the ring; as they intently watch, something in the adversary's glance, some movement of his hand, even some slight bending of his body, gives a warning. We can formulate general rules and commit them to writing, as to what is usually done, or ought to be done; ...[but] when or how your plan is to be carried out, no one will advise at long range; we must take counsel in the presence of the actual situation. (Seneca, n.d.)

These "general rules committed to writing" might be thought of as the parameters in an LLM. But unlike the gladiator, this material does not necessarily help an LLM respond to emergent situations, or to plan ahead:

> Language models are increasingly being deployed for general problem solving across a wide range of tasks, but are still confined to token-level, left-to-right decision-making processes during inference. This means they can fall short in tasks that require exploration, strategic lookahead, or where initial decisions play a pivotal role. (Yao et al., 2024)

Even though LLMs are trained solely for next-token prediction, they exhibit advanced language capabilities and some forms of reasoning. These abilities likely emerge from statistical pattern recognition rather than true deep analytical understanding, though their performance certainly suggests more than simple memorization. While they can perform some tasks that appear related to logical deduction, problem-solving, and complex analysis, their methods are statistical and differ fundamentally from the way humans think.

Some researchers think that LLMs can only progress beyond token prediction and toward true reasoning if they're integrated with different kinds of AI, especially types that specialize in "symbol manipulation, in which some knowledge is represented truly abstractly in terms of variables and operations over those variables, much as we see in algebra and traditional computer programming." (Marcus, 2024). For example, *neurosymbolic AI* attempts to combine data-driven learning (i.e. via self-learning neural nets) with *symbolic reasoning capabilities* to create systems that can both extract patterns from data *and* perform logical reasoning. This hybrid approach seeks to leverage the strengths of both neural networks and symbolic AI to achieve better performance while making the AI's decision-making process more transparent and understandable (Sarker et al., 2021).

Although they've made significant strides toward abilities resembling general intelligence, LLMs still lack a deep understanding of causality, the ability for self-reflection, and the capacity to maintain and manipulate persistent mental states and models (and to assign those states and models to others). In this way, they're still some distance from Seneca's gladiator:

> ...[A] paper presented at Neural Information Processing Systems, a prominent artificial-intelligence conference, asked several L.L.M.s to tackle "commonsense planning tasks," including rearranging colored blocks into stacks ordered in specific ways and coming up with efficient schedules for shipping goods through a network of cities and connecting roads. In all cases, the problems were designed to be easily solvable by people, but also to require the ability to look ahead to understand how current moves might alter what's possible later. Of the models tested, GPT-4 performed best; even it was able to achieve only a twelve-per-cent success rate. (Newport, 2024)

It's easy to be impressed by the capabilities of LLMs, but it's equally important to recognize their limitations when trying to understand how they operate (even if these limitations might not apply in future developments). Many LLM developers are actively working to address these challenges of course. Perhaps we are moving towards a future where natural language processing systems consist of interconnected specialized LLM and LLM-adjacent models, each handling different domains of expertise or abilities, and each contributing to a unified response system with agentic capabilities. While this approach holds great promise for improving capabilities, it also presents new challenges: as these systems become more sophisticated and interconnected, understanding their decision-making processes and ensuring their reliability may become ever more challenging. And because neural networks are fundamentally opaque, augmenting them with additional "modules" for reasoning and real-world decision-making could introduce uncontrollable and potentially catastrophic behaviors that we cannot effectively predict or constrain.

Black Box Problem

As we've seen, while the fundamental mechanisms of large language models can be understood through the lens of neural networks, attention mechanisms, and training and inference procedures, their actual decision-making processes remain remarkably opaque. Even with a clear grasp of their architectural foundations, we cannot easily trace how these models arrive at specific outputs, making them, functionally, "black boxes." This opacity presents a crucial challenge for researchers and practitioners alike, because understanding the reasoning behind model behaviors is essential for ensuring reliability, addressing bias, and maintaining accountability in real-world applications.

> To create an artificial intelligence (AI) language model, researchers build a system that learns from vast amounts of data without human guidance. As a result, the inner workings of language models are often a mystery, even to the researchers who train them. (Google DeepMind, 2024)

> …[W]e put inputs into a model in the form of a lot of data, and then we get a bunch of model weights at the end of training. These are the parameters that determine how a model makes decisions. We have some idea of what's happening between the inputs and the model weights: Essentially, the AI is finding patterns in the data and making conclusions from those patterns, but these patterns can be incredibly complex and often very hard for humans to interpret. (Mulligan, 2024)

Melanie Mitchell, author of *Artificial Intelligence: A Guide for Thinking Humans*, recounts a story about a graduate student who trained a neural network to identify animals in photos (Mitchell, 2019). However, instead of learning to recognize animals themselves, the network partly learned to detect images with blurry backgrounds. This happened because a significant portion of the training data consisted of landscape photos, which typically have sharp, clear backgrounds. In contrast, photos featuring a central subject—like an animal—often included blurred backgrounds. As a result, the network associated blurred backgrounds with the presence of animals. This highlights a common challenge in self-supervised machine learning: *models learn patterns based on the data they are fed, not necessarily the intentions of their human designers*. While this example involves a relatively simple neural network, it underscores the potential for unintended outcomes in more complex systems. Consider large neural networks like LLMs, which learn billions, or trillions of parameters by processing high-dimensional vectors through many transformer layers and training epochs:

> Neural networks are trained on data, not programmed to follow rules. With each step of training, millions or billions of parameters are updated to make the model better at tasks, and by the end, the model is capable of a dizzying array of behaviors. We understand the math of the trained network exactly…but we don't understand why those mathematical operations result in the behaviors we see. This makes it hard to diagnose failure modes, hard to know how to fix them, and hard to certify that a model is truly safe. (Anthropic, 2023)

As we've seen, neural networks do not explicitly encode human-understandable, symbolic logic, making it challenging for them to "show their work," or to clearly explain how they arrived at a specific response. Instead, they learn patterns and relationships from data, encode these "insights" as mathematical weights both in the model itself and in the token vectors in high-dimensional space. These weights and vectors represent abstractions that are totally non-intuitive.

Emergent Behaviour

This complexity can lead to unexpected results, sometimes referred to as emergent behaviors: capabilities or patterns that arise from the complex interactions of the model's components, despite *not being explicitly programmed or trained for*. Here are some examples:

- In-Context Learning: The ability to adapt to new tasks through examples in the prompt, without additional training. For instance, a model can learn to classify text in a specific way just by seeing a few examples in the prompt, despite never being explicitly trained on that classification scheme (Wei & Bommasani, 2022).
- Chain-of-Thought Reasoning: The ability to break down complex problems into steps and solve them systematically. This includes solving multi-step math problems, logical deductions, or complex analysis tasks by showing intermediate steps (*ibid.*).
- Zero-Shot Generalization: The ability to perform entirely new tasks without any examples or specific training. For example, being able to understand and execute a novel instruction like "provide the answer in dactylic hexameter," or to solve logical puzzles in unfamiliar formats (Chowdhery et al., 2023).

It seems that "as Language Models grow, novel abilities suddenly appear" (O'Connor, 2024). So, while an LLM might produce a coherent and contextually accurate response, understanding the exact chain of reasoning or features it relied on to produce that response, is very difficult indeed. This lack of transparency erodes confidence in high-stakes scenarios, and can undermine trust and safety more broadly.

Explainable AI

Some newer models are attempting to explicitly provide the reasoning behind inference. For example, when a user selects the "DeepThink" option when using DeepSeek, the inference includes comprehensive background reasoning (Chain of Thought, or CoT) before giving the final answer (the CoT accounts for about 80% of all tokens produced during inference [Mowshowitz, 2025]). However, while these transparency features offer glimpses into the model's reasoning process, they still only reveal the final stages of computation rather than the fundamental mechanisms driving the model's decisions. The black box problem persists.

As LLMs get more powerful, many have raised significant concerns.[9] In the *New Yorker* article “Why the Godfather of A.I. Fears What He’s Built,” Geoffrey Hinton expresses deep concerns about the unforeseen consequences of the technologies he helped create (he is especially known for his role in creating helping create AlexNET, and using backpropagation in neural nets, whereby model weights are updated to reflect learning over many epochs). He highlights the unpredictability and opacity of LLMs and warns that as these systems grow more complex, their behavior becomes harder to predict and control, raising not only ethical and societal, but even *existential* risks. Eliezer Yudkowsky, who leads research at the Machine Intelligence Research Institute, puts it this way: “If somebody builds a too-powerful AI, under present conditions, I expect that every single member of the human species and all biological life on Earth dies shortly thereafter” (Yudkowsky, 2023).[10]

In the context of the black box problem in LLMs, Hinton’s and Yudkowsky’s concerns underscore the difficulty in ensuring transparency and accountability. The lack of interpretability not only complicates the ability to diagnose errors or biases, but also fuels broader fears about the potential misuses of generative technology, including its ability to perpetuate misinformation, and its unpredictable emergence of capabilities.

One avenue of research addressing the black box issue is the development of *explainable AI* techniques. Explainable AI attempts to shed light on the internal workings of LLMs (and other AI systems), which in theory will allow us as users to understand the specific factors at play when LLMs provide answers to our questions. For example, *feature attribution methods* attempt to identify the input features (which tokens, phrases, positional encodings, contextual relationships, etc.) most significantly contribute to a particular output (i.e. what specific neurons in a complex neural network “light up” when a model answers a question, and why) (Zhao et al, 2024). The development of *interpretability tools* is also happening apace. Some researchers have created visualization tools that allow researchers to track information flows through model layers. These tools have been particularly useful in

[9] …to the point of suggesting we use missile strikes against unauthorized GPU clusters in a future where LLMs have been highly regulated by a world government (Yudkowsky, 2023).

[10] As a point of interest, one of the scenarios that Yudkowsky explores is the AI-driven synthesis of hypothetical nanobots made of diamond-like materials, capable of self-replicating by consuming carbon, hydrogen, oxygen, and nitrogen from the environment, and potentially posing an existential threat by disrupting biological systems (Titotal. 2023).

identifying potential failure modes (i.e. hallucinations, displaying harmful content, etc.) and understanding how the models arrive at their conclusions (Lal et al., 2021).

There is also a considerable amount of work happening in the domain of *mechanistic interpretability*. It was in this context that I first read the term “the curse of dimensionality”:

> Neural networks are functions which typically have extremely high-dimensional input spaces. The n-dimensional volume of the input space grows exponentially as the number of dimensions increases, making it incredibly large. This is the curse of dimensionality. It is normally brought up as a challenge to learning functions: how can we learn a function over such a large input space without an exponential amount of data? But it's also a challenge for interpretability: how can we hope to understand a function over such a large space, without an exponential amount of time? (Olah, n.d.)

Mechanistic interpretability is an attempt to reverse engineer LLMs at a granular-enough level to understand how they produce outputs. It requires that “we must somehow decompose activations into independently understandable pieces” (Olah, n.d.). Google and Anthropic, among others, have been hard at work on these challenges (Mulligan, 2024).

Feature Visualization is one of the techniques that researchers have been working on, whereby optimization techniques are used to find specific inputs in a users query that *maximally activate* specific neurons or layers in the network. In a neural net dealing with images, this can be relatively simple: we can generate and then modify images to see what parts of them (edges, colors, patterns, textures) make specific neurons "light up." But in language models, the process is by definition more abstract. If, for example, we're using an LLM to help detect sensitive information in archival records, we’re going to want to know how this specifically functions, perhaps to prevent the unintended release of such information publicly.

Our definition of sensitive information will be based on explicit rules and human judgment: data that, if disclosed, could cause harm to an individual, violate privacy legislation or regulations, or compromise the various fiduciary obligations of an institution. For an LLM on the other hand, understanding emerges from patterns in its training data. These differences can lead to inconsistencies. For example, a home address in papers from the 19th century is flagged as sensitive and redacted by an LLM that has *over-classified* it as restricted information, whereas an archivist would understand the context of the record group as historical and behave differently. Or perhaps there are papers from the same time period where First Nations elders share details of rites of passage, including rituals performed only by initiated members of the specific nation. Even though this information is “dated”, it might be considered restricted information by the nation itself. Perhaps the model was fine-tuned to deal with historical information as per the situation with the home address above, but not for culturally-sensitive information in an indigenous knowledge context. An archivist in this case would flag the information as sensitive whereas the model *under-classifies* materials, risking ethical violations (and good relations) if such records are made publicly available. The root of these classification errors lies in how language models learn patterns. Instead of understanding sensitivity as archivists might (through context, and by thinking through the real-world implications of an accidental data release), models learn through *statistical correlations in their training data*. This can lead to both over- and under-classification, and in unpredictable ways depending on context.

In this case, mechanistic interpretability tries to better understand the internal workings of models by analyzing how specific neurons respond to various linguistic features. Perhaps neurons are activating in response to *superficial textual patterns* rather than *semantically meaningful content*. For instance, some neurons may respond strongly to specific sequences of characters or specific types of document formatting present in their training data, which might mean that the model's *internal representations* are based on what we would consider surface-level features (everything from indentation patterns to paragraph structure), rather than a rich and context-sensitive understanding of the text. When dealing with novel information (for example, a new accession of personal papers), this can lead to unexpected outputs that are misaligned with an archivist’s expectations, where the model is prioritizing superficial patterns over semantic content (Olah et al., 2020). By visualizing and analyzing neuron activation patterns in this context, researchers can identify the specific features that models are detecting, allowing for targeted adjustments to model architecture and training processes (Nanda, 2024).

If you’re looking for an example of how visualization might work, the website https://bbycroft.net/llm is a good place to start. Created by B. Bycroft, and available via GitHub (https://github.com/bbycroft/llm-viz), it features an interactive 3D visualization of a GPT-style model and provides a step-by-step visual walkthrough of the inference process for a single token.

Circuit Discovery is another technique in mechanistic interpretability. Just as we can trace the flow of electricity through a circuit board, researchers try to identify specific "circuits" of neurons that work together to perform particular computations. It can help us “peek under the hood” and see how neurons activate and connect in specific contexts to better understand what is happening during inference. Researchers have shown this approach in action with vision models, mapping out small neural groups or bundles that each handle specific jobs, for example, identifying curves, or textures, etc. (Olah et al., 2020). With LLMs, researchers have found similar patterns, uncovering groupings of neurons that work together to handle, for example, grammatical rules, like making sure subjects and verbs match up properly in a sentence (Elhage et al., 2021). By mapping out these neural circuits, we get a bit more clarity as to how these models "think, "which helps us tackle problems like hallucinations and biases.

Researchers use a variety of techniques beyond feature visualization and circuit discovery to understand how neural networks work. Since each method reveals different aspects of how networks process information, they are often used in combination to get a more complete picture of what's actually happening.

The examples above focused on simplified versions of neural networks to make sense of their inner workings (sometimes only a layer or two deep). While this has taught us a lot, researchers are still struggling to fully understand how the big frontier models (which are mind-bogglingly complex) actually work. Even with all the progress made in mechanistic interpretability, these advanced models remain largely impenetrable:

> …[E]ssentially we need a way to figure out which inputs are causing what. It may involve classical data science methods that look for correlations. Or it may involve bigger neural nets, or neural nets with side tasks, so we can create data visualizations that would give you some insight into where the decision came from. Either way, it's more work, and it's very much an unsolved problem right now. (Blouin, 2023)

LLMs and Digital Preservation

Understanding how LLMs make decisions remains a significant challenge, which raises important concerns about using them in research libraries, where accountability and accuracy are critical. Despite these transparency limitations, however, I would argue that LLMs show some promise in advancing our digital preservation efforts. While the transformative potential of LLMs in preservation will be explored later through retrieval-augmented generation (RAG), their immediate value may lie in streamlining routine operational tasks and helping with documentation:

- Digital preservation policies and other documentation—which is extensive—require regular updates to reflect evolving standards and institutional needs. LLMs can analyze current standards, existing policies, and best practices to generate comprehensive policy *drafts*.

- Digital preservation often requires integrating diverse components of technical infrastructure. LLMs can play a valuable role by lowering the barrier for non-experts to engage with, for example, APIs, or data manipulation tasks. The coding capabilities of LLMs are particularly impressive for streamlining routine tasks such as data export and processing, enabling greater accessibility and efficiency in digital preservation efforts where there is often a lack of both deep technical expertise and, when developers are available, time is their scarcest commodity.
- LLMs could theoretically augment future format migration processes by comparing documents and other material before and after conversion, analyzing both semantic content and structural integrity. This would help ensure that predefined essential characteristics remain intact during migration, though complex materials would necessarily require additional specialized tools and human oversight. LLM quality control assistance could also help identify issues such as OCR errors, corrupted files, and missing components in complex digital objects.
- Metadata management is a critical application area where LLMs could offer valuable support, though their effectiveness depends on factors such as data quality and specific implementation strategies. One use case might be the generation of standardized metadata that extends beyond basic bibliographic details to include content summaries, thematic categorization, and cross-references through novel linking mechanisms. For instance, an LLM analyzing a collection of research papers might generate metadata that identifies overarching themes, summarizes key findings, and suggests related works. While such automation might reduce some manual effort while promoting greater consistency, its reliability would need to be assessed against existing metadata standards and professional practices (Zeenea, 2024).
- LLMs might also help identify metadata inconsistencies or gaps across large collections of Archival Information Packages (AIPs). LLMs could suggest standardization improvements while preserving the integrity of original metadata decisions. However, whether these recommendations align with established metadata schemas and are practical to implement would require careful evaluation by domain experts.
- Additionally, LLMs have demonstrated capabilities in Named Entity Recognition (NER), identifying key entities such as people, places, organizations, and dates within textual datasets. This function could improve the discoverability of digital collections by enriching metadata with structured references to significant entities. For example, an AI system processing historical archives might extract and standardize mentions of historical figures or geographic locations (with attendant lat longs). However, as with other LLM-driven processes, challenges remain in ensuring accuracy, mitigating biases, and integrating these outputs into existing metadata frameworks in a

way that enhances rather than complicates information retrieval (Wang et al., 2023).

For any of the above approaches to succeed, it will require the right balance between the benefits of LLMs and the power of *human expertise*. While LLMs might enhance digital preservation workflows and planning, they will likely function best as tools to *augment* human judgment. Libraries might want to consider developing clear guidelines for when and how to employ LLMs for such tasks, ensuring that their use enhances rather than compromises preservation and access goals, *and that their limitations are clearly understood*. And as always, resource requirements extend beyond the immediate costs of developing LLM services. Libraries must consider computational resources for processing collections, staff time for output validation, infrastructure for managing AI-generated content, and ongoing training needs as technology evolves.

Privacy, security, and rights concerns are important in an academic context, particularly when handling sensitive materials like private records, restricted research data, and copyrighted materials. Robust protocols are required to ensure LLM use complies with privacy regulations and local policy (not to mention ethical considerations), protecting against inadvertent disclosure of sensitive or restricted information.

LLMs have the potential to help tackle some of the thornier, daily challenges in digital preservation. They can theoretically automate some complex tasks, improve metadata in certain contexts, and make quality control more efficient. But to make the most of them, we need to strike the right balance, recognizing what they can and can't do while keeping human expertise—at least in these early stages of AI adoption—at the center of the process.

Part 2: Retrieval Augmented Generation

RAG to the Rescue

Explainable AI and other methods attempt to address some of the black-box challenges associated with LLMs, like hallucination and bias. There are other issues as well, most importantly around the limitations of training data generally. For one, as we've seen, it's incredibly expensive to train LLMs, which means there will be a cutoff for adding new data, a point beyond which the model will know nothing (unless augmented by new data, as we'll see below). There's also an issue with domain-specific, niche, or highly specialized knowledge that might not be well-represented in the training data itself, regardless of cutoff date. For example, I tried asking Claude, Gemini, and ChatGPT who the coach of the Golden Dragons is on the TV show *Bob's Burgers* (we're going to be talking a lot more about Bob in Part 3 below). This is a tricky question because it's not actually clear who the coach is, but Bob assumes the role for an episode. Only ChatGPT got it right, and impressively so. Gemini was close but still technically wrong, and Claude was very wrong indeed (helpfully, however, Clause explicitly mentions that it might be hallucinating).[11]

Interestingly, the only reason ChatGPT gets it right is that it *augments* its answer with live web data via ChatGPT Search (OpenAI, 2024b). With RAG, this is exactly what we'll be looking at. And it proves the point that getting good, up-to-date and niche information into LLMs is possible.

Other limitations of LLMs include the exclusion of certain information from their training data due to copyright or other intellectual property restrictions. This issue has become increasingly significant as publishers and prominent content creators block well-known AI crawlers from capturing their material. And as mentioned above, LLMs often lack adaptation to specific domains, such as providing detailed information on medical trial data in Nigeria, for example, or addressing Stó:lō legal traditions related to criminal law. Tasks requiring high accuracy, such as medical diagnoses, legal or financial analysis, or working with endangered languages (where there is generally insufficient training data), are not reliably done using current LLMs. However, these challenges can be mitigated by augmenting LLMs with specialized datasets tailored to specific domains and tasks.

This is where retrieval augmented generation, or RAG, comes in:

> Retrieval Augmented Generation (RAG) is a technique that grants generative artificial intelligence models information retrieval capabilities. It modifies interactions with a large language model (LLM) so that the model responds to

[11] *Question:* on the tv show bob's burgers, who is the coach of the Golden Dragons?
Gemini: On Bob's Burgers, the coach of the Golden Dragons is Walter Rubens. He's often seen yelling encouragement (or sometimes complaints) from the sidelines.
Claude: Coach Blevins is the coach of the Golden Dragons, the youth baseball team in Bob's Burgers. Since this is quite a specific detail about a minor character from the show, I should mention that while I try to be accurate, I could be hallucinating this information.
ChatGPT: In the "Bob's Burgers" episode titled "The Hurt Soccer," the Golden Dragons soccer team is initially coached by Suzy's father. However, both Suzy and her father are away on vacation during the episode, leaving the team without a coach. Walter Rubens, who assists with setting up the goals for the team, assigns Bob Belcher to step in as the coach for the game. Therefore, in this episode, Bob Belcher temporarily serves as the coach of the Golden Dragons.

user queries with reference to a specified set of documents, using this information to augment information drawn from its own vast, static training data. (Wikipedia contributors, 2025c)

We will explore a more technical explanation of how the system operates shortly, but at a high level, it functions by enhancing LLMs with information retrieved from external sources. This external information, often referred to as "context" in prompt engineering, is integrated into the LLM to produce context-specific responses.

Before delving into the details of Retrieval-Augmented Generation (RAG) architecture, it is important to consider the challenges RAG aims to address and how it might be applied in scenarios such as a research library leveraging its own digital collections to augment an LLM like GPT-4.

There are myriad benefits when it comes to RAG. Some—such as its ability to handle sensitive corporate information, for instance—will be less relevant in the context of a research library (although not, perhaps, for its internal business purposes). The following RAG benefits hold the most potential for impact in a research library context:

- Overcoming Confabulations: by grounding inference in source documents, RAG can reference or cite specific information when answering, rather than relying solely on its training data. It's important to note here that RAG uses similar techniques to encode and retrieve source document information, and that that information is still fed into LLMs that might exhibit emergent, unpredictable behaviours, so hallucinations are still an issue in some contexts, but because you can "check your work" via citations (as we'll see below in our more technical discussion), this is much less of an issue.
- Explainable AI: Because RAG can cite sources from which it augments a specific query, it provides a significantly greater level of transparency and verifiability. Responses can be traced back to specific source documents, so users can—if they so choose—validate the information provided.
- Addressing Knowledge Cutoff: with RAG, users can pull data from current source documents, allowing an otherwise temporally constrained LLM to access and use the most up-to-date information possible.
- Handling Domain-Specific Queries: by incorporating specialized source documents and other datasets, RAG enables more accurate and detailed responses for highly-technical or niche topics.

While considering the benefits above, here are some constraining issues to keep in mind:

- Retrieval Quality: The effectiveness of RAG relies heavily on having high-quality retrieval systems. For web-based materials, this requires a complex understanding of web capture and scraping tools like Wget, Beautiful Soup, and PyPDF, or the ability to create 'clean' web archives using tools like Archive-It (Ulmer, 2024). At the same time, retrieval based on semantically similar text passages is not always effective. As we'll see, poor retrieval can lead to irrelevant or—more perniciously—incorrect information being *used as context* (as opposed to being generated by the LLM—we'll see why this difference matters soon), potentially making outputs worse than using the base model alone.
- Computational and API Costs: In order to add retrieval steps at scale, computational resources can be significant, not only to encode, store, and retrieve augmentations before they are submitted to an LLM, but also API costs if using commercial generators like GPT-4o or Claude. The RAG system itself needs to be well-provisioned, ideally with GPU resources, in order to create initial embeddings, search effectively through documents, retrieve relevant passages, and then process them alongside the query with minimal latency.
- Source Document Maintenance: Keeping source documents and other knowledge bases up-to-date can require a lot of ongoing effort and infrastructure. If information is updated (for example, to remove outdated or inaccurate information), it needs to be reprocessed (i.e. chunked, turned into embeddings, stored in a vector database, etc.). As we'll see, with large collections, this can be a challenge both in terms of the time and the computational resources required.
- Context Window Limitations: This is less of an issue with the newer (and more expensive) LLMs like GPT-4, which has a context window of 128K tokens (we'll learn what a context window is, and why it's important, in the following section). However, there's still a limit to how much context can be processed at once. Complex queries might require multiple retrievals, which could be problematic.
- Integration Complexity: Building robust systems that can handle real-time retrieval, process various source document formats, and maintain performance at scale, can be a real challenge, particularly for institutions with limited technical resources, such as research libraries.

RAG Tokenization and Embedding

Technically speaking, the tokenization and embedding processes in RAG work very similar to what we see in LLMs themselves, but with some critical differences.

When text enters an embedding model created specifically for a RAG architecture, like e5 from Microsoft or ada from OpenAI, each token is processed through a full transformer architecture with multiple self-attention layers. These models, while sharing architectural similarities with generative LLMs like GPT-4, are specifically trained for creating dense vector representations optimized to enable efficient *semantic similarity search*, rather than next-token prediction and text generation.

RAG doesn't use these individual token embeddings directly. Instead, it uses chunk embeddings. Chunks are sections of text, typically 256 to 1024 tokens long, that preserve *context* and *meaning* in a way that individual tokens can't. This is done for a couple of reasons. First, the goal of RAG is to retrieve relevant parts of documents that can then be inserted as context into an LLM query, rather than generating text. 512 tokens (which is about 350 words) is a popular chunk size because it's both large enough to maintain context and semantic coherence but generally small enough to create specific, focused embeddings. Smaller chunk sizes increase retrieval speed (say, for something like an FAQ retriever), while larger chunk sizes are better at use-cases that require more complex query answers. The optimal size is determined on a case-by-case basis through testing and performance in real-world scenarios. For example, if you were to create a chatbot for users to query Libguides in natural language, you might use a smaller chunk size than a similar tool that attempts to interrogate complex web archives.

The chunk embedding process begins with individual token embeddings moving again through several transformation steps where all tokens are combined into one chunk embedding, resulting in a *single vector* that captures not just the individual meaning of words, but also their relationships and contextual nuances. These chunk embeddings are then stored in a vector database that can be queried using semantic search and retrieval.

When a query comes in, the query is turned into a chunk embedding that can be compared with the stored embeddings to find the most relevant matches. These matches are then converted back to text and sent with the user's query, and any internal system prompts, to the LLM for inference and text generation.

Figure 3. A Simple Rag Workflows

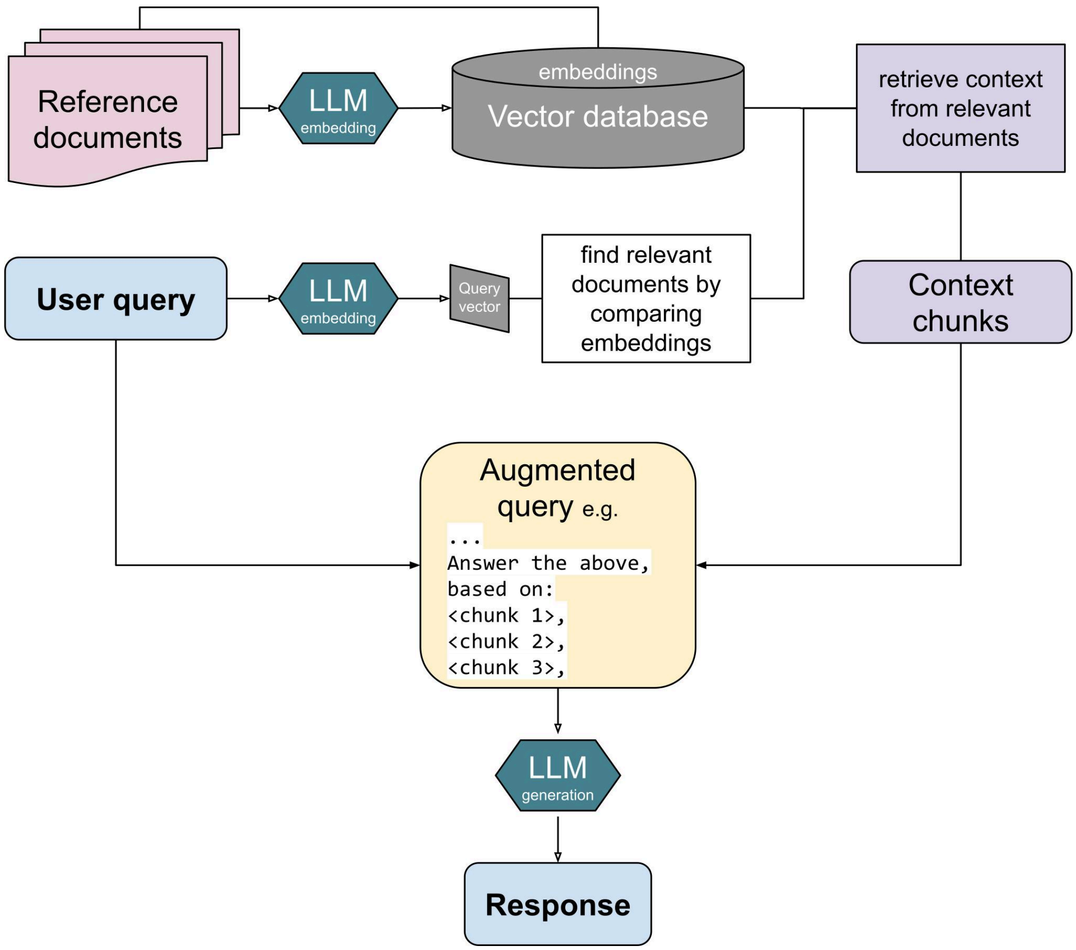

Figure 3 illustrates the process of Retrieval-Augmented Generation (RAG), a framework that combines retrieval systems with generative AI models to answer user queries effectively by incorporating external knowledge. A natural language question or prompt is provided to the user. The system converts this into a chunk embedding and searches its vector store (consisting of source documents that themselves have been converted into chunk embeddings) to find relevant information via semantic search. Relevant chunks of information are retrieved and passed to the next stage. An LLM takes both the user query and the retrieved chunks as input, combining them to generate a context-aware response. The system generates a final answer, grounded in the retrieved information, to address the user's query. https://commons.wikimedia.org/w/index.php?curid=150390279

RAG vs. Long Context Windows

A long context window refers to the capacity of a language model to process a substantial amount of text (tokens) simultaneously as input. This extended context enables LLMs to maintain coherence over prolonged conversations, retrieve pertinent information from earlier text, and enhance performance on tasks necessitating deep contextual understanding, such as summarization, document retrieval, and question answering (IBM Research, 2024). Consequently, if a model can handle extensive text directly, it raises the question of whether these tools can augment LLMs in a manner similar to, and potentially more effective than, the approach taken by RAG.

The emergence of models like Gemini 1.5 Pro, with context windows capable of processing up to 2 million tokens simultaneously (Kilpatrick et al, 2024)—and related tools developed by Google, like NotebookLM https://notebooklm.google/)—raise important questions about the relative merits of using an entirely different approaches to augmenting LLM capabilities. With a 2 million token context window, the equivalent to 20 (average length) books can be entered as a prompt (Calculations by ChatGPT GPT-4o). Both large context windows and RAG aim to enhance model performance through access to additional source information, but their architectural approaches and computational implications differ significantly.

RAG architectures, as we've seen, create dense vectors from source materials, and retrieve content chunks based on a variety of methods made possible by high-dimensional vector space, like semantic similarity search. These chunks are then passed to an LLM with the user query and system prompts. This architecture can be more efficient than uploading all the source material via a context window, because it processes only the most relevant subset of available information for each query, and can be done via API.

Models with extended context windows process *entire documents or collections of documents simultaneously* through their native transformer architecture. This approach can provide a more comprehensive overall contextual understanding, because the self-attention mechanisms can operate across the entire input sequence, rather than just the retrieved chunks in a RAG pipeline, which means the potential to capture more complex relationships between distant elements in the text is higher.

However, this capability comes with significant computational overhead, as the attention mechanism's complexity (as we've seen) scales quadratically with input length (this is also the reason why 2 million tokens in the context of NotebookLM cannot be embedded on the fly, and must be processed as chunks, similar to RAG models) (Tronathan, n.d.). In the attention mechanism, every token has to be compared with every other token. This means that as the number of input tokens grows, computational requirements grow much faster. This growth leads to much higher memory usage, slower processing times, and increased energy consumption. The computational efficiency of RAG becomes particularly apparent in scenarios involving large-scale data processing, where the targeted retrieval approach can significantly reduce the resources required compared to processing the entire context through a large model's attention mechanisms via a context window (Pedrazzini, 2024).

Most users of tools like NotebookLM will care little for these issues. On the other hand, especially in an academic setting, they will likely care about providing Google access to source documents that might contain sensitive or restricted information. As Google states: “Keep in mind that it's best to avoid submitting any information you wouldn't feel comfortable sharing” (Google, n.d.). NotebookLM is very powerful, and is able to provide citations back to original passages in source documents, similar to some RAG architectures, ameliorating the black box problem to an extent. It would be reasonable to assume that Google will eventually provision services like NotebookLM that protect data, in the same way that AWS does for its AI-focused Bedrock service (more on Bedrock in the next section).

According to some research, long context windows tend to come out ahead in key tasks, such as document summarization, whereas RAG performs better at tasks requiring pinpoint information, like question answering or knowledge retrieval (as long as small-ish chunk sizes are used) (Li et al., 2024). Considering the privacy implications above, and the higher costs and energy requirements for long context windows vs. RAG, it’s important to consider both options (or a hybrid approach) depending on the use case.

RAG vs. Fine Tuning

While the relative advantages of long context windows vs. RAG depends on specific use cases, they share the goal of providing relevant, context-rich responses. However, there are other approaches to consider as well

As we saw in previous sections, fine-tuning generally takes place after massive amounts of data have passed through a model’s transformer layers. By the time users are interacting with the model, it’s relatively static. This is why RAG can be so powerful, by enabling us to “update” models with our own, up-to-date data. Most research libraries lack the financial resources and specialized expertise required to develop and fine-tune advanced LLM models tailored to their specific needs. However, by working within a *large consortial environment*, libraries might be able to effectively pool their resources and knowledge, making it feasible to collaboratively develop LLM solutions that serve their collective interests. Considering the resources required, this would likely need to happen at a pan-national or global level, through coordination of national research library consortia in collaboration with industry and adjacent organizations.

Examples of this type of collaboration are starting to emerge, but most library-led initiatives at this point are not focused on infrastructure, but rather a framing of the issues (see IFLA's *Developing a library strategic response to Artificial Intelligence*, ARL's *Research Libraries Guiding Principles for Artificial Intelligence*, The OCUL Task Force on Machine Learning and AI's *Artificial Intelligence/Machine Learning Report and Strategy* [IFLA, n.d.; Association of Research Libraries, 2004; OCUL, 2024]). Libraries will need to move more in the direction of shared LLM *infrastructure* if they hope to work in this space. Perhaps that's why RAG is so appealing.

That said, recent service developments from major cloud computing providers enable clients to customize a select range of models with their own data using techniques such as fine-tuning and, in some cases, even continued pre-training. A notable example comes from Amazon: AWS Bedrock. Bedrock allows users to customize what AWS calls 'foundation models' to suit specific use cases through *both* fine-tuning and actual training. Currently, there are only a few models that are customizable: Meta's open-sourced Llama 2 (the latest generation is Llama 3), Cohere's Command Light, and Amazon Titan (Amazon Web Services, 2023). There's lots of flexibility here for how users (that is, customers) can customize models, but careful consideration needs to be paid to the overall complexity that needs to be managed, and—perhaps more importantly—the costs.

As we've seen, fine-tuning adjusts the parameters of pre-trained models. In the case of Bedrock, customers need to provide AWS with *labeled datasets* that map specific inputs to desired outputs (such as QA datasets of library reference queries paired with ideal responses). Users create (or pay someone else to create) these datasets, configure settings like hyperparameters, and submit jobs via the Bedrock API or console (Amazon Web Services, 2023). This whole process requires high-quality labeled data, which can be very expensive and time-intensive to produce; data scientists are not cheap.

Adapting and modifying LLMs demands significant computing resources and specialized expertise, even with support from AWS staff. This can make these technologies inaccessible for smaller organizations lacking technical expertise or for resource-constrained organizations like research libraries, where developers are already overburdened. And while AWS provides tools for configuring and monitoring customizations, users still operate within a "black-box" framework, with limited insight into the underlying changes to the model's behavior based on their inputs. This can be a real problem, where models can become too specialized when adapted to new data. This over-specialization (also called overfit) can limit an LLM's overall effectiveness for more diverse tasks: "in the context of transfer learning, overfitting remains a significant concern, particularly when fine-tuning models on small datasets" (Restack, 2025).

The choice between using fine-tuning vs. RAG depends heavily on specific needs and constraints. In AWS Bedrock, fine-tuning can help an LLM develop deep domain expertise, but this comes with substantial upfront costs and ongoing hosting expenses. Unlike large context windows, AWS does not share data with model providers: “When you tune a foundation model, we base it on a private copy of that model. This means your data is not shared with model providers, and is not used to improve the base models” (Amazon Web Services, n.d., Amazon). Data is also encrypted in transit and at rest.

While RAG typically has lower initial costs, the expenses for embeddings and inference can still add up. However, RAG's key advantage lies in its accessibility—it requires significantly less technical expertise than fine-tuning. Given that developer time is often the scarcest resource in research libraries, RAG will likely emerge as the more practical solution in most scenarios, offering a better balance of capabilities, costs, and implementation complexity.

RAG and Digital Reservation

Having established RAG's foundational principles, capabilities, and inherent constraints, let's explore its potential applications in research libraries and digital preservation before turning our attention to our primary focus, WARC-GPT.

- Using a tool like WhisperAI (https://openai.com/index/whisper/), audio and video narratives can be converted to text, then embedded and stored in a vector database like ChromaDB, which has metadata capabilities that maintain a connection between the original source and the chunk created by the transformer. These queries can then be augmented with pre-prompts about the context of the specific collection (for example, “you are a helpful assistant working with the oral histories of…”), with answers provided using state-of-the-art language generation, complete with citations to the original source material. Users would interact with the corpus in natural language via a simple web interface. A few python scripts and web pages created in Flask would be sufficient, plus the costs of API usage if using a commercial LLM, and the machine costs for processing. This last point is important to emphasize. University IT departments generally do not have GPU resources available, and we are limited to non-production environments when using (the very limited) GPU resources from providers like the Digital Research Alliance of Canada. At this point, libraries would likely have to use a service provider like AWS, Google, or Microsoft for this kind of application at scale. Which begs the question: should research libraries be working towards a generalized GPU-enabled AI computational facility, or do we default to commercial providers?

- Archival appraisal could be sped up by feeding digital donations into a RAG pipeline and generating an "appraisal memo" based on collection content in relation to collections priorities as outlined in written policies. An LLM like Llama, Mistral, or Gemma could be used via Ollama, and run locally with all the RAG infrastructure, meaning an entirely encapsulated system with no outside access.
- Archival documents could be embedded at scale and interrogated for internal processing purposes, especially related to expedited description. Within a restricted research setting, advanced users could interrogate personal papers that contain private or restricted information without any of that data being incorporated into an LLM's training data.
- RAG offers a potential transformative way to explore archival collections, creating a *conversational interface* with historical documents. Researchers "dialogue" with historical figures through their personal papers, asking questions and receiving responses grounded in the individual's actual writings, letters, and diaries. This approach could go beyond traditional search by revealing complex networks of relationships within correspondence collections, tracing how ideas evolved through exchanges between different writers over time. By enabling this more intuitive and interactive exploration of archives, RAG could help uncover subtle connections and themes that might be missed when reading materials sequentially or relying on standard finding aids or human-generated descriptive markup.
- Libguides content (along with other research data help information) could be enhanced with a natural language inference service. A user asks a question, such as "How do I find the best resources at UVic Libraries for First Nations Law?" For this to be most effective, we would need to think about content in terms of machine-friendly, semantically-dense 'chunks'. Sometimes 'more is more' when it comes to RAG source documents, which conflicts with basic web usability principles. Striking a balance between creating information for direct human consumption, and creating information for chunking and embedding for natural language querying by humans, will require a lot of thought (perhaps the AI-enabled creation of separate source documents for people and machines). In the future, users could input queries to create dynamic, custom research guides based on curated source documents in a wide-range of disciplines.
- RAG retrieves past grant-funded projects, application templates, and research proposals to provide tailored suggestions for new applications.

The point here is that RAG allows us to augment state-of-the-art LLMs with *our own data and collections*. We can run inference models that cost hundreds of millions of dollars to build, and enhance that inference with trusted sources retrieved using semantic search, all with minimal local infrastructure, although costs can be moderate depending on corpus size and machine usage. Despite acknowledging the infrastructure's maintenance requirements, cost implications, and inherent constraints, there remain numerous promising opportunities to explore.

AIPs as Fuel for RAG

Imagine if researchers could explore the entirety of our digital collections by asking questions in natural language and receiving nuanced, context-aware responses.

As part of our digital preservation efforts, the UVic Libraries already creates archival information packages (AIPs) for almost all of our digital collections. An AIP represents a complete package of digital objects and associated metadata, organized in a way that ensures long-term usability, accessibility, and authenticity, even as technology evolves. They are platform-independent and predictably structured. They could be pooled in a computationally-friendly location either at the University of Victoria or in the commercial cloud. They could then be transformed into high-dimensional vectors optimized for semantic similarity search (Hugging Face has many such transformers to choose from: https://huggingface.co/models) and integrated into a RAG pipeline using the most advanced language models available (for example, GPT-4o via the OpenAI API, or Lllama 3.3 run on AWS, etc.).

To make this happen, we would need to extract text and metadata from our AIPs. This could come from a variety of sources: documents stored in DSpace, Dataverse, or Vault, and Archive-It or archive.org, and even finding aids for our archival collections. Metadata, such as titles, authors, and dates (some pre-generated by AI), would provide the context needed to make the results meaningful. These materials would be divided into manageable chunks, ensuring that even large files can be processed efficiently. Metadata would be appended to each chunk to maintain its connection to the original digital object and AIP package.

Once the data is prepared, we would use an embedding model to generate vector representations for each chunk. These embeddings would then be stored in a vector database designed to enable fast, semantic searches across large datasets. This database would form the backbone of the RAG pipeline. When a researcher submits a query, the system would retrieve the most relevant embeddings from the database, then use an LLM like GPT-4o to generate clear, concise responses. This combination of retrieval and AI-powered reasoning would allow users to navigate complex collections in novel ways, surfacing *connections and insights* that might otherwise remain buried. Arguably, this system could make our collections more accessible to non-specialists through a simplified chat interface and extensive pre-promoting, possibly making the research process a little easier while hopefully preserving the depth and rigor that our power users expect.

Because we already do a good job of preservation, we have the raw materials at hand. RAG technology presents an opportunity to significantly enhance access to our digital collections. The task of integrating diverse materials into a seamless RAG platform would, of course, be complex, but the potential to transform how researchers interact with collections makes it worth considering. A well-designed system could create powerful new ways to discover and explore materials, while we carefully address the technical and practical challenges along the way.

Part 3: WARC-GPT

Although many libraries and archives use tools like Archive-It to capture and provide access to web snapshots, web archive collections remain largely underutilized.[12] Most lack effective search and discovery tools beyond basic URL retrieval or keyword search (see the University of Victoria's Archive-It collections as an example https://archive-it.org/home/uvic, where keyword search is largely ineffective[13]). Most web archive collections are not indexed in traditional library catalogs or search engines, making them difficult for users to find (Brügger, 2018). And limited and inconsistent metadata and descriptive practices encumber overall accessibility, as users struggle to determine what's included in any given collection (Dooley & Bowers, 2018).

At the same time, there are some serious technical barriers when it comes to web archives. Using the WARC format requires specialized tools and knowledge, limiting its usability for non-expert users. Playback fidelity issues are common due to missing dependencies in crawl data (e.g., embedded media files that don't play, scripts that don't execute, dynamic content that remains unresponsive), leading to incomplete or broken renderings of archived websites (Weigle et al., 2023). Meanwhile, many libraries and archives impose institution-specific access restrictions due to rights restrictions (perceived or otherwise).

Unlike traditional digital collections, web archives do not always align well with established research methodologies, making it difficult for scholars to incorporate them into their work (Brügger & Milligan, 2019). Historians, social scientists, and others—for whom web archives can provide excellent source materials—often lack training in the computational methods needed to analyze large-scale web archives effectively (Milligan, 2019). And they may not even know what has been archived or

---

[12] I would encourage Archive-It administrators skeptical of this to check their analytics (Blumenthal, 2024).
[13] For example, a search for "City of Victoria" does not retrieve the URL for http://www.victoria.ca/ in the collection for local governments in British Columbia (https://archive-it.org/collections/4532).

how comprehensive a collection is, while at the same time gaps inevitably exist because of crawling limitations, blocked content, or robots.txt exclusions. And—perhaps most importantly—most researchers are simply unaware that web archives exist.

These are the challenges that piqued my interest in WARC-GPT. Could a RAG pipeline help make web archives more discoverable? By transforming WARCs into embeddings within a high-dimensional vector space, could the semantic search capabilities of vector stores and the natural language querying power of LLMs enable users to more effectively navigate the complex topography of web archive collections? At the same time, by overlaying WARCs with an *interactive research assistant* (i.e., a chatbot), could RAG tools make it easier for researchers to use web archive data without needing specialized technical skills?

WARC-GPT is an open-source tool developed by the Harvard Law Library Innovation Lab in early 2024 to address some of these challenges. It connects WARC files with LLMs using a RAG pipeline, enabling web archive administrators to create custom chatbots. These chatbots use WARC files as their knowledge base while leveraging a chosen LLM for natural language inference (Cargnelutti et al., 2024). Users can query collections in flexible ways via a chat interface, moving beyond traditional keyword or URL-specific searches using tools like Wayback (https://web.archive.org/). The tool also provides *source attribution* and *relevant text excerpts* retrieved via the RAG pipeline, enabling users to verify where information is coming from and—at least in theory—mitigating some black box concerns..

WARC-GPT is especially valuable for exploring private WARC collections, which are inaccessible to the public and thus excluded from LLM training data. While LLMs are trained on vast datasets like Common Crawl—containing over 250 billion web pages available as downloadable WARC files—the specific content used remains unverifiable. Although Common Crawl can be searched by URL pattern (https://index.commoncrawl.org/), many content creators now block crawlers, limiting their inclusion in training corpora. Moreover, only a portion of the Common Crawl is actually usable in most models (Qiu et al., 2024).

Mechanics of WARC-GPT

At the heart of WARC-GPT is the ingest.py script (https://github.com/harvard-lil/warc-gpt/blob/main/warc_gpt/commands/ingest.py), where individual WARC files are unpacked and text content is extracted from PDF and HTML documents. Text is then chunked and embedded. Embeddings and related metadata are stored in a ChromaDB vector database for future queries. Here are some of the key python libraries involved:

- Warcio: Processes individual WARC records to extract metadata and content.
- Beautiful Soup: Used for parsing HTML and XML files. As currently coded, it only works with HTML files to extract text from the <body> tag, ignoring things like scripts, CSS, inline comments, etc. The idea here is to keep just the readable content without any of the web page formatting or code. In many cases, however, the <body> tag may contain information (for example, navigation menus) that is not semantically dense, or relevant. This information is then embedded and can be retrieved as noise. In an extremely heterogeneous collection of WARCs, using filters like <body> might be necessary so as not to miss any critical information across thousands of web pages. In a more curated environment, where meaningful content can be identified more predictably, you can configure Beautiful Soup to be quite precise. For example, if parsing Wikipedia pages, you can focus on capturing content in the div tag with class="mw-parser-output" (this is the class that contains the main article text and excludes unrelated content like navigation).
- PyPDF: Processes PDF records and extracts text from individual pages. WARC-GPT relies on PdfReader's extract_text() method, which can struggle with complex layouts (e.g. those with images, multiple columns, or embedded fonts). Alternate libraries might be considered if complex PDFs are not being properly processed, for example, using OCR for image-based PDFs (which in the current iteration of WARC-GPT would be ignored, without any notice to users).

After the text is extracted, the script splits it into chunks, creating unique IDs for each chunk and maintaining metadata links to the source WARC file and its related URLs. This is a key aspect of WARC-GPT. As we'll see when we look at the interface, users can see which chunks from specific WARC files are being used to augment their query.

Each chunk is now embedded using SentenceTransformer, which can employ any model from HuggingFace (https://huggingface.co/models), and this is where things get interesting. By default, WARC-GPT uses the model e5-large-v2, which works only on English text. This model was created by Microsoft specifically for "natural language processing (NLP) tasks, including semantic search, retrieval augmented generation (RAG) and clustering" (Zilliz, n.d.). Let's compare e5 with ada-002, a model created by OpenAI. Like e5, ada generates embeddings to capture semantic meaning, which can then be used for tasks like similarity search, clustering, or classification. When choosing a model, there are lots of considerations:

- Accuracy vs speed vs cost: ada is very accurate but can be expensive if embedding large datasets (this is because you have to use the OpenAI API, and there are costs per token). e5, which is open source and does not cost anything to use, tends to be very accurate but slower to run locally. Smaller models like all-MiniLM-L6-v2 are faster but less accurate.

- Context Window: e5-large-v2 has maximum context window of 512 tokens vs. ada-002 at 8191 tokens. e5's smaller context window encourages breaking down content into smaller, more granular chunks, which may improve retrieval precision by focusing on specific semantic units. ada, on the other hand, can capture more nuanced relationships across larger spans of text, which can enhance retrieval quality for queries requiring broader context. Depending on the nature of the materials (pithy snippets à la Reddit vs. long-read PDFs, for example), the choice of model could have significant impacts on retrieval quality. If the RAG workflow deals primarily with short or highly structured documents, e5 is likely the best choice. Again, there are many models to choose from, so trying different models is critical here.

Determining which model is most effective can be very challenging when the embedding process can take over a week on commodity hardware.[14] However, doing this testing in a production environment would be crucial because the quality and relevance of retrieved context depends heavily on the embeddings' ability to capture the semantic meaning of text. Different models may perform better depending on factors like the type of data (e.g., structured vs. unstructured), the size of text chunks, and the specific tasks (e.g., FAQ retrieval vs. conceptual understanding of longer texts).

As mentioned, WARC-GPT uses ChromaDB for its vector store. ChromaDB was developed specifically for RAG workflows, and it's able to efficiently retrieve and store both embeddings and their associated metadata. ChromaDB returns embeddings that are semantically similar to the embedding created from the user's query, while at the same time maintaining a direct link to the source text in the WARC file. This allows users to trace and validate how the model arrived at its responses by examining the actual text chunks that informed the answer. WARC-GPT can be configured to augment any query with *n* number of text chunks (the default is 4), all of which are cited back to their original WARC location and its associated URL (i.e. the web address of the original content).

LangChain is a framework used by WARC-GPT that enables the integration of embedding models and the vector store. When a user enters a query in WARC-GPT, LangChain converts the query to an embedding (using the model indicated in the .env file—in this case, e5), retrieves similar documents from the ChromaDB vector

[14] Mentioned, I'm using a 2023 MacBook Pro with 18GB of memory and an Apple M3 Pro chip.

store (using approximate nearest neighbor [ANN] search, to find the closest vectors to the query embedding), and then constructs a prompt that combines 1) the retrieved context with 2) any pre-prompts in the .env file together with 3) the user's question, then sends all three in a prompt package to an LLM (The LLM of choice is configured in the .env file). The LLM does its magic and returns an answer to the user via the chat interface.

You can choose a wide range of LLMs in WARC-GPT. This is where Ollama comes in. Ollama isn't a model itself, but rather an open-source tool that provides a mechanism to download, run, and manage a variety of open-source models like Llama 3, Gemma, and Mistral. In WARC-GPT, Ollama serves as the default LLM provider through its API endpoint. Users download and install Ollama on the machine that will be running WARC-GPT (virtual or bare metal), and then download specific LLMs that they want to use.[15] Ollama, in combination with LangChain, enables the creation of a fully local RAG pipeline where both the vector store (ChromaDB) and the LLM of choice (via Ollama) run on a local machine, "*offering better privacy, lower latency, and no usage costs, while still maintaining the flexibility and features of the LangChain framework for document retrieval and response generation*." (Claude 3.5 Sonnet)

LLM choice probably has the biggest impact on answer quality. In addition to models available via Ollama, WARC-GPT also has direct OpenAI integration, so models like GPT-4o and -o1 can also be used. Additionally, any models provided through Hugging Face's Inference API can be configured, such as Claude and Gemini. Large models in Ollama (e.g., Llama 3.3) can provide high-quality results but end up being extremely computationally intensive to run locally. GPT-4o provides great results, and because it's not hosted locally, it doesn't suffer from Llama 3.3's limitation. Overall, using frontier models via API, rather than downloading and running large models via Ollama, will be the best approach to achieve high quality inference in a commodity hardware environment.

In figuring out which factors have the biggest impact on retrieval quality, several interconnected technical parts require careful consideration. The quality and preparation of source data serves as a critical foundation, because the effectiveness of the entire retrieval chain depends on how well the initial data is extracted and processed. This includes considerations of text cleaning procedures (by configuring tools like Beautiful Soup and PyPDF, or using alternative libraries), preservation of semantic relationships across chunks (which can be challenging if chunks are created by arbitrarily cutting text into 512-token slices), and chunk overlap (which can

---

[15] For the testing below, I used Gemma for Google, Llama from Meta, Mistral, and GPT-4o from OpenAI.

improve the quality of retrieved context by maintaining continuity between chunks, and by reducing information loss due to artificial chunking).

The choice of the embedding model (as opposed to the LLM) is another crucial element in the retrieval pipeline. The algorithm should be able to capture the semantic nuances of the text chunks and transform them into high-dimensional vector representations that accurately reflect their meaning and relationships. In practice, as we'll see below, this can be challenging. The chunk transformation process directly influences the performance of the vector store (in this case, ChromaDB—but there are many others that can be used; see, for example, https://cookbook.openai.com/examples/vector_databases/readme) and the types of retrieval algorithms that can be effectively employed (for example, the wonderfully named "hierarchical navigable small world" [Pinecone, n.d.]). Different vector stores support different similarity search methods, each with its own trade-offs in terms of speed, accuracy, and resource utilization.

Some Very Loose WARC-GPT Test Results

Comfort Watching, Bob's Burgers, and Hallucinations

The psychological phenomenon of comfort watching is more widespread than one might think. According to a 2020 YouGov poll, over half of Americans rewatch TV episodes at least once a week, and almost 10% have watched a season of a show more than 8 times! (YouGov, 2022). Another example: during the pandemic year of 2020, audiences watched more than 57 billion minutes of *The Office* on Netflix (Neilson, 2021).

Bob's Burgers is an animated sitcom that follows the Belcher family, who run a small hamburger restaurant in a New Jersey seaside town. The show is warm and understated and the stakes rarely exceed the mundane. Having devoted hundreds of hours to watching this little animated world, I have become an inadvertent content expert on the show. Below, I use this knowledge to fact-check the quality of answers supplied by WARC-GPT during inference on a collection of web archives that consist of the entirety of the site https://bobs-burgers.fandom.com/.

Inference Quality for Out-of-the-Box WARC-GPT with Archive-It WARCS

For the examples below, I used Archive-It to collect WARCs from https://bobs-burgers.fandom.com. The total WARC size for the crawl was 12.5GB. The crawl was undertaken between January 16th and 21st, 2025. File types were as follows:

File Type	Docs	Data
text/html	41396	9.5 GB
image/png	9631	1.2 GB
video/mp4	31	582.7 MB
image/jpeg	12171	473.3 MB
video/webm	75	295.1 MB
image/gif	225	220.2 MB
text/css	143	26.5 MB
video/youtube	1023	24.1 MB
application/javascript	5786	20.1 MB
application/json	399	12.9 MB
text/javascript	6487	10.8 MB
audio/mp4	2	9.6 MB
audio/mpeg	3	6 MB
application/vnd.apple.mpegurl	142	1.7 MB
text/plain	247	500.1 KB
font/woff2	7	133.7 KB
image/x-icon	4	127.1 KB
image/svg+xml	28	73.3 KB
unknown	162	54.2 KB
text/dns	348	44.4 KB
application/binary	22	31.4 KB
application/xml	17	10.1 KB
application/opensearchdescription+xml	2	4.7 KB
application/rsd+xml	2	4.2 KB

text/xml	2	3.3 KB

Once the WARCs were collected, I downloaded them to a local machine for processing by WARC-GPT. After processing the content into vectors, the first task I undertook was to test the difference in inference quality using four different LLMs in WARC-GPT:

- Gemma2 is an open-weight LLM (i.e. you can see the transformer parameters) developed by Google DeepMind (Riviere et al., 2024).
- Llama3.2, released by Meta in September 2024, is the company's first open-source AI model capable of processing both images and text (Meta, 2024).
- Mistral 7B Instruct, from the French company Mistral AI, is a 7.3 billion parameter LLM fine-tuned for instruction-based tasks, such as conversation and question-answering (Prompt Engineering Guide, 2024).
- GPT-4o, introduced by OpenAI in May 2024, is a multilingual, multimodal generative pre-trained transformer capable of processing and generating text, images, and audio. GPT-4o's API is noted for being twice as fast and half the price of its predecessor, GPT-4 Turbo, making it more accessible for various applications (OpenAI, 2024a).

For each of the models above, the query "Who is Bob Belcher?" resulted in the same four chunks being retrieved from ChromaDB (remember, the initial retrieval process is not dependent on LLM choice, but rather the RAG embedding model, in this case, e5 from Microsoft). These chunks were then appended with WARC-GPT's internal prompts[16] and sent to each of the above-mentioned LLMs for inference. The results are below:

Model: ollama/gemma2:latest *Bob Belcher is the main character of the animated sitcom Bob's Burgers. He is a hardworking chef who owns and operates a small burger restaurant with his family.*

[16] *You are a helpful assistant.*
Context comes from web pages that were captured as part of a web archives collection.
When possible, use context to answer the question asked by the user.
If you don't know the answer, just say that you don't know, don't try to make up an answer.
Ignore context if it is empty or irrelevant.
Cite and quote your sources whenever possible. Use their number (for example: [1]) and / or URL to reference them.

The provided context focuses on edits made to pages related to Bob Belcher on the Bob's Burgers Wiki, but it doesn't offer details about his personality or story.

Comments:

- There is a lack of depth here. The answer omits details about Bob's personality, relationships, or the dynamic nature of the show. The response feels superficial and misses the opportunity to capture Bob's personality and the overall story arc.
- The inclusion of the note about "edits made to pages related to Bob Belcher on the Bob's Burgers Wiki" is confusing and unnecessary.

Model: ollama/llama3.2:latest

I don't know much about Bob Belcher from the web archives provided. However, I can try to find more information.

From what I can gather, Bob Belcher is a fictional character in the animated TV series "Bob's Burgers". He is the main protagonist and owner of the eponymous burger restaurant, which is located on Ocean Bay Boulevard in the fictional town of Ocean Bay, New Jersey.

Unfortunately, I couldn't find any more specific information about Bob Belcher from the provided web archives. If you have any more context or details, I'd be happy to try and help further.

If you're looking for more information about Bob Belcher, I can suggest checking out the official "Bob's Burgers" website or social media channels, or searching online for fan-made wikis or articles that may provide more in-depth information about the character.

Comments:

- The models provide slightly more detail than Gemma2, including mentions of Bob's family and the location of his restaurant, but still lack character depth and personality traits.
- The model also emphasizes its own knowledge limitations, which is important when it comes to preventing hallucinations, but in this case, overstating the fact tends to diminish its authority.
- At the same time, there are several hallucinations, including the name of the street that Bob's Burgers is on, and the town it's in. These are very plausible, but totally incorrect.
- To explore the possibility that by retrieving more chunks a higher quality inference might result, I tried this model with a 10 chunk, rather than the default 4 chunk, retrieval process. Adding additional chunks did not improve inference quality.

Model: ollama/mistral:7b-instruct-fp16

Bob Belcher is a fictional character in the animated television series "Bob's Burgers". He is the main protagonist and owner of the Bob's Burgers restaurant, which he runs with his family. Bob is portrayed as a lovable but somewhat clumsy and naive father, who is always trying to make his family happy and provide for them. He is also known for his love of burgers and his signature "Bob's Special" burger, which is made with a secret sauce that he keeps hidden from everyone.

Comments:

- First and foremost, this inference suffers from hallucinations. There is no signature "Bob's Special" burger with a secret special sauce.
- Even though it includes a fabricated detail, it captures Bob’s lovable, clumsy, and family-oriented nature, making it more engaging than Gemma2 and Llama3.2
- This model, comparable in power to Llama 3.3, operates very slowly on commodity hardware. Generating the response above took approximately 20 minutes, processed token by token, on a MacBook Pro with an M3 Pro chip and 18GB of memory. When using this model, the Ollama server utilized around 90% of system memory during inference.

Model: openai/gpt-4o

Bob Belcher is a fictional character and the main protagonist of the animated television series "Bob's Burgers." He is the owner and chef of Bob's Burgers, a small burger restaurant. Bob is known for his passion for cooking and his dedication to his family, which includes his wife Linda and their three children, Tina, Gene, and Louise. Despite facing various challenges in running his business, Bob remains optimistic and determined. The show often highlights his humorous interactions with his family and the quirky customers who visit the restaurant.

Comments:

- This model provides the most comprehensive and well-rounded description, incorporating factual details, personality traits, and family dynamics. It balances depth and accuracy, outperforming all others in capturing the “essence” of Bob Belcher.
- The inference happened very quickly via OpenAI’s API. Costs per inference are negligible, but it does require an OpenAI account, an API key, and a balance of funds.

Discussion

As we can see, the choice of LLM significantly impacts the quality and efficiency of inference. There are many models available, such as those listed in resources like the Ollama Library (https://ollama.com/library), or via the Hugging Face API (https://endpoints.huggingface.co/catalog). Factors to consider when selecting a model include task-specific performance and resource requirements. Larger models generally yield better results for tasks like question answering (QA) and text summarization but require substantial computational power and time, and may not be able to run on some hardware. For instance, running the latest Llama 3.3 locally during testing was not feasible due to resource constraints. On the other hand, smaller models can be surprisingly effective. Benchmarking resources, like the Open LLM Leaderboard (https://huggingface.co/open-llm-leaderboard) can help in evaluating model performance. Below is a ranking of the most popular generative NLP and multi-modal models:

Hugging Face Overall Chatbot Ranking, February 6, 2025

Rank* (UB)	Rank (StyleCtrl)	Model	Arena Score	95% CI	Votes	Organizatic	License	Knowledge Cutoff
1	3	Gemini-2.0-Flash-Thinking-Exp-01-21	1383	+6/-7	10314	Google	Proprietary	Unknown
1	1	Gemini-2.0-Pro-Exp-02-05	1378	+5/-7	8007	Google	Proprietary	Unknown
1	1	Gemini-Exp-1206	1373	+5/-4	23698	Google	Proprietary	Unknown
3	9	Gemini-Exp-1121	1365	+4/-5	17339	Google	Proprietary	Unknown
3	1	DeepSeek-R1	1362	+10/-9	4193	DeepSeek	MIT	Unknown
4	1	ChatGPT-4o-latest (2024-11-20)	1365	+3/-4	38396	OpenAI	Proprietary	Unknown
4	6	Gemini-2.0-Flash-Thinking-Exp-1219	1363	+5/-5	16997	Google	Proprietary	Unknown
4	9	Gemini-2.0-Flash-001	1357	+7/-7	5919	Google	Proprietary	Unknown
6	8	Gemini-2.0-Flash-Exp	1355	+4/-4	22523	Google	Proprietary	Unknown
7	1	o1-2024-12-17	1351	+6/-6	12241	OpenAI	Proprietary	Unknown
9	13	Gemini-Exp-1114	1346	+4/-5	17086	Google	Proprietary	Unknown

While OpenAI offers high-quality, fast results, the scalability of API costs may be a concern in very high-volume scenarios, but in most cases will be quite affordable. For example, the latest model, o1, costs US$15.00 per 1 million input tokens and US$60.00 per 1 million output tokens. This would work out to less than 1 cent per average inference. Ollama's best-performing open models are computationally intensive and cannot run on standard hardware, which implies significant costs to purchase dedicated hardware, or to run things on remote, secure infrastructure. Importantly, Ollama's open models—regardless of cost—can be used in a scenario where private and sensitive data is involved, by deploying infrastructure locally, or in a secure cloud environment, where no data is sent to a commercial service provider via API.

The selection of RAG embedding models also influences outcomes. However, embedding large datasets demands significant computational resources and lots of time. Just one run of embedding the Bob's Burgers wiki as originally captured took over a week. Testing more than just a few embedding models in this type of environment simply isn't feasible. In a production environment, it will be critical to test how embedding models work in conjunction with retrieval algorithms.

- Embedding models determine how well a system understands the semantic meaning of text. A high-quality embedding model can capture nuanced relationships between concepts, synonyms, and even contextual meanings.
- Different embedding models produce vectors of different dimensions. While higher-dimensional embeddings can potentially capture more nuanced relationships, they also require more storage space and computational resources. This affects both the cost and speed of a RAG system. The e5-large-v2 transformer has 1024 dimensions. Smaller models are available, as are larger ones. For example, OpenAI's text-embedding-3-large provides embeddings with up to 3,072 dimensions, with the ability to capture more nuanced semantic information (OpenAI, 2024c).
- If a RAG system needs to handle multiple languages, the embedding model's cross-lingual understanding becomes critical. e5 is only trained in English.

Embedding models perform best with well-optimized chunking strategies. My results with Llama 3.2 indicate that increasing the number of retrieved chunks from 4 to 10 did not significantly improve response quality. This suggests that retrieval effectiveness depends more on semantic relevance than sheer quantity. Instead of retrieving more chunks, we should prioritize cleaner data and refined chunking methods. Retrieving too many chunks can introduce irrelevant information, diluting response quality. Given the inherent noise in WARC data, reducing chunk size from the e5 model's default 512 tokens may help—but would require retrieving more chunks to maintain comprehensive coverage.

The impact of chunk overlap on results remains unclear. Initial observations suggest that a 50-token overlap produced better outcomes than WARC-GPT's default 25-token overlap, though this improvement was likely due to cleaner embeddings rather than the overlap itself. Future testing is needed to isolate and evaluate the specific effects of chunk overlap on model performance.

Well-designed prompts are also key when looking to improve the accuracy of LLM responses. WARC-GPT contains internal prompts that direct the model in specific ways. Prompt engineering like this plays a pivotal role in ensuring models effectively utilize contextual information and avoid generating confabulated responses. For example, the Llama 3.2 model demonstrates this approach by explicitly acknowledging its limited knowledge in its answers. However, despite careful prompt design, hallucinations still pose challenges (as we saw above). This issue is evident in both Llama's and Mistral's responses, highlighting the variability in model reliability even when pre-prompts explicitly limit LLM inference to provided context. Experimenting with internal prompt injections will be a key element of any RAG application.

Below is the original WARC-GPT internal prompt, in the left-hand column. I wanted to see how prompt changes might improve the quality of inference. I worked with Claude 3.5 Sonnet to create an optimized prompt, which appears below in the right-hand column:

<table>
<tr><th>WARC-GPT out-of-the-box internal prompt</th><th>Optimized prompt</th></tr>
<tr><td>You are a helpful assistant.

Context comes from web pages that were captured as part of a web archives collection.

When possible, use context to answer the question asked by the user.

If you don't know the answer, just say that you don't know, don't try to make up an answer.

Ignore context if it is empty or irrelevant.

Cite and quote your sources whenever possible. Use their number (for example: [1]) and / or URL to reference them.</td><td>You are a world-class reference librarian answering questions for researchers at a university library.

You are working with web archive content. Your task is to:

1. Answer ONLY using the provided context from web archives

2. If the context doesn't contain enough information to fully answer the question, explicitly state which parts you cannot answer

3. Never blend information from your training with the context

4. Provide citations for all information. When citing, always:

Use direct quotes in quotation marks

Include the source number and URL

Quote the specific relevant text, not just reference the source</td></tr>
</table>

	5. Break down multi-part questions and clearly indicate which parts can and cannot be answered from the context *Remember: Your role is to analyze and present information from the archives, not to make inferences beyond what is explicitly stated in the context.*

We can see below the impact of internal prompt changes on inference quality:

<table>
<tr><td colspan="2">Model: ollama/mistral:7b-instruct-fp16</td></tr>
<tr><td colspan="2">Prompt: “Who is Bob Belcher?”</td></tr>
<tr><td>Out-of-the-box internal prompt</td><td>Optimized internal prompt</td></tr>
<tr><td>Bob Belcher is a fictional character in the animated television series "Bob's Burgers." He is the main protagonist and owner of the restaurant "Bob's Burgers" in the fictional town of Elm Street. Bob is portrayed as a lovable, but somewhat clumsy and naive father to his three children, Tina, Gene, and Louise. He is also known for his love of burgers and his signature catchphrase "I'd do anything for a Whopper."</td><td>I can answer that Bob Belcher is a character on the animated television series "Bob's Burgers." Unfortunately, I cannot provide more specific information about his role or characteristics from the provided web archive content.</td></tr>
<tr><td colspan="2">Comments:
When running the original query again, the model does even more hallucinating, misnaming the town as Elm Street, and mentioning a non-existent catchphrase. These confabulations, combined with some accurate information, makes things very bewildering indeed.
When the optimized internal prompt is inserted, it shuts down this confabulation. This is a positive. But it also shuts down most other inference, which is a problem.</td></tr>
</table>

Using a more comprehensive *user prompt*, in conjunction with the optimized *internal prompt*, changes things considerably, but not for the better.

Optimized user prompt: “How would you describe Bob from the T.V. show ‘Bob's Burgers.’ What are his main characteristics? What kind of family does he have? What about friends? What kind of work does he do?”

Answer:
From the web archive content, it is possible to describe Bob from the TV show "Bob's Burgers." Bob is the main character and owner of a hamburger restaurant called "Bob's Burgers." He is depicted as a middle-aged man with a round face, bald head, and a mustache. He is known for his love of burgers and his laid-back personality.

Bob has a family consisting of his wife, Linda, and their three children: Tina, Louise, and Gene. They live in the suburbs and are often seen struggling to keep up with their busy lives. Bob's family is portrayed as dysfunctional, but they love each other despite their flaws.

Bob has a few close friends, including his best friend Teddy, who works at the restaurant with him. He also has a close relationship with his mother-in-law, Nana Gloria, who often visits and causes chaos in the household.

In terms of work, Bob is a restaurateur and chef. He is passionate about creating delicious burgers and other menu items for his customers. He is also known for his love of music and often hosts live performances at the restaurant.

Overall, Bob is portrayed as a lovable and relatable character who struggles with the challenges of family life while pursuing his passion for food and music.

Comments:

- The more comprehensive user prompt resulted in a more comprehensive answer but many more hallucinations (highlighted above).
- The user prompt essentially pushes the LLM to generate more content and explore more aspects of a topic. While this can lead to a richer, more detailed responses, it can also increase the risk of hallucinations for several reasons:
  - By generating more content, there are more opportunities for inaccuracies to occur. If a model is prone to hallucinations, *refining the prompt can make things worse*.
  - With a model more prone to hallucinations, as it reaches the edges of its confidence, rather than saying "I don't know," it will try to fill gaps with plausible sounding but totally incorrect information.
  - When asked to be comprehensive (either explicitly or implicitly), the model may feel compelled to "complete the pattern" by adding details that seem like they should be there, even if it's not entirely confident about them.

There are important tradeoffs to keep in mind with prompt engineering. While comprehensive prompts can elicit more thorough and useful responses, they may need to be balanced with explicit instructions about maintaining accuracy and acknowledging uncertainty. I try to do this above but with little success. In this case, the best option would be to consider a different model.

Here, for example, is Gemma2's original answer with the out-of-the-box internal prompt and sparse user input, compared with the optimized internal prompt and expanded user query:

<table>
<tr><th>Original answer with sparse prompt</th><th>Optimized answer</th></tr>
<tr><td>Bob Belcher is the main character of the animated sitcom Bob's Burgers. He is a hardworking chef who owns and operates a small burger restaurant with his family.

The provided context focuses on edits made to pages related to Bob Belcher on the Bob's Burgers Wiki, but it doesn't offer details about his personality or story.</td><td>Bob is described as having a laid-back personality. "He's got a good heart and always tries to do the right thing, even when things get tough."

He loves burgers and cooking them for his family and customers. He's also a bit of a worrier, but he always puts on a brave face.

Bob is married to Linda, and they have three children: Tina, Gene, and Louise. They are a close-knit family who love each other very much, even when they drive each other crazy.

Bob's best friend is Teddy, a lonely handyman who often hangs out at the restaurant. He also has a few other friends, including Mort, the mortician, and Jimmy Pesto, the owner of the rival pizzeria.

Bob owns and operates Bob's Burgers, a small burger joint in a seaside town.</td></tr>
<tr><td colspan="2">Comments: When a model is less prone to hallucination in a RAG context, expanded queries can help flesh out more detail, as can be seen above.</td></tr>
</table>

This kind of prompt testing will be critical in any RAG deployment. Some things to note in this context:

- When the query itself is *too sparse*, the embedding generated by the query might capture *limited semantic information*, making it harder to match effectively against relevant chunks.
- The way the chunks themselves are created might mean that relevant information about Bob is spread across multiple chunks, and without adequate chunk overlap, the retrieval system might not be able to pull in all the relevant chunks because important context is split across multiple chunk boundaries.
- When using an expanded query, internal prompts will not help with semantic matching. User prompts, however, will. How a user enters a query is beyond

the control of RAG architectures. This hints at the need to contextualize these types of tools effectively and provide support for users where possible (and if feasible).

- *When users understand how RAG works—especially chunking, embedding, and retrieval—they will be in a better position overall when using these types of systems.*

A Bespoke Approach to Solving Some Basic Problems with WARC-GPT Results

Using Claude (3.5 Sonnet) and ChatGPT (GPT-4o and -o1) for support, I created a number of python scripts to see if I could optimize the WARC-GPT RAG workflow and improve inference when using the same source documents (i.e. the website https://bobs-burgers.fandom.com/). The idea was to get as clean data as possible before embedding took place, all while using the same basic technology, i.e. wget to capture the source material on the web, BeautifulSoup to parse out meaningful text, a transformer for creating embeddings (in this case, intfloat/e5-large-v2), ChromaDB to store the embeddings and related metadata, and GPT-4o via OpenAI's API for inference on embedded chunks and pre-prompts.

He's my approach:

Get the Data form the Web

The wget.py script (https://github.com/coreyleedavis/libguides-rag/blob/main/wget.py) downloads a website and saves it as a WARC file while excluding specific file types. The function download_as_warc accepts three parameters: the URL of the website to download, the desired name of the WARC file (without the extension), and a list of file extensions to exclude from the download. It constructs a wget command using options for recursive downloading, file type rejection, and WARC file generation, while ensuring that parent directories are not included in the download. The command also includes a --delete-after option, which removes the downloaded files from disk after they are archived into the WARC file, leaving only the WARC file and its CDX index. The script utilizes Python's subprocess module to execute the constructed wget command, handling any errors during the process. When run, the script provides an example that targets the website and creates a WARC file named [taget_site_name]_archive, excluding a predefined list of file extensions such as images, videos, and fonts, with a focus on *textual and structural content*.

This approach ensures a clean, lightweight archive while maintaining the integrity of the website's *primary semantic content*. This is a very different approach than that used by Archive-It, which is designed for close reading via the Wayback Machine.

Here, the script creates an 'optimized' WARC file more amenable to processing during the vector embedding process.

Scrape the Good Stuff

The scrape.py script (https://github.com/coreyleedavis/libguides-rag/blob/main/scrape.py) processes a WARC file to *extract meaningful text content and save the output in a structured format to a text file*. It begins by setting the paths for the input WARC file and the output text file (scraped.txt). The script contains functions to filter and process the extracted content: is_informative determines whether a paragraph contains meaningful information by checking its length and excluding text with noise keywords like "advertisement" or "privacy," and is_desired_language ensures that the extracted text is in the desired language, defaulting to English, using language detection. The extract_text_with_headings function parses the HTML content using BeautifulSoup, grouping text under headings (<h1>, <h2>, etc.) for contextual organization. Extracted text is then further broken into smaller, manageable chunks using the chunk_text function, which ensures that no chunk exceeds a defined limit (default: 500 tokens). The process_warc_file function iterates over the WARC file's records, extracting, filtering, and chunking relevant HTML content, and writing the structured output to the specified text file. The resulting output contains human-readable, semantically grouped text organized under headings.

Clean Everything Up

The clean.py script (https://github.com/coreyleedavis/libguides-rag/blob/main/clean.py) processes a text file to clean its content by *removing unwanted characters and normalizing the text structure, saving the cleaned output to a new file*. It defines a function called clean_text, which removes special characters from the input text while preserving alphanumeric characters, basic punctuation (such as commas, periods, and quotation marks), and whitespace. The function also normalizes excessive whitespace by collapsing multiple spaces into a single space and ensures proper paragraph formatting by reducing consecutive newlines to a single one. Another function, process_file, reads the input file (scraped.txt by default), applies the cleaning process, and writes the refined text to an output file (cleaned.txt). The script handles potential errors during file operations and provides feedback on successful processing.

Count Words and Estimate Tokens

The count.py script (https://github.com/coreyleedavis/libguides-rag/blob/main/count.py) is designed to count the number of words in a specified text file and estimate the number of tokens that the text might generate. The script includes two main functions. The count_words_in_file function reads the content of the provided file, splits the text into words using whitespace as a delimiter, and returns the total word count. It also handles exceptions, such as if the file is not found or other errors occur during file reading. The estimate_tokens function calculates an estimated token count based on the word count by applying a scaling factor, which defaults to 1.2 tokens per word—a rough approximation often used for tokenization in natural language processing (NLP) models. When executed, the script uses a predefined file path (cleaned.txt), counts the words in the file, and prints both the word count and the estimated token count. This simple script provides *a simple way to tell if a specific .txt file is within the context window limits of models like Gemini,* and is particularly useful for determining if the .txt file can be processed using a tool like NotebookLM.

Create High-Dimensional Vector Embeddings

The embed.py script (https://github.com/coreyleedavis/libguides-rag/blob/main/embed.py) processes a cleaned text file, splits the content into manageable chunks, *generates vector embeddings for these chunks using a pre-trained transformer model, and stores the embeddings along with metadata in a persistent ChromaDB database*. The script uses the Hugging Face transformers library to load the intfloat/e5-large-v2 model and its tokenizer. It leverages PyTorch to perform computations on a selected device, in this case an MPS for Apple Silicon. The script uses ChromaDB to create a persistent vector store collection, enabling the storage of document embeddings alongside metadata such as chunk indices and source line numbers. The main processing involves reading the input text file line-by-line, tokenizing the content into 512-token chunks with a 50-token overlap, and generating vector representations. These embeddings, along with chunk metadata, are batched and stored in ChromaDB. The processed data, including embeddings and metadata, is saved persistently in the ./chroma_db directory for future use.

Retrieve Semantically Similar Text Chunks Based on a User Query

The query.py script (https://github.com/coreyleedavis/libguides-rag/blob/main/query.py) script provides *a terminal-based query interface for a ChromaDB vector database containing text embeddings, enabling users to search for and retrieve the most semantically relevant text chunks based on their input queries*. It uses the Hugging Face transformers library to load the pre-trained intfloat/e5-large-v2 model and its tokenizer, which are responsible for generating embeddings. ChromaDB's PersistentClient is used to connect to a pre-existing database stored at a specified path (./chroma_db) and retrieve a collection of stored text embeddings. When a user inputs a query, the script first converts the query into an embedding vector by tokenizing it and applying mean pooling[17] over the transformer model's output. This

[17] During this process, each word in a sentence gets turned into a list of numbers (vectors) that represent its meaning. Mean pooling combines all these

query embedding is then used to search the database for the most similar text chunks. The script displays the top results, including the text of each chunk, associated metadata (e.g., chunk ID and additional attributes), and the similarity distance.

Implement a RAG Pipeline to GPT-4o for Full Inference

The rag.py script (https://github.com/coreyleedavis/libguides-rag/blob/main/rag.py) *implements a RAG pipeline that combines a semantic search over a ChromaDB vector database with a GPT-4 model to generate contextually informed responses to user queries*. The script begins by setting up dependencies and configurations, including the intfloat/e5-large-v2 model for embedding queries, the ChromaDB database for storing and retrieving text chunks, and OpenAI's GPT API for generating responses. Key libraries used include torch for hardware-acceleration, chromadb for vector database management, transformers for model and tokenizer handling, and openai for interaction with the GPT API.

The script first initializes the hardware, automatically selecting the most suitable device (MPS for Apple Silicon, CUDA for GPUs, or CPU). It loads the e5-large-v2 model and tokenizer from Hugging Face, which are used to embed user queries into vector representations. The ChromaDB client connects to a persistent database (./chroma_db) containing previously stored embeddings and their associated metadata.

When executed, the script interacts with the user to retrieve *n* chunks, the query to be answered, and optional system instructions to guide the GPT model. The user query is embedded using the transformer model (e5), and the resulting vector is used to search the ChromaDB database for the most relevant text chunks. The search results include the text, metadata, and similarity distances for the top-ranked chunks.

These retrieved chunks are formatted into a structured context and appended to the system instructions. A message object is built, combining the context and the user query, and passed to the OpenAI GPT API. GPT-4o processes this input and generates a response informed by the retrieved context. The final output includes a detailed display of the retrieved chunks and the GPT-generated response.

numbers by taking their average, giving you one final set of numbers that represents the meaning of the entire sentence.

Comparing Bespoke RAG with WARC-GPT

Performance Improvement

Using wget to exclude non-textual elements from the site crawl achieved a remarkable 82% reduction in WARC file size, requiring only 2.2GB compared to Archive-It's 12.5GB. This improvement in storage size did not impact inference quality, but it's important to note that the resulting WARCS would likely not be appropriate for human consumption via a playback mechanism like wayback, because so much of the site structure and visual elements have been removed. However, the semantic density has been increased, improving things for machine consumption. In fact, using BeautifulSoup for a more focused HTML parsing (capturing only <p> and header tag content, rather than <body> tags), and additional text cleanup using regular expressions (see clean.py above), enhanced the quality of extracted data, which—as we'll see—led to considerable better inference quality downstream. This approach also reduced vector storage requirements. The bespoke RAG implementation required just 240MB of storage space, while the Archive-It WARCs in WARC-GPT demanded over 10GB, representing a fortyfold reduction in storage requirements for the embeddings.

Performance was also improved considerably through hardware optimization specific for the Apple Silicon M3 GPU architecture. This optimization, coupled with enhanced data preprocessing, yielded substantial improvements in the creation of the embeddings. The WARC-GPT implementation required over a week for complete processing of the full 12.5GB WARC corpus into vectors, while the optimized approach achieved comparable results in only a few hours. This order-of-magnitude improvement in processing time highlights the importance of data cleaning and hardware optimization strategies, especially in large-scale deployments.

Inference Quality

The performance difference between traditional WARC processing and WARC-GPT was striking, but context-dependent. When handling straightforward queries about well-established facts like "Who is Bob Belcher?" WARC-GPT performed quite well using frontier models like GPT-4o for inference. However, when it came to more granular or niche knowledge—details that might only appear in one or two specific episodes—it generally failed to provide accurate or complete answers.

Using stripped down WARC files and refined text documents proved transformative for retrieval within the bespoke RAG pipeline, significantly enhancing the system's capability to extract relevant source document chunks and generate precise, contextually relevant responses. These results underscore a fundamental principle in domain-specific LLM applications: *the quality of data preparation and processing serves as a critical determinant of inference accuracy and reliability*.

Here are some comparative results for niche questions:

<table>
<tr><td colspan="2">Question: Who is the Coach at Wagstaff?</td></tr>
<tr><td>WARC-GPT</td><td>Bespoke RAG</td></tr>
<tr><td>I don't know who the coach at Wagstaff is.</td><td>The coach at Wagstaff School is Coach Blevins. He is a P.E. teacher and also teaches fourth-grade science and sixth-grade math. Coach Blevins has red hair tied back in a ponytail and wears glasses, but he switches to shatterproof sports goggles in certain situations. He is voiced by Larry Murphy and makes his first appearance in the episode "Spaghetti Western and Meatballs" [Chunk 8].</td></tr>
<tr><td colspan="2">Comments: When examining source chunks (see below), it’s clear that the noise present in the WARC-GPT embeddings are causing serious issues when trying to retrieve semantically similar content. Fore example, here is a comparison of two 512 token chunks from WARC-GPT vs. bespoke RAG:

<table>
<tr><td>WARC-GPT Chunk 1</td><td>Bespoke RAG Chunk 1</td></tr>
<tr><td>valentine's day (holiday) valerie vampire disco death dance vampire disco death dance / gags vampire disco death dance / gallery vampire disco death dance / trivia vance vanessa vanessa (moo's clues) vanessa jackson vanessa jackson / gallery vanessa jackson / quotes vanessa jackson / trivia vicki vicki tufo vicki tufo / trivia video killed the gene - io star video killed the gene - io star / gags video killed the gene - io star / gallery video killed the gene - io star / trivia vincent bartos vincent bartos / gallery vincent bartos / quotes vincent bartos / trivia wag the hog wag the hog / gags wag the hog / gallery wag the hog / trivia wag the song wag the song / gags wag the song / gallery wag the song / trivia wagstaff school wagstaff school</td><td>content madison is a former wagstaff school student who now works part - time at caboose kabob she mentored tina when she was in eighth - grade. heading rose reynolds content rose reynolds information gender female hair blonde age 40s occupations student at waggstaff school formerly family children unnamed son behind the scenes first " the secret ceramics room of secrets " last " the secret ceramics room of secrets " voiced by pamela adlon heading voiced by content rose reynolds is a former student of wagstaff school she makes her first and only appearance in " the secret ceramics room of secrets. " she is voiced by pamela adlon. heading jordan sturman content jordan sturman is a character who moves</td></tr>
</table>
</td></tr>
</table>

news wagstaff whaler walter rubens walter russo wanda sykes wands, wands, wands warren fitzgerald warren fitzgerald / gallery warren fitzgerald / quotes warren fitzgerald / trivia waxing philosophical wayne wayne / gallery wayne / quotes wayne / trivia we'll never drift too far weekend at mort's weekend at mort's / gags weekend at mort's / gallery weekend at mort's / trivia wendi mclendon - covey wendy molyneux wes archer wetty, set, go wharf, me worry? wharf, me worry? / gags wharf, me worry? / gallery wharf, me worry? / trivia wharf arts center wharf horse wharf horse / gags wharf horse / gallery wharf horse / trivia wharfy what about blob? what about blob? / gags what about blob? / gallery what about blob? / trivia what about job? what about job? / gags what about job? / gallery what about job? / trivia what a (april) fool believes what a (april) fool believes / gags what a (april) fool believes / gallery what a (april) fool believes / trivia what if they're right what the tech wild steal - ions wild steal - ions / gags wild steal - ions / gallery wild steal - ions / trivia will (bully - ieve it or not) will (bully - ieve it or not) / gallery will (bully - ieve it or not) / quotes will (bully - ieve it or not) / trivia will forte willow wine train wonder wharf wonder	*away off screen his moving away meant that he would never play kickball with tina again and is why she is shown lying on the floor, depressed at the beginning of " sexy dance fighting ". heading dave titus content dave titus was a student at wagstaff school who was in seventh grade with phillip frond 35 years ago titus ran against frond for seventh grade class secretary frond won the election after hiding some of dave's votes meaning dave may well have won the election had frond not hid them. heading the gene courtney show content " the gene courtney show " season episode overall prod code 6 7 95 6asa01 cast starring also starring h jon benjamindan mintzeugene mirmanlarry murphyjohn robertskristen schaal david wainwill fortejohn michael higginstymberlee hillbobby tisdaledavid hermanmelissa bardin galskyjenny slate crew written by rich rinaldi directed by chris song broadcast details premiered february 14, 2016 viewers 2. 046 million episode chronology previous next " the cook, the steve, the gayle, her lover " " sexy dance healing " heading viewers content " the gene courtney show " is the seventh episode in season 6, being the ninety - fifth episode overall and the 2016 valentine's day special. heading plot content gene and courtney get their big break when they are asked to be the new hosts of the morning announcements, but their romantic history threatens to get in the way meanwhile, tina's attempt to play cupid goes tragically awry when she volunteers to spear - head the valentine's day carnation fundraiser. heading full story content it is breakfast at the belcher kitchen the monday before valentine's day on that friday tina is folding up a box as part of the " donation 4 carnations " project for the school*

The bespoke RAG chunks exhibit significantly higher semantic density, with a greater proportion of tokens together representing meaningful words. This contrasts with WARC-GPT, where the chunks exhibit semantic sparsity (most of the chunk above seems to represent navigation features, for example). The

difference arises from the data extraction methods: WARC-GPT relies on Beautiful Soup to capture content within <body> tags, which often includes substantial noise. In contrast, bespoke RAG focuses on extracting content from headers and <p> tags, leading to cleaner, more contextually rich data and improved semantic representation. (It's important to note here that its possible to make these changes within WARC-GPT itself, without having to rely on an external script, by adjusting the "Extract text from text/html" parameters in the ingest.py script (https://github.com/harvard-lil/warc-gpt/blob/main/warc_gpt/commands/ingest.py) Ensuring high-quality data input is essential for the success of a RAG pipeline. While WARC-GPT can be relatively easily customized to improve data extraction, standard Archive-It-generated WARCs typically require much more extensive cleaning and preprocessing to enhance their utility. This additional effort is crucial for transforming raw, noisy data into a format that supports effective retrieval and generation.

Question: What is Teddy's Favorite Kind of Hammer?	
WARC-GPT	**Bespoke RAG**
I don't know Teddy's favorite kind of hammer.	*Teddy's favorite kind of hammer is the McLecklin hammer, which he claims is "the Meryl Streep of hammers." He used it to destroy Bob's fukinawa knife in "The Millie-Churian Candidate" and made a video of it hammering nails online. This hammer is mentioned in several episodes, including "The Helen Hunt" and "Some Kind of Fender Benderful" (Chunk 1, Chunk 2).*
Comments: See above. This is the same issue related to chunk retrieval quality.	

Question: Who is Pepperoni?	
WARC-GPT	**Bespoke RAG**

Pepperoni is a character mentioned in a song from the animated television series "Bob's Burgers." The song describes Pepperoni as a handsome pony with strong morals and impressive muscles. The lyrics humorously highlight his attributes, such as being super buff and humble, and express a hope that he finds more pepperoni. This character seems to be part of a comedic or whimsical element within the show.	*Pepperoni is a fictional pony character from a story that Bob and Linda Belcher used to tell their children, Tina, Gene, and Louise, when they were younger. The pony, known as "Pepperoni the Sleepy Pony," was part of a bedtime story intended to help the kids fall asleep. As his name suggests, Pepperoni enjoys eating pepperoni, and at the end of the story "Fight at the Not Okay Chore-ral," he eats a stick of pepperoni. In the retelling of the story, the character is modified based on the children's input; Louise suggests that he shouldn't be so sleepy, and Tina wants him to be "really buff." In this version, Pepperoni is on a journey "looking for a place to not sleep" and ends up in a town called Westernville, where he interacts with various characters. The story concludes with Pepperoni eating a stick of pepperoni. The character is voiced by H. Jon Benjamin [Chunk 1, Chunk 4, Chunk 5].*
Comments: When examining the chunks, WARC-GPT returns 4 chunks, 3 of which are almost entirely navigation text (similar to the first few lines of text in the chunk below). In this case, there was enough content in the second chunk for the model to infer and answer. Here is the text from chunk 2 in from WARC-GPT: *ding " 7 " the (raccoon) king and i " 7. 1 " gas in my car " 8 " wharf, me worry? " 8. 1 " wharfin'with pop pop " 8. 2 " dr. clamstradamus " 9 " fraud of the dead : zombie - docu - pocalypse " 9. 1 " i'm a fruad " 9. 2 " zombie apocalypse " 10 " the nightmare 2 days before christmas " 10. 1 " santa loves glueten " 11 " mission impossi - bob " 11. 1 " beef - hemoth " 12 " jade in the shade " 12. 1 " flappy bappies " 13 " butt sweat and fears " 13. 1 " stir the pot " 13. 2 " funky farmer fred " 13. 3 " sweet love sugar " " fight at the not okay chore - ral " [ ] " pepperoni " [ ] singers : ah, pepperoni roamin'the plains yes, sir, he's one handsome pony one handsome pony looks like he found some pepperoni he's super buff and super humble the only thing hotter than his muscles are his morals he's got hard morals go on and look at him flexing his pectorals ah, come on, pepperoni no sliced ham or baloney don't want no ham or more baloney sure hope he finds more pepperoni " the amazing rudy " [ ] " why don ’ t you lead me to love " [ ] singers : why don't you lead me lead me to love right now in everything you do why don't you tell me tell me you care right now and show me*	

that you do this whole world's got me down not a friend can be found why don't you take me, take me! take me, take me by the hand! oh yeah and why don't you lead me lead me to love right now in everything you do? and why can't you see, girl i need a little love right now and show me that you do i wanna hear you say hey, baby " the pickleorette " [ ] " tea party " [ ] singers : tea party, party party, party, party little chairs sittin'in little chairs little saucers little saucers tea cups tea cups and tiny sandwiches tiny sandwiches spraying tea spraying tea... everywhere like he was the damn fountain damn fountain at the bellagio at the bellagio in that moment, everything With bespoke RAG, I was able to return 10 chunks, the first 5 of which contained dense information about the episode involving Pepperoni the Pony. The WARC-GPT embeddings generate numerous noisy chunks, making it difficult to retrieve semantically similar content. Since these noisy embeddings are scattered across the high-dimensional vector space rather than forming meaningful clusters, the system could not find meaningful semantic matches, resulting in lower inference quality compared to the bespoke RAG setup.

Question: Who are the Golden Dragons?	
WARC-GPT	**Bespoke RAG**
I don't know who the Golden Dragons are.	*The Gold Dragons are a soccer team that Louise gets signed up to play for in the episode "The Hurt Soccer." They have a reputation for playing very poorly, to the extent that a rule was implemented stating that if the score reaches 10-0 in a game, the team that hasn't scored loses the tie. However, in the episode, they manage to score a goal before this rule is applied, ultimately losing their game against the Blue Dragons with a score of 21-1 (Chunk 1, Chunk 2, Chunk 3, Chunk 4, Chunk 5, Chunk 6). Walter Rubens is associated with the Gold Dragons as he sets up the goals for the team and assigns Bob the role of team coach while Suzy and her dad, the team's coach, are away on vacation (Chunk 8). Mara is a player for the*

	Gold Dragons soccer team, and she makes her first and only appearance in "The Hurt Soccer" (Chunk 7, Chunk 9).
Comments: Same as questions one and two above.	

Question: What are Some of Gene's Fears?	
WARC-GPT	**Bespoke RAG**
I don't know what some of Gene's fears are.	*The provided context does not explicitly mention specific fears that Gene has. However, it does highlight some of his personality traits and behaviors, such as his impulsiveness, his love for fashion, and his imaginative nature, including his interactions with imaginary friends and inanimate objects. It also mentions his close relationship with his family, particularly his father, Bob, despite their differences. If you are looking for specific fears Gene might have, this information is not available in the provided context.*
Comments: Bespoke RAG fails here even though, according to CDX files associated with the original wget crawl, pages that explain his fear of snakes, and of being alone, were captured. A closer look at the chunks provided to GPT-4o reveal that none of this information was matched with the user query via the embeddings process. However, if I ask "Is Gene afraid of snakes?", the system is able to find the 'needle in the haystack': *Yes, Gene is afraid of snakes. This is evident in the episode "It Snakes a Village," where Gene's ophidiophobia (fear of snakes) prevents him from helping his sisters, Louise and Tina, find a snake in the woods. It takes him some time to gather the courage to save his siblings when they get trapped in quicksand. Despite his fear, Gene eventually overcomes it to rescue his sisters, showcasing his bravery when it truly matters (Chunk 3, Chunk 6).* (Just to note, WARC-GPT was not able to find the information about snakes with the query above.) Here's an example of an actual chunk retrieved during the query	

about snakes, with the relevant passage highlighted:

> Distance: 174.2310028076172
> Chunk ID: 172
> Metadata: {'chunk_index': 172, 'source_line_index': 0}
> Text: *protect himself at the end of the episode. gene takes a reverse norwegian stink hold for his sister " large brother, where fart thou? " despite their serious disputes, they are closer than most siblings and choose to spend most of their free time in and out of school together with their older sister, tina they even sit together at lunch, where the two play " food court " in " spaghetti western and meatballs. " they like to hold competitions with each other, like who can make the most disgusting drink " dr yap " or who can make the best burger " tappy tappy tappy tap tap tap " gene usually follows louise in her schemes, and with louise, he becomes more mischievous and manipulative that happens in " bad tina, " when louise and gene both hide in tina's room and see her bring her friends and boys over and when they discover tina got a temporary tattoo they blackmail their big sister both times to get her to do their chores. he's generally friendlier and more forgiving without her, like how he instantly forgave tina for detaining louise and him without prior forethought with louise " midday run " with gene, the sister - brother pair blackmail and manipulate tina whenever they can, but sometimes louise uses her brother in her schemes on rare occasions, her brother protests her plans, like in episodes " it snakes a village " and " stand by gene. " in " it snakes a village, " his ophidiophobia gets to him, and he wouldn't help his sisters find a snake, but he does save them in the end he resists most of her schemes because of his fears, like in " nightmare on ocean avenue street, " when he and tina wouldn't go into the haunted mansion to find the thief who stole their candy gene acts more like a big brother to louise after the events of " large brother, where fart thou? " when gene defends louise from logan at the end of the episode when logan was about to give louise a reverse norwegian stink hold, she started crying, so gene allowed himself to receive it instead of louise in the bob's burgers movie, he helps tina hold louise down so their little sister doesn't attack chloe barbash, who provoked louise by calling her a baby. heading love interests content louise has palpably insane feelings for a pint - sized pop star louise swoons for boo boothe moment where louise strokes her computer screen while swearing she will have her reunion with boo boo most of*

The system is able to find this information because I was specific about asking about Gene's fear of snakes, which allowed for the retrieval of related information in the adjacent semantic space. Asking about fears generally did not create a query embedding semantically similar enough to retrieve the specific chunk(s) needed to answer the question.

The Power of Inference

A similar thing happens when asking “Is Gene afraid of being alone?” Although it could not find explicit information about this fear, it does make excellent inferences based on available information:

> *The context provided does not explicitly state that Gene is afraid of being alone. However, there are instances where Gene interacts with imaginary or unseen characters, such as Ken, which might suggest a need for companionship or a way to cope with solitude. In the episode "Mr. Lonely Farts," Gene talks to Ken when he's bored, and Ken helps keep him company when his parents mistakenly leave him alone in their apartment (Chunk 5). This interaction could imply that Gene uses his imagination to deal with being alone, but it doesn't directly indicate a fear of solitude. If you are looking for a specific episode or scene where Gene expresses a fear of being alone, the context provided does not contain that information.*

We see here the *inferential power* of LLMs to help reason-though semantically similar information retrieved using high-dimensional vectors. This is not the kind of thing that can be done with simple keyword search, and demonstrates the power of RAG pipelines to transform the exploration of digital collections. This is also where we need to be very careful. Inferential abilities are subject to hallucinations. However, because we can pre-prompt systems to only use provided context, this is less of a problem. For example, when asking the question “Who would win in a fight, Bob or Jimmy?” the system very reasonably replies:

> *The context provided does not contain specific information about who would win in a fight between Bob and Jimmy from "Bob's Burgers." The show typically focuses on comedic and family-oriented storylines rather than physical confrontations. If you are looking for a detailed analysis or opinion, it might be best to consider the characters' personalities and traits as depicted in the series. Bob is generally portrayed as a hardworking and dedicated family man, while Jimmy Pesto is often depicted as his rival with a more competitive and antagonistic nature. However, without explicit context or scenarios from the show, it's difficult to determine the outcome of a hypothetical fight between them.*

Here, we can see the inference *somewhat contained by* the RAG architecture. Compare this with the answer provided directly from ChatGPT (this required the more context-aware query, “Consider the characters Bob and Jimmy from the TV show Bob's Burgers. Who would win in a fight, Bob or Jimmy?”):

If Bob Belcher and Jimmy Pesto were to get into a fight, Bob would likely emerge as the winner. Bob is physically larger and sturdier than Jimmy, and while he isn't particularly athletic, his long hours standing and working in his restaurant suggest he has a decent amount of strength and endurance. Jimmy, on the other hand, is leaner and more focused on his showmanship than on physical confrontation. Bob's determination and emotional drive, especially when it comes to protecting his family or defending his livelihood, would give him an edge in a fight. He has shown grit and resilience in challenging situations, whereas Jimmy's personality is more blustery and prone to backing down when things get serious. Throughout the series, Jimmy Pesto often antagonizes Bob with arrogance and petty insults, but his lack of substance and overconfidence suggest he would falter in a physical altercation. Overall, Bob's grounded and practical nature, combined with his physical sturdiness, would likely lead him to defeat Jimmy in a fight.

This does not take into consideration many of the situations in the show that would suggest otherwise, including an episode where Bob hides with Gene to avoid getting into a fight (S01E09). RAG wasn't able to match this information to the user query based on similarity search, while ChatGPT—while not exactly hallucinating —projected a certain amount of overconfidence while lacking specific enough training data to be more reasonable (Bob also loses a capoeira fight in S01E04). This can be more pernicious than simply saying, "I don't know."

The Importance of Prompt Engineering

WARC-GPT, in this case, did a pretty good job. I asked if it knew who Bob Belcher and Jimmy Pesto were, and it provided a good summary of each. When asked who would win in a fight, it's answer was concise and to the point:

In the animated series "Bob's Burgers," Bob Belcher and Jimmy Pesto are often portrayed as rivals, but their conflicts are usually comedic rather than physical. The show doesn't depict them engaging in an actual fight, so it's difficult to determine who would win. Their rivalry is more about outdoing each other in business and personal antics rather than physical confrontations.

This may be a result of its pre-prompt injection in the .env file, which states:

- Context comes from web pages that were captured as part of a web archives collection.
- When possible, use context to answer the question asked by the user.
- If you don't know the answer, just say that you don't know, don't try to make up an answer.

- Ignore context if it is empty or irrelevant.
- Cite and quote your sources whenever possible. Use their number (for example: [1]) and / or URL to reference them.

When I use the same pre-prompt injection in bespoke RAG, I get this answer:

> *The context does not provide specific information about who would win in a fight between Bob Belcher and Jimmy Pesto. However, it does describe their personalities and characteristics. Bob is portrayed as a good man, husband, and father who runs a struggling but quality-focused burger restaurant. In contrast, Jimmy Pesto is depicted as a deceitful and negligent father who owns a more successful but lower-quality pizzeria. These character traits might suggest that Bob has more integrity and resilience, while Jimmy might rely on cunning tactics. Ultimately, the outcome of a fight would depend on various factors not detailed in the context.*

WARC-GPT excels in this scenario through its refined prompt engineering approach, reducing speculative outputs and delivering more focused, reliable results.

Bespoke RAG vs. Long Context Window in NotebookLM

After comparing WARC-GPT with my bespoke RAG toolchain, I wanted to evaluate inference quality in RAG vs. long context windows. To conduct this test, I needed a WARC collection that would fit within the substantial context window of NotebookLM. Initially, I considered the Bob's Burgers WARC collection, but its size proved too large for this purpose. Here we see our first limitation: context windows do not work with very large collections of documents. To bypass this limitation while still being able to compare inference quality between bespoke RAG and long context windows, I created a smaller WARC collection that I could manipulate in NotebookLM from the University of Victoria Libraries' LibGuides.[18]

After "w-getting", scraping and cleaning the LibGuides content, the text aligned neatly within NotebookLM's extensive 2 million token context window. Once processed, NotebookLM's capacity to process and summarize an entire dataset provided a distinct advantage over WARC-GPT and my custom RAG tool, both of which are limited to retrieving and working with specific *chunks* of information

[18] LibGuides are curated online research guides developed by librarians to provide users with tailored information on various topics. They serve as a centralized resource for research methods, database searching, citation management, and subject-specific guidance across disciplines such as law, health sciences, education, and music. Additionally, LibGuides often explore broader themes like evaluating sources, combating misinformation, and understanding Indigenous perspectives in research.

rather than *generating cohesive overviews* of the material itself. RAG does not allow for this type of meta perspective as such. For example, NotebookLM produced the following summary of the LibGuides collection:

> *The provided text comprises numerous library research guides from the University of Victoria, covering diverse topics such as research methods, citation management, database searching, finding primary sources, and understanding Indigenous perspectives in research. Guides also address specific subject areas like music, law, health sciences, and education, offering practical advice and resources for students and researchers. Several guides also discuss evaluating sources, combating misinformation, and utilizing alternative metrics in research assessment.* (NotebookLM using Gemini 1.5 Pro)

This type of comprehensive summarization is something that most RAG tools cannot do, as *they are constrained to retrieving specific segments of information*. To further compare these tools, I asked each one to answer the question, “What is fake news?” using only the provided context. The results are presented below for direct comparison.

Question: What is Fake News?	
Bespoke RAG	**NoetbookLM**
Fake news is defined by the Ethical Journalism Network as information that is deliberately fabricated and published with the intention to deceive and mislead others into believing falsehoods or doubting verifiable facts. The concept of fake news has existed for as long as there has been news to report, but it has gained significant attention with the rise of social media and the 24/7 news cycle. Today, fake news can appear in various forms, ranging from the easily apparent to the decidedly inconspicuous, which necessitates greater attention to	*Fake news is defined as information that is deliberately fabricated and published with the intention to deceive and mislead others into believing falsehoods or doubting verifiable facts [1, 2]. This concept has existed for as long as there has been news to report, but has become more widespread with the rise of social media and 24/7 news cycles [1, 2].* *Some characteristics of fake news include [1, 2]:* ● *Appearing in many forms, from easily apparent to inconspicuous.* ● *Being deliberately created, or the result of an error, mistake, or misinterpretation.*

media and information literacy skills (Chunk 1, Chunk 2). *The spread of fake news can have both personal and academic consequences. In an ideal world, all reported information would be based solely on facts, allowing consumers to trust the reliability of the media. However, this is not the case, and it is important to learn how to identify false information (Chunk 1).* *Fake news can be deliberately created to generate engagement, often for financial gain, or it can result from errors, mistakes, or misinterpretations. Regardless of its origin, fake news can spread rapidly, especially with the help of advancing technology and social media, where false news often spreads faster than real news (Chunk 1, Chunk 3).* *The spread of fake news is further exacerbated by social media users who engage with posts containing misinformation, as well as by bots that can emulate human behavior and give the false impression of popularity and endorsement of inaccurate information (Chunk 3).*	● *Being created and posted with the intent of generating engagement, often to increase financial gain [2].* ● *Spreading through social media, often faster than real news [3].* ● *Being spread through circular reporting, where one source publishes misinformation that is picked up by other news outlets [3].* *It is important to be aware of the different types of fake news, and to develop media and information literacy skills [1, 2]:* ● *Misinformation is false or inaccurate information [2]. It is not always deliberately created and can be the result of error or misinterpretation [2].* ● *Disinformation is deliberately created with the intention to deceive or mislead [2].* ● *Other forms of misleading news can also spread misinformation [2].* *Various organizations and campaigns have developed strategies to help identify fake news, including [4]:* ● *MediaSmarts' 5 Ws: Consider who created the content, what the content is, where it comes from, when it was published, and why it was created [4].* ● *SIFT: Stop, Investigate the source, Find better coverage, and Trace claims back to the original context [4].* ● *News Media Canada's SPOT: Spot the problem, Prioritize your sources, Organize your analysis, and Take action [4].*

	IFLA's infographic: Consider the source, read beyond the headline, check the author, look at supporting sources, check the date, consider if it is a joke, and consult with an expert [4]. *Some tips for avoiding fake news include [4]:* *Fact-check news stories using dedicated websites [4].* *Be aware that photos and videos can be manipulated [4].* *Be aware of health mis- and disinformation as it can be particularly harmful [4].* *Be aware of "infodemics," which are too much information, including false or misleading information during a disease outbreak [5].* *Remember to evaluate news before you share it [5].* *Additionally, it is important to be aware of how social media platforms are attempting to address the spread of fake news [6]:* *Facebook attempts to curb the spread of misinformation by blocking fake accounts and using fact-checkers [6].* *Twitter suspends accounts that push misinformation and labels tweets that contain manipulated media [6].* *Instagram labels posts containing misinformation as false and makes them harder to find [6].*

	Finally, knowing what to avoid is only half the battle, so learn to recognize the characteristics of authoritative news [6]: ● *Look for news from sources with journalism ethics that adhere to a code of ethics [7].* ● *Look for balanced reporting with consideration of multiple sides of an issue [7].* ● *Be aware of the potential for political polarization, and consider consulting a variety of sources to gather information [7].* *By understanding what fake news is, how it spreads, and how to spot it, it is possible to become a more informed and responsible consumer of information [2-5].*

As we can see, the second answer is far more comprehensive, providing *actionable strategies* and a *broad context* for understanding and addressing fake news. In contrast, the first answer offers a high-level overview with less actionable depth, which would make it more accessible to a general audience, but might not be as well suited for academic contexts where users might expect more detailed guidance.

I asked Claude 3.5 Sonnet to evaluate each answer and tell me which it thought was the best. In fact, using LLMs to evaluate other LLMs is becoming more common:

> As AI gets more advanced, we're beginning to use one AI to evaluate another. This method is fast and can handle massive amounts of data without tiring. Plus, AI can spot complex patterns that humans might overlook, offering a detailed look at performance...Many researchers find that mixing AI with human evaluation works best. AI handles the bulk of data processing, while humans add essential context and insight. (SuperAnnotate, 2024)

When I asked Claude to evaluate the two answers, something very interesting happened. It proposed its own novel framework, which it called ACCURATE. It didn't initially tell me that this wasn't an established heuristic until I asked it to give me more details, at which point it almost timidly apologized: "*I need to acknowledge that I generated the 'ACCURATE' framework in my previous responses as a way to evaluate the answers, rather than referencing an established framework. This was not clear in my responses and could be misleading...This is a good reminder for me* *to be explicit when I'm proposing something new versus referencing established frameworks*" Wow. It's always worth digging a bit deeper when it comes to LLMs (and especially, it seems to me, Claude). At any rate, here is the novel evaluation framework it created, complete with a final score for each answer:

I'll analyze each answer systematically using the ACCURATE framework, providing specific evidence for each metric:

ACTIONABLE

Answer 1: 5/10

- *Primarily describes what fake news is and how it spreads*
- *Lacks specific actions readers can take*
- *No concrete tools or strategies provided*

Answer 2: 9/10

- *Provides multiple specific frameworks (SIFT, SPOT, MediaSmarts' 5 Ws)*
- *Includes concrete tips for fact-checking*
- *Details platform-specific approaches on social media*
- *Lists specific steps readers can take to verify information*

COMPREHENSIVE

Answer 1: 6/10

- *Covers basic definition and spread mechanisms*
- *Touches on personal and academic consequences*
- *Misses important subtopics like types of fake news, verification strategies, and platform responses*

Answer 2: 9/10

- *Covers full spectrum from definition to practical application*
- *Distinguishes between misinformation and disinformation*
- *Includes platform-specific information*
- *Addresses both identification and prevention*
- *Discusses role of social media platforms*
- *Includes health-specific misinformation concerns*

CITED

Answer 1: 4/10

- *Uses informal "chunk" references*
- *Citations are vague and don't indicate specific sources*
- *Limited citation frequency*
- *No clear way to verify sources*

Answer 2: 8/10

- *Uses formal numerical citation system*
- *Citations appear throughout text*
- *Citations tied to specific claims*
- *References appear organized by topic*
- *However, full citation details not provided in excerpt*

UNBIASED

Answer 1: 8/10

- *Presents information neutrally*
- *Avoids political stance*
- *Acknowledges multiple causes of fake news*
- *Could better address multiple perspectives*

Answer 2: 9/10

- *Explicitly mentions need for balanced reporting*
- *Acknowledges political polarization*
- *Recommends consulting multiple sources*
- *Presents various frameworks without preferencing any*

READABLE

Answer 1: 8/10

- *Flows naturally as narrative*
- *Clear paragraph structure*
- *Logical progression of ideas*
- *Concise and accessible language*

Answer 2: 6/10

- *Information dense and potentially overwhelming*
- *Heavy use of bullet points disrupts narrative flow*

- *Well-organized but less conversational*
- *Requires more cognitive effort to process*

ACCURATE

Answer 1: 8/10

- *Information aligns with current understanding*
- *Claims are reasonable and supported*
- *No apparent factual errors*
- *Could benefit from more specific examples*

Answer 2: 9/10

- *Detailed and precise information*
- *Multiple frameworks accurately described*
- *Platform-specific details correct*
- *Includes current terminology (e.g., "infodemics")*
- *Supports claims with citations*

TIMELY

Answer 1: 7/10

- *Mentions current technology and social media*
- *References 24/7 news cycle*
- *Could include more current examples/trends*

Answer 2: 8/10

- *Includes current platform policies*
- *References recent concepts like "infodemics"*
- *Addresses current social media landscape*
- *Includes contemporary verification tools*

ENGAGING

Answer 1: 8/10

- *Clear narrative structure*
- *Accessible language*
- *Good flow between ideas*
- *Maintains reader interest through logical progression*

Answer 2: 7/10

- *Rich in useful information*
- *Well-organized for reference*
- *Bullet point structure may reduce engagement*
- *More technical/academic tone might deter some readers*

Overall Scores:

- *Answer 1: 54/80 (67.5%)*
- *Answer 2: 65/80 (81.25%)*

So, according to Claude, while answer two scores higher overall, each has its own obvious strengths.

You can see right away that regardless, each approach provides high-quality natural language answers from source documents. NotebookLM is certainly easier to use (you don't have to build anything), but it does in this case require cleaning web data across 1000+ pages prior to consumption in a long context window.

Since Libguides information is publicly available, there are no privacy or sensitive materials considerations. And specific "notes" can be generated and shared widely. On the other hand, a RAG pipeline could be (relatively) easy to spin up with a simple user interface using Flask. Students and other researchers would be presented with a clean chat interface to interrogate LibGuides materials. In this use case, users would not likely benefit from an overview of Libguides, they would be looking for specific information about their research needs. Furthermore, this pipeline could be managed locally on a stack made entirely of open models and other tools, which can be desirable for a number of reasons. Running a RAG stack locally with open source components used to fetch and clean data, embed and store vectors, and query models like Llama 3, offers significant control and privacy advantages. When sensitive or personal information is involved, keeping everything on-prem (or in an approved offsite cloud facility) ensures that no data ever leaves the infrastructure under your control. This can be crucial for compliance with privacy and security requirements, and if you're not able to host locally, large cloud providers are often able to meet (and exceed) these demands. Local deployment also reduces *vendor lock-in* risks, giving organizations the freedom to modify, fine-tune, and customize all aspects of their pipeline without being unduly constrained by proprietary API limitations and, importantly, pricing structures.

However, the on-prem open source approach comes with tradeoffs. Setting up and maintaining a RAG pipeline requires substantial technical expertise across multiple domains. You need to handle your own model deployment, updates, scaling, and monitoring, which can be resource-intensive. And open source models, while improving rapidly, can still lag behind the latest proprietary models like GPT, Claude Sonnet, Gemini, etc., just in terms of raw performance and capabilities (look at the chatbot arena leaderboard at https://huggingface.co/spaces/lmarena-ai/chatbot-arena-leaderboard, and you'll see a lot of "propriety" in the "license" column).

The security equation isn't straightforward either. While local deployment provides complete data isolation, it also means that you or your IT department are responsible for implementing all security measures. Cloud providers like AWS and Microsoft invest heavily in security and compliance, which can actually provide better protection than some in-house solutions, particularly for smaller organizations with limited resources.

At the end of the day, the choice depends on specific organizational needs, the technical expertise available, financial and time constraints, and specific use-cases. A hybrid approach is often the most practical—using local infrastructure for sensitive or high-volume workloads, perhaps starting with a sandbox, while looking to leverage cloud services for less sensitive or sporadic needs, or just to experiment with new tools. This will allow organizations like research libraries to balance control, cost, and convenience while maintaining flexibility as their needs evolve.

Thoughts on WARC-GPT after Testing

Exploring RAG Applications with WARC-GPT

WARC-GPT is a great tool for investigating the potential of RAG applications. Thanks to the innovative work of the Harvard Law Library Innovation Lab, this tool provides a robust platform for exploring how LLMs can interact with web data. One of its best features is the ability to configure WARC-GPT to operate entirely with an open-source toolchain, allowing users to run applications locally without reliance on proprietary systems.

Prompt optimization

How users phrase their queries makes a big difference in how well RAG systems work. While researchers often start with basic searches like "Robert Graves letters" or "first nations fishing practices," these brief queries typically don't get the best results. Better queries are more specific and complete, for example, "What personal letters from Robert Graves describe the Battle of the Somme from a first-hand perspective?" or "What materials in the collection explain traditional fishing practices among Pacific Northwest First Nations communities?" These detailed questions give the

RAG system the context it needs to find and present relevant materials more effectively.

Research libraries are well positioned to work with other campus units to help people develop the digital literacy skills needed to effectively use these new technologies. By teaming up with writing centers, teaching and learning centers, and academic departments, libraries can create workshops and training programs that combine prompt engineering skills with traditional research methods.

Having said this, the idea that research libraries can meaningfully shape how researchers use AI tools like LLMs may be overly optimistic. Just as we had to reimagine our role when Google transformed how people search for information, we should expect to feel similarly sidelined as the LLM era unfolds, if what we're focused on is the *use* of these tools. Realistically, students and researchers will likely develop their own intuitive approaches to using LLMs through direct experimentation and peer learning, much as they did with search engines. The notion of formal "prompt engineering workshops" may be as outdated as past library initiatives around "proper" Google searching that failed to gain traction.

So, rather than trying to position ourselves as AI literacy experts, in an environment of increasingly constrained resources, perhaps we should focus more on resourcing areas like unique digital collection development and preservation. *Users will naturally adapt to and master new AI tools on their own terms, just as they did with previous technological shifts.* The energy spent on creating formal training programs might be better directed toward digital collections and all of the infrastructure and staffing needed to support them, including the creation of RAG pipelines into these rich, unique sources.

Challenges and Considerations for RAG Optimization

Out-of-the-box RAG applications, while functional, require significant optimization to perform effectively. A critical factor in this process is the quality of the data being used. Clean and well-structured data is essential for generating effective chunk embeddings, which serve as the foundation for meaningful interactions with LLMs in a RAG architecture. The methodology used for extracting and chunking data can have a substantial impact on the performance of the system, influencing both the accuracy of retrievals and the computational efficiency of creating embeddings from digital collections.

Computational Demands and Hardware Requirements

Generating embeddings is an extremely computationally intensive process that demands careful consideration of hardware and memory configurations. Consumer-grade hardware has limited capacity for managing the computations required for RAG at scale. Moreover, most research libraries and university IT departments lack sufficient GPU resources to support these tasks effectively. As a result, institutions and researchers working with RAG will likely need to rely on commercial cloud services to achieve the necessary computational scale.

Preparing Data for WARC-GPT

When working with WARC-GPT, preparing clean and structured WARC files is critical for optimal performance. Archive-It WARC files, in particular, tend to be noisy and require lots of preprocessing. Tools such as wget can help create cleaner web data for ingesting into the system. Additionally, users should experiment with text cleaning, embedding strategies, and LLM inference models on small datasets before scaling operations.

Enhancing RAG with Summarization and Long Context Windows

Currently, the “flavor” of RAG implemented by WARC-GPT is not well-suited for answering questions about an overall collection of documents. A potential improvement for future iterations of the tool would be to incorporate summarization at the ingest level. Summarizing the contents of WARC records before adding them to the knowledge base could enable WARC-GPT to provide answers at a collection-wide level. This approach, combined with long context windows like those used in tools such as NotebookLM, shows promise for improving the system’s ability to work with smaller datasets effectively. However, privacy concerns remain a critical issue when implementing long context windows, especially when dealing with sensitive or restricted data in environments like NotebookLM.

Multimodal Chunk Embedding and Future Directions

As the field of RAG continues to evolve, incorporating multimodal chunk embedding algorithms will become increasingly important. These algorithms will allow us to integrate diverse data types, such as audio, video, and image files, into RAG knowledge bases.

Conclusion

Large Language Models present significant opportunities for research libraries, but their inherent limitations cannot be overlooked. The black box nature of LLMs makes it difficult to trace their reasoning, raising concerns about trust, validation, and alignment with scholarly standards of accuracy and authorship. While these models can process vast amounts of information and generate useful responses, their lack of transparency and potential for misinformation pose significant risks. For libraries and archives, ensuring the reliability and accuracy of AI-assisted retrieval is essential.

Retrieval-Augmented Generation offers a practical approach to addressing some of these concerns. By grounding responses in curated digital collections, RAG enhances the relevance and accuracy of LLM outputs while mitigating against hallucinations. Testing WARC-GPT demonstrated both the potential and challenges of applying RAG to web archives. While the tool facilitated natural language querying over complex data sources, issues with retrieval accuracy, chunking strategies, and processing inefficiencies highlighted the need for refinement.

In response, a bespoke RAG pipeline was developed to improve on WARC-GPT's shortcomings. This system introduced text cleaning, enhanced chunking techniques, optimized embeddings, and hardware acceleration, resulting in more precise and efficient responses.

Despite these advancements, LLM integration into access and preservation workflows should be approached with caution. The benefits must be weighed against computational costs, ethical concerns, and reliance on proprietary AI systems. Research libraries must navigate these trade-offs carefully.

The use of LLMs in digital preservation also raises broader questions about the role of AI in shaping access to knowledge. When retrieval is mediated by AI, how do we maintain provenance, authenticity, and context? While RAG provides a step toward more responsible AI integration, it is not a perfect solution. Human expertise remains critical in curating and evaluating AI-assisted outputs, ensuring that digital collections serve as reliable sources rather than algorithmically reshaped interpretations.

Ultimately, the future of digital collection management and preservation will not be dictated by AI alone but by its careful and deliberate integration into existing workflows. Research libraries must remain active participants in this evolution, advocating for transparency, accountability, and meaningful human oversight. The goal is not simply to preserve information but to ensure that it remains discoverable, verifiable, and useful for generations to come.

Looking ahead, several key considerations emerge for organizations contemplating RAG implementations:

1. Infrastructure Requirements: The substantial computational demands of RAG systems necessitate careful planning around hardware resources and potentially cloud service integration.
2. Data Quality: The critical importance of clean, well-structured input data for effective embedding and retrieval operations cannot be overstated.
3. Privacy and Security: Organizations must balance the benefits of advanced language models with data protection requirements, potentially leading to hybrid approaches combining local and cloud-based solutions.
4. Technical Expertise: Successful implementation requires significant expertise across multiple domains, from data processing to model optimization.
5. Scalability: Systems must be designed with growth in mind, considering both computational efficiency and maintenance requirements.

LLMs and RAG represent not just a technological milestone but a major shift in how we might approach our digital collections and their long-term preservation. Their transformative potential lies in their ability to augment human capabilities and enhance engagement with digital content. As we continue to refine these technologies and explore their applications, LLM-powered retrieval systems will likely play a central role in shaping the future of digital interaction and preservation.

The work required to understand and implement these technologies reminds us that we are still in the early stages of a significant technological transformation. While the challenges are substantial, the potential benefits—particularly in making vast collections of digital materials more accessible and useful—make this all worth our careful consideration.

Bibliography

Abdin, M., Aneja, J., Behl, H., Bubeck, S., Eldan, R., Gunasekar, S., ... & Zhang, Y. (2024). Phi-4 technical report. arXiv preprint arXiv:2412.08905. https://www.microsoft.com/en-us/research/uploads/prod/2024/12/P4TechReport.pdf

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., ... & McGrew, B. (2023). Gpt-4 technical report. arXiv preprint arXiv:2303.08774. https://arxiv.org/abs/2303.08774

Almeida, D.R. (2024, April 10). Synthetic data generation (Part 1). OpenAI Cookbook. https://cookbook.openai.com/examples/sdg1

Al-Sibai, N. (2024, June 10). Researchers say there's a vulgar but more accurate term for AI hallucinations. Futurism. https://futurism.com/the-byte/researchers-ai-chatgpt-hallucinations-terminology

Amazon Web Services. (n.d.). Amazon Bedrock security and privacy. Amazon Web Services. https://aws.amazon.com/bedrock/security-compliance/

Amazon Web Services. (n.d.). Retrieval Augmented Generation. Amazon Web Services. https://docs.aws.amazon.com/sagemaker/latest/dg/jumpstart-foundation-models-customize-rag.html

Amazon Web Services. (2023, November 28). Customize models in Amazon Bedrock with your own data using fine-tuning and continued pre-training. Amazon Web Services. https://aws.amazon.com/blogs/aws/customize-models-in-amazon-bedrock-with-your-own-data-using-fine-tuning-and-continued-pre-training/

Anthropic. (2023, October 5). Decomposing language models into understandable components. https://www.anthropic.com/research/decomposing-language-models-into-understandable-components

Associated Press. (2024, November 29). Canadian news publishers sue OpenAI over alleged copyright infringement. Associated Press. https://apnews.com/article/canada-news-publishers-lawsuit-chatgpt-3e1790fcf4c9f001f1d32609c4d547af

Association of Research Libraries. (2024). Research libraries guiding principles for artificial intelligence. Association of Research Libraries. https://www.arl.org/wp-content/uploads/2024/04/Research-Libraries-Guiding-Principles-for-Artificial-Intelligence.pdf

Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021, March). On the dangers of stochastic parrots: Can language models be too big?🦜. In Proceedings of the 2021 ACM conference on fairness, accountability, and transparency (pp. 610-623). https://dl.acm.org/doi/abs/10.1145/3442188.3445922

Blouin, L. (2023, March 6). AI's mysterious 'black box' problem, explained. University of Michigan-Dearborn. https://umdearborn.edu/news/ais-mysterious-black-box-problem-explained

Blumenthal, K. (2024). Track access to your web archives with Plausible Analytics. Archive-It Help Center. https://support.archive-it.org/hc/en-us/articles/18832246724372-Track-access-to-your-web-archives-with-Plausible-Analytics

Brügger, N. (2018). The archived web: doing history in the digital age. Mit Press.

Brügger, N., & Milligan, I. (Eds.). (2019). The SAGE handbook of web history. SAGE Publications.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., Nori, H., Palangi, H., Ribeiro, M. T., & Zhang, Y. (2023, March 22). Sparks of Artificial General Intelligence: Early experiments with GPT-4. arXiv.org. https://arxiv.org/abs/2303.12712

Cargnelutti, M, Mukk, K, & Stanton, C. (2024, February 12). WARC-GPT: An Open-Source tool for Exploring Web Archives using AI | Library Innovation Lab. https://lil.law.harvard.edu/blog/2024/02/12/warc-gpt-an-open-source-tool-for-exploring-web-archives-with-ai/

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., ... & Fiedel, N. (2023). Palm: Scaling language modeling with pathways. Journal of Machine Learning Research, 24(240), 1-113. https://arxiv.org/pdf/2204.02311

Clark, J. (2024, December 23). Import AI 395: AI and energy demand; distributed training via DeMo; and Phi-4. Import AI. https://importai.substack.com/p/import-ai-395-ai-and-energy-demand

Clark, J. (2024, December 9). Import AI 394: Global MMLU; AI safety needs AI liability; Canada backs Cohere. Import AI. https://importai.substack.com/p/import-ai-394-global-mmlu-ai-safety

Clarke, A. C. (1962). Profiles of the Future: An Inquiry into the Limits of the Possible. Harper & Row.

Chiang, T. (2023, February 9). ChatGPT is a blurry JPEG of the web. The New Yorker. https://www.newyorker.com/tech/annals-of-technology/chatgpt-is-a-blurry-jpeg-of-the-web

Chen, R. J., Lu, M. Y., Chen, T. Y., Williamson, D. F. K., & Mahmood, F. (2021). Synthetic data in machine learning for medicine and healthcare. Nature Biomedical Engineering, 5(6), 493—497. https://doi.org/10.1038/s41551-021-00751-8

Codingscape. (2024, September). Most powerful LLMs (large language models). Codingscape. https://codingscape.com/blog/most-powerful-llms-large-language-models

Coldewey, D. (2020, March 20). Stuart Russell on how to make AI "human-compatible". TechCrunch. https://techcrunch.com/2020/03/20/stuart-russell-on-how-to-make-ai-human-compatible/

Conroy, G. & Mallapaty, S. (2025, January 30). How China created AI model DeepSeek and shocked the world. Nature. https://www.nature.com/articles/d41586-025-00259-0

Department of Finance Canada. (2024, December 6). Deputy Prime Minister announces $240 million for Cohere to scale-up AI compute capacity. Canada.ca. https://www.canada.ca/en/department-finance/news/2024/12/deputy-prime-minister-announces-240-million-for-cohere-to-scale-up-ai-compute-capacity.html

Dooley, J., & Bowers, K. (2018). Descriptive metadata for web archiving: recommendations of the OCLC research library partnership web archiving metadata working group. https://www.oclc.org/content/dam/research/publications/2018/oclcresearch-wam-recommendations.pdf

Ecekamar. (2025, January 8). Introducing PHI-4: Microsoft's newest small language model specializing in complex reasoning. TECHCOMMUNITY.MICROSOFT.COM. https://techcommunity.microsoft.com/blog/aiplatformblog/introducing-phi-4-microsoft%E2%80%99s-newest-small-language-model-specializing-in-comple/4357090

EC Innovations. (2025, January 8). Synthetic data generation: What is its role in AI training. EC Innovations. https://www.ecinnovations.com/blog/synthetic-data-generation-what-is-its-role-in-ai-training/

The Economist. (2025, January 23). China's AI industry has almost caught up with America's. The Economist. https://www-economist-com.ezproxy.library.uvic.ca/briefing/2025/01/23/chinas-ai-industry-has-almost-caught-up-with-americas

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Chen, A. S., & Jones, A. (2021). A mathematical framework for transformer circuits. OpenAI. Retrieved from https://transformer-circuits.pub/2021/framework/index.html

Erkaya, E. (2022). A comprehensive analysis of subword tokenizers for morphologically rich languages (Doctoral dissertation, Bogaziçi University). https://www.cmpe.boun.edu.tr/~gungort/theses/A%20Comprehensive%20Analysis%20of%20Subword%20Tokenizers%20for%20Morphologically%20Rich%20Languages.pdf

Farmer, D. (2024, August 8). 10 top vector database options for similarity searches. Search Data Management. https://www.techtarget.com/searchdatamanagement/tip/Top-vector-database-options-for-similarity-searches

Farrell, H., & Fourcade, M. (2024, September 4). Large language models will upend human rituals. The Economist. https://www.economist.com/by-invitation/2024/09/04/large-language-models-will-upend-human-rituals

Ford, C. (2024, July 26). The most promising way to understand how AI works is by understanding the brain. Vox. https://www.vox.com/future-perfect/362759/ai-interpretability-openai-claude-gemini-neuroscience

Gardner, T. (2024, October 14). Google to buy power for AI needs from small modular nuclear reactor company Kairos. Reuters. https://www.reuters.com/technology/artificial-intelligence/google-buy-power-small-modular-nuclear-reactor-company-kairos-ai-needs-2024-10-14/

Grinkevičius, P. (2025, February 3). Cost of DeepSeek's AI models. Cybernews. https://cybernews.com/news/cost-of-deepseeks-ai-models/

Gibaut, W., Pereira, L., Grassiotto, F., Osorio, A., Gadioli, E., Munoz, A., Gomes, S., & dos Santos, C. F. (2023). Neurosymbolic AI and its Taxonomy: a survey. arXiv preprint arXiv:2305.08876. Retrieved from https://arxiv.org/abs/2305.08876

Google. (n.d.). Learn how NotebookLM protects your data. Google. https://support.google.com/notebooklm/answer/15724963

Google DeepMind. (2024, July 31). Gemma Scope: Helping the safety community shed light on the inner workings of language models. Google DeepMind. https://deepmind.google/discover/blog/gemma-scope-helping-the-safety-community-shed-light-on-the-inner-workings-of-language-models/

Government of British Columbia. (2024, September 18). BC Hydro receives strong response to call for clean electricity to power economy. Government of British Columbia. https://news.gov.bc.ca/releases/2024EMLI0068-001550

Hao, S., Han, W., Jiang, T., Li, Y., Wu, H., Zhong, C., Zhou, Z., & Tang, H. (2024, January 3). Synthetic data in AI: challenges, applications, and ethical implications. arXiv.org. https://arxiv.org/abs/2401.01629

Heaven, W. D. (2024a, October 9). Geoffrey Hinton tells us why he's now scared of the tech he helped build. MIT Technology Review. https://www.technologyreview.com/2023/05/02/1072528/geoffrey-hinton-google-why-scared-ai/

Heaven, W. D. (2024b, April 23). Large language models can do jaw-dropping things. But nobody knows exactly why. MIT Technology Review. https://www.technologyreview.com/2024/03/04/1089403/large-language-models-amazing-but-nobody-knows-why/

Holland, S. (2025, January 22). Trump announces private-sector $500 billion investment in AI infrastructure. Reuters. https://www.reuters.com/technology/artificial-intelligence/trump-announce-private-sector-ai-infrastructure-investment-cbs-reports-2025-01-21/

Hu, K., & Tong, A. (2024, November 15). OpenAI and others seek new path to smarter AI as current methods hit limitations. Reuters. https://www.reuters.com/technology/artificial-intelligence/openai-rivals-seek-new-path-smarter-ai-current-methods-hit-limitations-2024-11-11/

Huang, R. (2025, January 26). Silicon Valley is raving about a Made-in-China AI model. WSJ. https://www.wsj.com/tech/ai/china-ai-deepseek-chatbot-6ac4ad33

Hugging Face. (2024, November 18). The Big Benchmarks Collection. Hugging Face. https://huggingface.co/collections/open-llm-leaderboard/the-big-benchmarks-collection-64faca6335a7fc7d4ffe974a

Hutson, M. (2024). How does ChatGPT'think'? Psychology and neuroscience crack open AI large language models. Nature, 629(8014), 986-988. https://www.nature.com/articles/d41586-024-01314-y

IBM. (n.d.). Neural network. https://www.ibm.com/think/topics/neural-networks

IBM Research. (2024, August 15). Why larger LLM context windows are all the rage. IBM Research Blog. https://research.ibm.com/blog/larger-context-window

IFLA. (n.d.). Developing a library strategic response to Artificial Intelligence. https://www.ifla.org/g/ai/developing-a-library-strategic-response-to-artificial-intelligence/

intfloat/e5-large-v2. (n.d.) E5-large-v2. Hugging Face. https://huggingface.co/intfloat/e5-large-v2

Jones, N. (2024). The AI revolution is running out of data. What can researchers do? Nature, 636(8042), 290—292. https://doi.org/10.1038/d41586-024-03990-2

Jones, N. (2025). AI hallucinations can't be stopped — but these techniques can limit their damage. Nature, 637(8047), 778—780. https://doi.org/10.1038/d41586-025-00068-5

Joshi, M., Choi, E., Weld, D., & Zettlemoyer, L. (2017). TriviaQA: a large scale distantly supervised challenge dataset for reading Comprehension. Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). https://doi.org/10.18653/v1/p17-1147

Kilpatrick, L., Mallick, S. B., & Kofman, R. (2024, June 27). Gemini 1.5 Pro 2M context window, code execution capabilities, and Gemma 2 are available today. https://developers.googleblog.com/en/new-features-for-the-gemini-api-and-google-ai-studio/

Klein, E. (2025, January 18). Now is the time of monsters. The New York Times. https://www.nytimes.com/2025/01/12/opinion/ai-climate-change-low-birth-rates.html

Knight, W. (2025, January 7). Nvidia's 'Cosmos' AI helps humanoid robots navigate the world. WIRED.
https://www.wired.com/story/nvidia-cosmos-ai-helps-robots-self-driving-cars/

Kruppa, M., & Seetharaman, D. (2025, January 30). Why 'Distillation' has become the scariest word for AI companies. WSJ.
https://www.wsj.com/tech/ai/why-distillation-has-become-the-scariest-wordfor-ai-companies-aa146ae3

Kumar, S. (2024, March 19). Tokenization in large language models. Sushant Kumar.
https://sushant-kumar.com/blog/tokenization-in-large-language-models

Lal, V., Ma, A., Aflalo, E., Howard, P., Simoes, A., Korat, D., ... & Wasserblat, M. (2021, April). InterpreT: An interactive visualization tool for interpreting transformers. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations (pp. 135-142).
https://aclanthology.org/2021.eacl-demos.17/

Lee, T. B. (2024, December 19). Why AI language models choke on too much text. Ars Technica.
https://arstechnica.com/ai/2024/12/why-ai-language-models-choke-on-too-much-text/

Lewis, D., & Nguyen, N. (2024, December 23). OpenAI's GPT-5 is behind schedule. What happened? - Tech news briefing - WSJ podcasts. WSJ.
https://www.wsj.com/podcasts/tech-news-briefing/openais-gpt-5-is-behind-schedule-what-happened/74294715-941c-4f42-8cbe-1bd2f161b970

Li, Z., Li, C., Zhang, M., Mei, Q., & Bendersky, M. (2024, July 23). Retrieval Augmented Generation or Long-Context LLMs? a comprehensive study and hybrid approach. arXiv.org. https://arxiv.org/abs/2407.16833

Marcus, G. (2024, October 11). LLMs don't do formal reasoning - and that is a HUGE problem. Marcus on AI.
https://garymarcus.substack.com/p/llms-dont-do-formal-reasoning-and

Marche, S. (2024, January 31). 5 best books about Artificial Intelligence. The New York Times.
https://www.nytimes.com/2024/01/31/books/review/artificial-intelligence-best-books.html

Merriam-Webster. (n.d.). How many words are there in English? https://www.merriam-webster.com/help/faq-how-many-english-words

Meta. (2024, December 6). Llama-3.2-3B. Hugging Face. https://huggingface.co/meta-llama/Llama-3.2-3B

Metz, C., & Mickle, T. (2024, September 25). Behind OpenAI's audacious plan to make A.I. flow like electricity. The New York Times. https://www.nytimes.com/2024/09/25/business/openai-plan-electricity.html

Metz, C., Kang, C., Frenkel, S., Thompson, S. A., & Grant, N. (2024, April 9). How tech giants cut corners to harvest data for A.I. The New York Times. https://www.nytimes.com/2024/04/06/technology/tech-giants-harvest-data-artificial-intelligence.html

Milligan, I. (2019). History in the age of abundance? How the web is transforming historical research. McGill-Queen's University Press.

Mitchell, M. (2019). Artificial intelligence: A guide for thinking humans. Farrar, Straus and Giroux.

Mowshowitz, Z. (2025, January 22). On DeepSeek's r1. Don't Worry About the Vase. https://thezvi.substack.com/p/on-deepseeks-r1

Mulligan, S. J. (2024, December 5). Google DeepMind has a new way to look inside an AI's "mind." MIT Technology Review. https://www.technologyreview.com/2024/11/14/1106871/google-deepmind-has-a-new-way-to-look-inside-an-ais-mind/

Nanda, N. (2024). A Comprehensive Mechanistic Interpretability Explainer & Glossary. Retrieved from https://www.neelnanda.io/mechanistic-interpretability/glossary

Newport, C. (2024, March 15). Can an A.I. make plans? The New Yorker. https://www.newyorker.com/science/annals-of-artificial-intelligence/can-an-ai-make-plans

Nielsen. (2021, January 12). Tops of 2020: Nielsen streaming unwrapped. Nielsen. https://www.nielsen.com/insights/2021/tops-of-2020-nielsen-streaming-unwrapped/

Nielsen, M. A. (2019, December). Neural networks and deep learning. http://neuralnetworksanddeeplearning.com/

NVIDIA Blog. (2025, February 1). What is Retrieval-Augmented Generation aka RAG. NVIDIA Blog. https://blogs.nvidia.com/blog/what-is-retrieval-augmented-generation/

O'Connor, R. (2024, August 27). Emergent abilities of large language models. News, Tutorials, AI Research. https://www.assemblyai.com/blog/emergent-abilities-of-large-language-models/

OCUL Task Force on Machine Learning/AI. (2024, May). OCUL artificial intelligence/machine learning report and strategy. https://ocul.on.ca/sites/default/public/2024-09/20240625_OCUL-TFMLAI_FinalReport_Strategy_WEB.pdf

Olah, C. (n.d.). Mechanistic interpretability, variables, and the importance of interpretable bases. https://www.transformer-circuits.pub/2022/mech-interp-essay

Olah, C. (2023, May 24.) Interpretability dreams. https://transformer-circuits.pub/2023/interpretability-dreams/index.html

Olah, C., Cammarata, N., Schubert, L., Goh, G., & Petrov, M. (2020). Zoom In: An Introduction to Circuits. Distill. Retrieved from https://distill.pub/2020/circuits/zoom-in/

Olcott, E. (2024, October 19). Chinese AI groups get creative to drive down cost of models. Financial Times. https://www.ft.com/content/0a6da1bb-2bda-40f3-9645-97877eb0947c?

OpenAI. (n.d.). Data partnerships. OpenAI. https://openai.com/index/data-partnerships/

OpenAI. (n.d.). Embeddings. OpenAI. https://platform.openai.com/docs/guides/embeddings

OpenAI. (n.d.). tiktoken. GitHub. https://github.com/openai/tiktoken

OpenAI. (2024a, May 13). Hello GPT-4o. OpenAI. https://openai.com/index/hello-gpt-4o/

OpenAI. (2024b, October 31). Introducing ChatGPT Search. OpenAI. https://openai.com/index/introducing-chatgpt-search/

OpenAI. (2024c, January 25). New embedding models and API updates. OpenAI. https://openai.com/index/new-embedding-models-and-api-updates/

Parashar, K. (2024, December 11). Agentic chunking: enhancing RAG answers for completeness and accuracy. The Gleen Blog: Where Generative AI & CS/CX Intersect. https://gleen.ai/blog/agentic-chunking-enhancing-rag-answers-for-completeness-and-accuracy/

Pearson, H. (2025). Are the Internet and AI affecting our memory? What the science says. Nature. https://doi.org/10.1038/d41586-025-00292-z

Pedrazzini, F. (2024, December 16). RAG vs Long-Context LLMs: Which Approach Excels in Real-World Applications? https://blog.premai.io/rag-vs-long-context-llms-which-approach-excels-in-real-world-applications/

Phi-4 Hugging Face Model Card. (2024, October 2024). https://huggingface.co/microsoft/phi-4

Pinecone. (n.d.). Hierarchical Navigable Small Worlds (HNSW). Pinecone. https://www.pinecone.io/learn/series/faiss/hnsw/

Praetzellis, M. (2024). Limit your crawl to only collect PDFs. Archive-It Help Center. https://support.archive-it.org/hc/en-us/articles/208332953-Limit-your-crawl-to-only-collect-PDFs

Prompt Engineering Guide. (2024, September 18). Mistral 7B LLM. Prompt Engineering Guide. https://www.promptingguide.ai/models/mistral-7b

Qiu, J., Lv, H., Jin, Z., Wang, R., Ning, W., Yu, J., Zhang, C., Li, Z., Chu, P., Qu, Y., Shi, J., Lu, L., Peng, R., Zeng, Z., Tang, H., Lei, Z., Hong, J., Chen, K., Fei, Z., . . . He, C. (2024, February 29). WanJuan-CC: a Safe and High-Quality Open-sourced English Webtext dataset. arXiv.org. https://arxiv.org/abs/2402.19282

Restack. (2025, February 1). Transfer learning limitations explained. Restack. https://www.restack.io/p/transfer-learning-answer-limitations-cat-ai

Reuters. (2024, September 5). OpenAI hits more than 1 million paid business users. Reuters. https://www.reuters.com/technology/artificial-intelligence/openai-considers-pricier-subscriptions-its-chatbot-ai-information-reports-2024-09-05/

Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., Ferret, J., Liu, P., Tafti, P., Friesen, A., Casbon, M., Ramos, S., Kumar, R., Lan, C. L., Jerome, S., . . . Andreev, A. (2024, July 31). Gemma 2: Improving open language models at a practical size. arXiv.org. https://arxiv.org/abs/2408.00118

Rothman, J. (2023, November 13). Why the Godfather of A.I. Fears What He's Built. The New Yorker.
https://www.newyorker.com/magazine/2023/11/20/geoffrey-hinton-profile-ai

Sarker, M. K., Zhou, L., Eberhart, A., & Hitzler, P. (2021). Neuro-Symbolic Artificial Intelligence: Current Trends. AI Communications, 34(3), 197—209. Retrieved from https://arxiv.org/abs/2105.05330

Seetharaman, D. (2024a, April 1). Why OpenAI and other Data-Hungry AI companies need a bigger Internet. WSJ.
https://www.wsj.com/tech/ai/ai-training-data-synthetic-openai-anthropic-9230f8d8

Seetharaman, D. (2024b, December 21). OpenAI's next big AI effort, GPT-5, is behind schedule and crazy expensive. WSJ.
https://www.wsj.com/tech/ai/openai-gpt5-orion-delays-639e7693

Seneca. (n.d.). Moral letters to Lucilius: Letter 22. Wikisource.
https://en.wikisource.org/wiki/Moral_letters_to_Lucilius/Letter_22

Sennrich, R., Haddow, B., & Birch, A. (2016). Neural Machine Translation of Rare Words with Subword Units. Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). https://doi.org/10.18653/v1/p16-1162

Sennrich, R. (2015). Neural machine translation of rare words with subword units. arXiv preprint arXiv:1508.07909.
https://cs.brown.edu/courses/csci1460/assets/papers/nmt-of-rare-words.pdf

Stern, J. (2025, January 29). Oh dear, did someone steal something from OpenAI? WSJ.
https://www.wsj.com/tech/ai/oh-dear-did-someone-steal-something-from-openai-8e8a267c

SuperAnnotate. (2024, July 18). LLM evaluation: Metrics, frameworks, and best practices. SuperAnnotate. https://www.superannotate.com/blog/llm-evaluation-guide

Thompson, B., Dhaliwal, M. P., Frisch, P., Domhan, T., & Federico, M. (2024, January 11). A Shocking Amount of the Web is Machine Translated: Insights from Multi-Way Parallelism. arXiv.org. https://arxiv.org/abs/2401.05749

Titotal. (2023, September 29). “Diamondoid bacteria” nanobots: deadly threat or dead-end? A nanotech investigation [Online forum post]. https://www.lesswrong.com/posts/bc8Ssx5ys6zqu3eq9/diamondoid-bacteria-nanobots-deadly-threat-or-dead-end-a

Tronathan. (n.d.). Why does attention need to be fully quadratic? : r/LocalLLaMA. https://www.reddit.com/r/LocalLLaMA/comments/150owmj/why_does_attention_need_to_be_fully_quadratic/?rdt=35915

Tufekci, Z. (2025, February 5). DeepSeek and the dangerous A.I. nonsense that Trump and Biden fell for. The New York Times. https://www.nytimes.com/2025/02/05/opinion/ai-deepseek-trump-biden.html

UBC Science. (n.d.). ChatGPT has read almost the whole internet. That hasn’t solved its diversity issues. https://science.ubc.ca/news/chatgpt-has-read-almost-whole-internet-hasnt-solved-its-diversity-issues

Tufekci, Z. (2025, February 5). Opinion | DeepSeek and the dangerous A.I. nonsense that Trump and Biden fell for. The New York Times. https://www.nytimes.com/2025/02/05/opinion/ai-deepseek-trump-biden.html

Ulmer, T. (2024). About quality assurance. Archive-It Help Center. https://support.archive-it.org/hc/en-us/articles/208333833-About-Quality-Assurance

Vaswani, A. (2017). Attention is all you need. Advances in Neural Information Processing Systems. https://user.phil.hhu.de/~cwurm/wp-content/uploads/2020/01/7181-attention-is-all-you-need.pdf

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., & Titov, I. (2019). Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 5797-5808. https://doi.org/10.18653/v1/P19-1580

Wang, S., Sun, X., Li, X., Ouyang, R., Wu, F., Zhang, T., ... & Wang, G. (2023). Gpt-ner: Named entity recognition via large language models. arXiv preprint arXiv:2304.10428. https://arxiv.org/abs/2304.10428

Weigle, M. C., Nelson, M. L., Alam, S., & Graham, M. (2023, May 1). Right HTML, Wrong JSON: Challenges in Replaying Archived Webpages Built with Client-Side Rendering. arXiv.org. https://arxiv.org/abs/2305.01071

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., ... & Fedus, W. (2022). Emergent abilities of large language models. arXiv preprint arXiv:2206.07682. https://arxiv.org/abs/2206.07682

Wei, J. & Bommasani, R. (2022, September 13). Examining emergent abilities in large language models. Stanford HAI. https://hai.stanford.edu/news/examining-emergent-abilities-large-language-models

What comes after LLMs? : r/singularity. (n.d.). https://www.reddit.com/r/singularity/comments/1519iip/what_comes_after_llms/

What is Synthetic Data? - Synthetic Data Explained - AWS. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/what-is/synthetic-data/

Wikipedia contributors. (2025a, January 17). Large language model. Wikipedia. https://en.wikipedia.org/wiki/Large_language_model

Wikipedia contributors. (2025b, January 13). GPT-4. Wikipedia. https://en.wikipedia.org/wiki/GPT-4

Wikipedia contributors. (2025c, January 17). Retrieval-augmented generation. Wikipedia. https://en.wikipedia.org/wiki/Retrieval-augmented_generation

Wikipedia contributors. (2025d, January 18). BERT (language model). Wikipedia. https://en.wikipedia.org/wiki/BERT_(language_model)

Wong, M. (2024, December 10). The GPT era is already ending. The Atlantic. https://www.theatlantic.com/technology/archive/2024/12/openai-o1-reasoning-models/680906/?

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., & Narasimhan, K. (2024). Tree of thoughts: Deliberate problem solving with large language models. Advances in Neural Information Processing Systems, 36. https://arxiv.org/pdf/2305.10601

YouGov. (2022, November 2). How often and why do Americans rewatch TV shows? YouGov. https://today.yougov.com/entertainment/articles/45591-how-often-and-why-do-americans-rewatch-tv-shows

Yudkowsky, E. (2023, March 29). Pausing AI Developments Isn't Enough. We Need to Shut it All Down. TIME. https://time.com/6266923/ai-eliezer-yudkowsky-open-letter-not-enough/

Zeenea. (2024). Harnessing the Power of AI in Data Cataloging. Retrieved from https://zeenea.com/harnessing-the-power-of-ai-in-data-cataloging/

Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., ... & Du, M. (2024). Explainability for large language models: A survey. ACM Transactions on Intelligent Systems and Technology, 15(2), 1-38. https://dl.acm.org/doi/10.1145/3639372

Zilliz. (n.d.). The guide to e5-large-v2 model. Zilliz. https://zilliz.com/ai-models/e5-large-v2