



Search Medium

Write

Sign up

Sign In



★ Member-only story

RAG vs Finetuning — Which Is the Best Tool to Boost Your LLM Application?

The definitive guide for choosing the right method for your use case

[Heiko Hotz](#) · [Follow](#)Published in [Towards Data Science](#) · 19 min read · Aug 24

1.94K



16

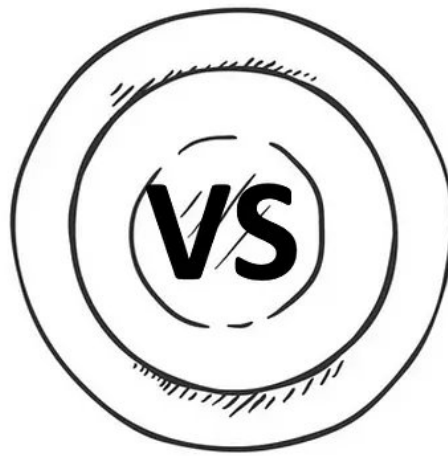
**RAG****Finetuning**

Image by author

Prologue

As the wave of interest in Large Language Models (LLMs) surges, many developers and organisations are busy building applications harnessing their power. However, when the pre-trained LLMs out of the box don't perform as expected or hoped, the question on how to improve the performance of the LLM application. And eventually we get to the point of where we ask

ourselves: Should we use Retrieval-Augmented Generation (RAG) or model finetuning to improve the results?

Before diving deeper, let's demystify these two methods:

RAG: This approach integrates the power of retrieval (or searching) into LLM text generation. It combines a retriever system, which fetches relevant document snippets from a large corpus, and an LLM, which produces answers using the information from those snippets. In essence, RAG helps the model to “look up” external information to improve its responses.

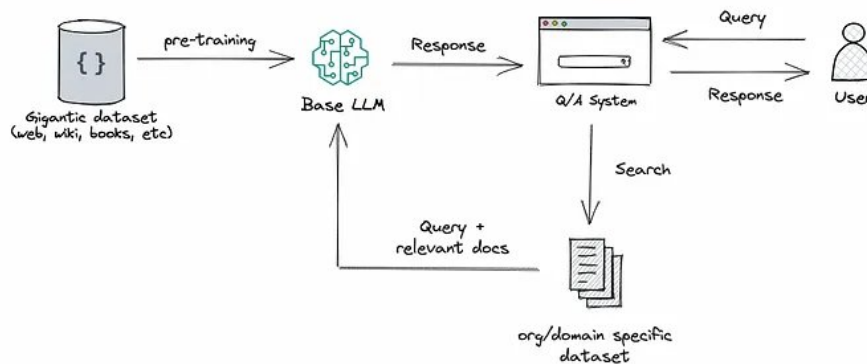


Image by author

Finetuning: This is the process of taking a pre-trained LLM and further training it on a smaller, specific dataset to adapt it for a particular task or to improve its performance. By finetuning, we are adjusting the model's weights based on our data, making it more tailored to our application's unique needs.

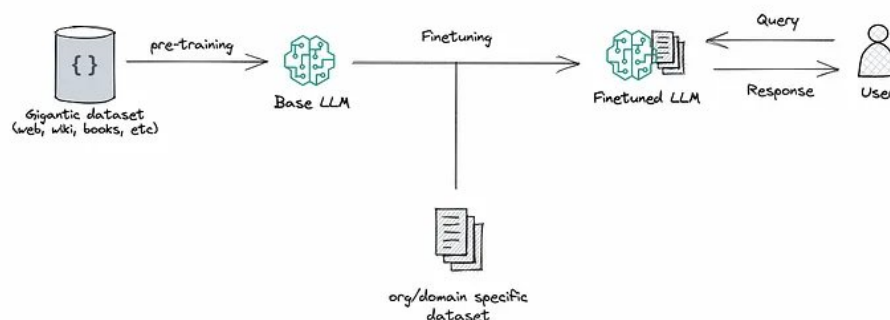


Image by author

Both RAG and finetuning serve as powerful tools in enhancing the performance of LLM-based applications, but they address different aspects of the optimisation process, and this is crucial when it comes to choosing one over the other.

Previously, I would often suggest to organisations that they experiment with RAG before diving into finetuning. This was based on my perception that both approaches achieved similar results but varied in terms of complexity, cost, and quality. I even used to illustrate this point with diagrams such as this one:



Image by author

In this diagram, various factors like complexity, cost, and quality are represented along a single dimension. The takeaway? RAG is simpler and less expensive, but its quality might not match up. My advice usually was: start with RAG, gauge its performance, and if found lacking, shift to finetuning.

However, my perspective has since evolved. I believe it's an oversimplification to view RAG and finetuning as two techniques that achieve the same result, just where one is just cheaper and less complex than the other. They are fundamentally distinct — instead of *co-linear* they are actually *orthogonal* — and serve different requirements of an LLM application.

To make this clearer, consider a simple real-world analogy: When posed with the question, “Should I use a knife or a spoon to eat my meal?”, the most logical counter-question is: “Well, what are you eating?” I asked friends and family this question and everyone instinctively replied with that counter-question, indicating that they don't view the knife and spoon as interchangeable, or one as an inferior variant of the other.

What is this about?

In this blog post, we'll dive deep into the nuances that differentiate RAG and finetuning across various dimensions that are, in my opinion, crucial in determining the optimal technique for a specific task. Moreover, we'll be looking at some of the most popular use cases for LLM applications and use the dimensions established in the first part to identify which technique

might be best suited for which use case. In the last part of this blog post we will identify additional aspects that should be considered when building LLM applications. Each one of those might warrant its own blog post and therefore we can only touch briefly on them in the scope of this post.

Why should you care?

Choosing the right technique for adapting large language models can have a major impact on the success of your NLP applications. Selecting the wrong approach can lead to:

- Poor model performance on your specific task, resulting in inaccurate outputs.
- Increased compute costs for model training and inference if the technique is not optimized for your use case.
- Additional development and iteration time if you need to pivot to a different technique later on.
- Delays in deploying your application and getting it in front of users.
- A lack of model interpretability if you choose an overly complex adaptation approach.
- Difficulty deploying the model to production due to size or computational constraints.

The nuances between RAG and finetuning span model architecture, data requirements, computational complexity, and more. **Overlooking these details can derail your project timeline and budget.**

This blog post aims to prevent wasted effort by clearly laying out when each technique is advantageous. With these insights, you can hit the ground running with the right adaptation approach from day one. The detailed comparison will equip you to make the optimal technology choice to achieve your business and AI goals. **This guide to selecting the right tool for the job will set your project up for success.**

So let's dive in!

. . .

Key considerations for boosting performance

Before we choose RAG vs Finetuning, we should assess the requirements of our LLM project along some dimensions and ask ourselves a few questions.

Does our use case require access to external data sources?

When choosing between finetuning an LLM or using RAG, one key consideration is whether the application requires access to external data sources. If the answer is yes, RAG is likely the better option.

RAG systems are, by definition, designed to augment an LLM's capabilities by retrieving relevant information from knowledge sources before generating a response. This makes this technique well-suited for applications that need to query databases, documents, or other structured/unstructured data repositories. The retriever and generator components can be optimised to leverage these external sources.

In contrast, while it is possible to finetune an LLM to learn some external knowledge, doing so requires a large labelled dataset of question-answer pairs from the target domain. This dataset must be updated as the underlying data changes, making it impractical for frequently changing data sources. The finetuning process also does not explicitly model the retrieval and reasoning steps involved in querying external knowledge.

So in summary, if our application needs to leverage external data sources, using a RAG system will likely be more effective and scalable than attempting to “bake in” the required knowledge through finetuning alone.

Do we need to modify the model's behaviour, writing style, or domain-specific knowledge?

Another very important aspect to consider is how much we need the model to adjust its behaviour, its writing style, or tailor its responses for domain-specific applications.

Finetuning excels in its ability to adapt an LLM's behaviour to specific nuances, tones, or terminologies. If we want the model to sound more like a medical professional, write in a poetic style, or use the jargon of a specific industry, finetuning on domain-specific data allows us to achieve these customisations. This ability to influence the model's behaviour is essential for applications where alignment with a particular style or domain expertise is vital.

RAG, while powerful in incorporating external knowledge, primarily focuses on information retrieval and doesn't inherently adapt its linguistic style or domain-specificity based on the retrieved information. It will pull relevant content from the external data sources but might not exhibit the tailored nuances or domain expertise that a finetuned model can offer.

So, if our application demands specialised writing styles or deep alignment with domain-specific vernacular and conventions, finetuning presents a more direct route to achieving that alignment. It offers the depth and customisation necessary to genuinely resonate with a specific audience or

expertise area, ensuring the generated content feels authentic and well-informed.

Quick recap

These two aspects are by far the most important ones to consider when deciding which method to use to boost LLM application performance. Interestingly, they are, in my opinion, orthogonal and can be used independently (and also be combined).

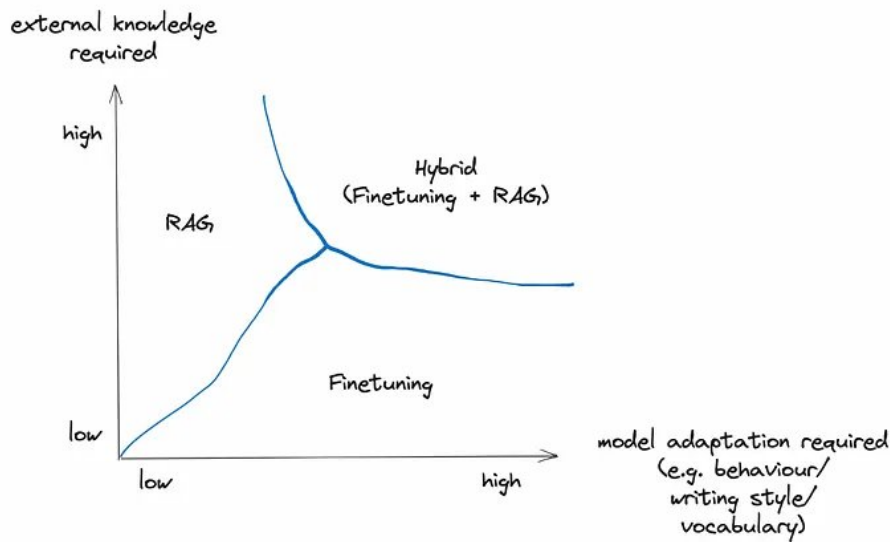


Image by author

. . .

However, before diving into the use cases, there are a few more key aspects we should consider before choosing a method:

How crucial is it to suppress hallucinations?

One downside of LLMs is their tendency to hallucinate — making up facts or details that have no basis in reality. This can be highly problematic in applications where accuracy and truthfulness are critical.

Finetuning can help reduce hallucinations to some extent by grounding the model in a specific domain's training data. However, the model may still fabricate responses when faced with unfamiliar inputs. Retraining on new data is required to continuously minimise false fabrications.

In contrast, RAG systems are inherently less prone to hallucination because they ground each response in retrieved evidence. The retriever identifies relevant facts from the external knowledge source before the generator constructs the answer. This retrieval step acts as a fact-checking mechanism, reducing the model's ability to confabulate. The generator is constrained to

synthesise a response supported by the retrieved context.

So in applications where suppressing falsehoods and imaginative fabrications is vital, RAG systems provide in-built mechanisms to minimise hallucinations. The retrieval of supporting evidence prior to response generation gives RAG an advantage in ensuring factually accurate and truthful outputs.

How much labelled training data is available?

When deciding between RAG and finetuning, a crucial factor to consider is the volume of domain- or task-specific, labelled training data at our disposal.

Finetuning an LLM to adapt to specific tasks or domains is heavily dependent on the quality and quantity of the labelled data available. A rich dataset can help the model deeply understand the nuances, intricacies, and unique patterns of a particular domain, allowing it to generate more accurate and contextually relevant responses. However, if we're working with a limited dataset, the improvements from finetuning might be marginal. In some cases, a scant dataset might even lead to overfitting, where the model performs well on the training data but struggles with unseen or real-world inputs.

On the contrary, RAG systems are independent from training data because they leverage external knowledge sources to retrieve relevant information. Even if we don't have an extensive labelled dataset, a RAG system can still perform competently by accessing and incorporating insights from its external data sources. The combination of retrieval and generation ensures that the system remains informed, even when domain-specific training data is sparse.

In essence, if we have a wealth of labelled data that captures the domain's intricacies, finetuning can offer a more tailored and refined model behaviour. But in scenarios where such data is limited, a RAG system provides a robust alternative, ensuring the application remains data-informed and contextually aware through its retrieval capabilities.

How static/dynamic is the data?

Another fundamental aspect to consider when choosing between RAG and finetuning is the dynamic nature of our data. How frequently is the data updated, and how imperative is it for the model to stay current?

Finetuning an LLM on a specific dataset means the model's knowledge becomes a static snapshot of that data at the time of training. If the data undergoes frequent updates, changes, or expansions, this can quickly render the model outdated. To keep the LLM current in such dynamic environments, we'd have to retrain it frequently, a process that can be both

time-consuming and resource-intensive. Additionally, each iteration requires careful monitoring to ensure that the updated model still performs well across different scenarios and hasn't developed new biases or gaps in understanding.

In contrast, RAG systems inherently possess an advantage in environments with dynamic data. Their retrieval mechanism constantly queries external sources, ensuring that the information they pull in for generating responses is up-to-date. As the external knowledge bases or databases update, the RAG system seamlessly integrates these changes, maintaining its relevance without the need for frequent model retraining.

In summary, if we're grappling with a rapidly evolving data landscape, RAG offers an agility that's hard to match with traditional finetuning. By always staying connected to the most recent data, RAG ensures that the responses generated are in tune with the current state of information, making it an ideal choice for dynamic data scenarios.

How transparent/interpretable does our LLM app need to be?

The last aspect to consider is the degree to which we need insights into the model's decision-making process.

Finetuning an LLM, while incredibly powerful, operates like a black box, making the reasoning behind its responses more opaque. As the model internalises the information from the dataset, it becomes challenging to discern the exact source or reasoning behind each response. This can make it difficult for developers or users to trust the model's outputs, especially in critical applications where understanding the "why" behind an answer is vital.

RAG systems, on the other hand, offer a level of transparency that's not typically found in solely finetuned models. Given the two-step nature of RAG – retrieval and then generation – users can peek into the process. The retrieval component allows for the inspection of which external documents or data points are selected as relevant. This provides a tangible trail of evidence or reference that can be evaluated to understand the foundation upon which a response is built. The ability to trace back a model's answer to specific data sources can be invaluable in applications that demand a high degree of accountability or when there's a need to validate the accuracy of the generated content.

In essence, if transparency and the ability to interpret the underpinnings of a model's responses are priorities, RAG offers a clear advantage. By breaking down the response generation into distinct stages and allowing insight into its data retrieval, RAG fosters greater trust and understanding in its outputs.

Summary

Choosing between RAG and finetuning becomes more intuitive when considering these dimensions. If we need lean towards accessing external knowledge and valuing transparency, RAG is our go-to. On the other hand, if we're working with stable labelled data and aim to adapt the model more closely to specific needs, finetuning is the better choice.

	RAG	Finetuning
External knowledge req'd?	✓	✗
Changing model behaviour req'd?	✗	✓
Minimise hallucinations?	✓	✗
Training data available?	✗	✓
Is data (mostly) dynamic?	✓	✗
Interpretability req'd?	✓	✗

Image by author

In the following section, we'll see how we can assess popular LLM use cases based on these criteria.

...

Use cases

Let's look at some popular use cases and how the above framework can be used to choose the right method:

Summarisation (in a specialised domain and/or a specific style)

1. **External knowledge required?** For the task of summarizing in the style of previous summaries, the primary data source would be the previous summaries themselves. If these summaries are contained within a static dataset, there's little need for continuous external data retrieval. However, if there's a dynamic database of summaries that frequently updates and the

goal is to continually align the style with the newest entries, RAG might be useful here.

2. Model adaptation required? The core of this use case revolves around adapting to a specialised domain or a and/or a specific writing style. Finetuning is particularly adept at capturing stylistic nuances, tonal variations, and specific domain vocabularies, making it an optimal choice for this dimension.

3. Crucial to minimise hallucinations? Hallucinations are problematic in most LLM applications, including summarisation. However, in this use case, the text to be summarised is typically provided as context. This makes hallucinations less of a concern compared to other use cases. The source text constrains the model, reducing imaginative fabrications. So while factual accuracy is always desirable, suppressing hallucinations is a lower priority for summarisation given the contextual grounding.

4. Training data available? If there's a substantial collection of previous summaries that are labelled or structured in a way that the model can learn from them, finetuning becomes a very attractive option. On the other hand, if the dataset is limited, and we're leaning on external databases for stylistic alignment, RAG could play a role, although its primary strength isn't style adaptation.

5. How dynamic is the data? If the database of previous summaries is static or updates infrequently, the finetuned model's knowledge will likely remain relevant for a longer time. However, if the summaries update frequently and there's a need for the model to align with the newest stylistic changes continuously, RAG might have an edge due to its dynamic data retrieval capabilities.

6. Transparency/Interpretability required? The primary goal here is stylistic alignment, so the "why" behind a particular summarisation style might be less critical than in other use cases. That said, if there's a need to trace back and understand which previous summaries influenced a particular output, RAG offers a bit more transparency. Still, this might be a secondary concern for this use case.

***Recommendation:** For this use case **finetuning** appears to be the more fitting choice. The primary objective is stylistic alignment, a dimension where finetuning shines. Assuming there's a decent amount of previous summaries available for training, finetuning an LLM would allow for deep adaptation to the desired style, capturing the nuances and intricacies of the domain. However, if the summaries database is extremely dynamic and there's value in tracing back influences, considering a hybrid approach or leaning towards*

Question/answering system on organisational knowledge (i.e. external data)

1. External knowledge required? A question/answering system relying on organisational knowledge bases inherently requires access to external data, in this case, the org's internal databases and document stores. The system's effectiveness hinges on its ability to tap into and retrieve relevant information from these sources to answer queries. Given this, RAG stands out as the more suitable choice for this dimension, as it's designed to augment LLM capabilities by retrieving pertinent data from knowledge sources.

2. Model adaptation required? Depending on the organization and its field, there might be a requirement for the model to align with specific terminologies, tones, or conventions. While RAG focuses primarily on information retrieval, finetuning can help the LLM adjust its responses to the company's internal vernacular or the nuances of its domain. Thus, for this dimension, depending on the specific requirements finetuning might play a role.

3. Crucial to minimise hallucinations? Hallucinations are a major concern in this use case, due to the knowledge-cutoff of LLMs. If the model is unable to answer a question based on the data it has been trained on, it will almost certainly revert to (partially or entirely) making up a plausible but incorrect answer.

4. Training data available? If the organization has a structured and labeled dataset of previously answered questions, this can bolster the finetuning approach. However, not all internal databases are labeled or structured for training purposes. In scenarios where the data isn't neatly labeled or where the primary focus is on retrieving accurate and relevant answers, RAG's ability to tap into external data sources without needing a vast labeled dataset makes it a compelling option.

5. How dynamic is the data? Internal databases and document stores in organisations can be highly dynamic, with frequent updates, changes, or additions. If this dynamism is characteristic of the organisation's knowledge base, RAG offers a distinct advantage. It continually queries the external sources, ensuring its answers are based on the latest available data. Finetuning would require regular retraining to keep up with such changes, which might be impractical.

6. Transparency/Interpretability required? For internal applications, especially in sectors like finance, healthcare, or legal, understanding the reasoning or source behind an answer can be paramount. Since RAG

provides a two-step process of retrieval and then generation, it inherently offers a clearer insight into which documents or data points influenced a particular answer. This traceability can be invaluable for internal stakeholders who might need to validate or further investigate the sources of certain answers.

***Recommendation:** For this use case a RAG system seems to be the more fitting choice. Given the need for dynamic access to the organisation's evolving internal databases and the potential requirement for transparency in the answering process, RAG offers capabilities that align well with these needs. However, if there's a significant emphasis on tailoring the model's linguistic style or adapting to domain-specific nuances, incorporating elements of finetuning could be considered.*

Customer Support Automation (i.e. automated chatbots or help desk solutions providing instant responses to customer inquiries)

1. External knowledge required? Customer support often necessitates access to external data, especially when dealing with product details, account-specific information, or troubleshooting databases. While many queries can be addressed with general knowledge, some might require pulling data from company databases or product FAQs. Here, RAG's capability to retrieve pertinent information from external sources would be beneficial. However, it's worth noting that a lot of customer support interactions are also based on predefined scripts or knowledge, which can be effectively addressed with a finetuned model.

2. Model adaptation required? Customer interactions demand a certain tone, politeness, and clarity, and might also require company-specific terminologies. Finetuning is especially useful for ensuring the LLM adapts to the company's voice, branding, and specific terminologies, ensuring a consistent and brand-aligned customer experience.

3. Crucial to minimise hallucinations? For customer support chatbots, avoiding false information is essential to maintain user trust. Finetuning alone leaves models prone to hallucinations when faced with unfamiliar queries. In contrast, RAG systems suppress fabrications by grounding responses in retrieved evidence. This reliance on sourced facts allows RAG chatbots to minimise harmful falsehoods and provide users with reliable information where accuracy is vital.

4. Training data available? If a company has a history of customer interactions, this data can be invaluable for finetuning. A rich dataset of previous customer queries and their resolutions can be used to train the model to handle similar interactions in the future. If such data is limited, RAG can provide a fallback by retrieving answers from external sources like product documentation.

5. How dynamic is the data? Customer support might need to address queries about new products, updated policies, or changing service terms. In scenarios where the product line up, software versions, or company policies are frequently updated, RAG's ability to dynamically pull from the latest documents or databases is advantageous. On the other hand, for more static knowledge domains, finetuning can suffice.

6. Transparency/Interpretability required? While transparency is essential in some sectors, in customer support, the primary focus is on accurate, fast, and courteous responses. However, for internal monitoring, quality assurance, or addressing customer disputes, having traceability regarding the source of an answer could be beneficial. In such cases, RAG's retrieval mechanism offers an added layer of transparency.

Recommendation: For customer support automation a **hybrid approach** might be optimal. Finetuning can ensure that the chatbot aligns with the company's branding, tone, and general knowledge, handling the majority of typical customer queries. RAG can then serve as a complementary system, stepping in for more dynamic or specific inquiries, ensuring the chatbot can pull from the latest company documents or databases and thereby minimising hallucinations. By integrating both approaches, companies can provide a comprehensive, timely, and brand-consistent customer support experience.

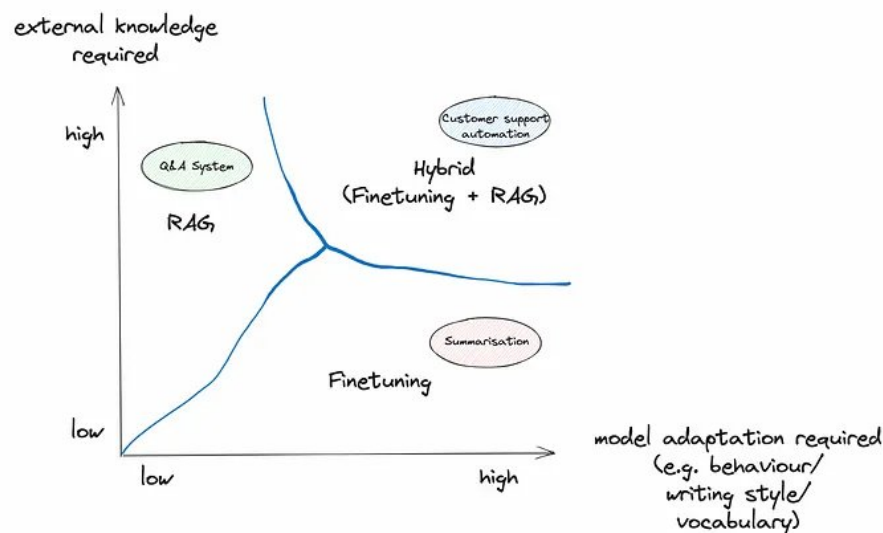


Image by author

...

Additional aspects to consider

As mentioned above, there are other factors that should be considered when deciding between RAG and finetuning (or both). We can't possibly dive deep

into them, as all of them are multi-faceted and don't have clear answers like some of the aspects above (for example, if there is no training data the finetuning is just simply not possible). But that doesn't mean we should disregard them:

Scalability

As an organisation grows and its needs evolve, how scalable are the methods in question? RAG systems, given their modular nature, might offer more straightforward scalability, especially if the knowledge base grows. On the other hand, frequently finetuning a model to cater to expanding datasets can be computationally demanding.

Latency and Real-time Requirements

If the application requires real-time or near-real-time responses, consider the latency introduced by each method. RAG systems, which involve retrieving data before generating a response, might introduce more latency compared to a finetuned LLM that generates responses based on internalised knowledge.

Maintenance and Support

Think about the long-term. Which system aligns better with the organisation's ability to provide consistent maintenance and support? RAG might require upkeep of the database and the retrieval mechanism, while finetuning would necessitate consistent retraining efforts, especially if the data or requirements change.

Robustness and Reliability

How robust is each method to different types of inputs? While RAG systems can pull from external knowledge sources and might handle a broad array of questions, a well finetuned model might offer more consistency in certain domains.

Ethical and Privacy Concerns

Storing and retrieving from external databases might raise privacy concerns, especially if the data is sensitive. On the other hand, a finetuned model, while not querying live databases, might still produce outputs based on its training data, which could have its own ethical implications.

Integration with Existing Systems

Organisations might already have certain infrastructure in place. The compatibility of RAG or finetuning with existing systems — be it databases, cloud infrastructures, or user interfaces — can influence the choice.

User Experience

Consider the end-users and their needs. If they require detailed, reference-backed answers, RAG could be preferable. If they value speed and domain-

specific expertise, a finetuned model might be more suitable.

Cost

Finetuning can get expensive, especially for really large models. But in the past few months costs have gone down significantly thanks to parameter efficient techniques like [QLoRA](#). Setting up RAG can be a large initial investment — covering the integration, database access, maybe even licensing fees — but then there's also the regular maintenance of that external knowledge base to think about.

Complexity

Finetuning can get complex quickly. While many providers now offer one-click finetuning where we just need to provide the training data, keeping track of model versions and ensuring that the new models still perform well across the board is challenging. RAG, on the other hand, can also get complex quickly. There's the setup of multiple components, making sure the database stays fresh, and ensuring the pieces — like retrieval and generation — fit together just right.

. . .

Conclusion

As we've explored, choosing between RAG and finetuning requires a nuanced evaluation of an LLM application's unique needs and priorities. There is no one-size-fits-all solution; success lies in aligning the optimisation method with the specific requirements of the task. By assessing key criteria — the need for external data, adapting model behaviour, training data availability, data dynamics, result transparency, and more — organisations can make an informed decision on the best path forward. In certain cases, a hybrid approach leveraging both RAG and finetuning may be optimal.

The key is avoiding assumptions that one method is universally superior. Like any tool, their suitability depends on the job at hand. Misalignment of approach and objectives can hinder progress, while the right method accelerates it. As an organisation evaluates options for boosting LLM applications, it must resist oversimplification and not view RAG and finetuning as interchangeable and choose the tool that empowers the model to fulfil its capabilities aligned to the needs of the use case. The possibilities these methods unlock are astounding but possibility alone isn't enough — execution is everything. The tools are here — now let's put them to work.


. . .

Heiko Hotz

👉 Follow me on [Medium](#) and [LinkedIn](#) to read more about Generative AI, Machine Learning, and Natural Language Processing.

👥 If you're based in London join one of our [NLP London Meetups](#).

📰 My thoughts on AI News on [Naughty Neural](#).



Heiko Hotz (AI Expert, he/him)
Generative AI @ AWS ♦ Founder of NLP London ♦ AI Consultant ♦ I work with tech leaders to increase adoption of AI within their organisation. If you'd like to know how, check out my "ABOUT" section below.

AI/ML Consulting
London Business School

Image by author

[Llm](#)

[AI](#)

[Machine Learning](#)

[Mlops](#)

[Editors Pick](#)

👏 1.94K

💬 16



Written by Heiko Hotz

Follow



[1.98K Followers](#) · Writer for [Towards Data Science](#)

Senior Solutions Architect for Generative AI @ AWS — All opinions are my own

More from Heiko Hotz and Towards Data Science





[Heiko Hotz](#) in [Towards Data Science](#)

Build Your Own ChatGPT-Like App with Streamlit

Leverage OpenAI's APIs to bypass the official ChatGPT app

6 min read · Apr 3

421 7



[Giuseppe Scalamogna](#) in [Towards Data Science](#)

New ChatGPT Prompt Engineering Technique: Program Simulation

A potentially novel technique for turning a ChatGPT prompt into a mini-app.

9 min read · Sep 3

1.1K 11



[Maarten Grootendorst](#) in [Towards Data Science](#)

Topic Modeling with Llama 2

Create easily interpretable topics with Large Language Models

★ · 12 min read · Aug 22

673 9



[Heiko Hotz](#) in [Towards Data Science](#)

Create Your Own Stable Diffusion UI on AWS in Minutes

Deploy a text-to-image web app with just one command

8 min read · Jan 3

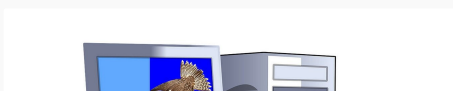
71 11

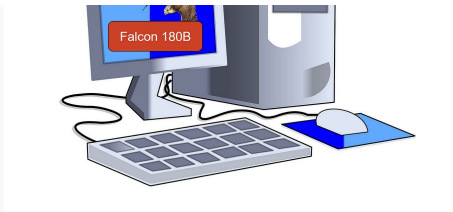


[See all from Heiko Hotz](#)

[See all from Towards Data Science](#)

Recommended from Medium





Benjamin Marie in [Towards Data Science](#)

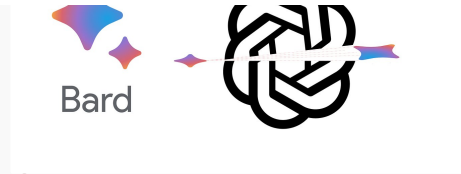
Falcon 180B: Can It Run on Your Computer?

Yes, if you have enough CPU RAM

🌟 : [7 min read](#) : [Sep 12](#)

938

5



AL Anany

The ChatGPT Hype Is Over — Now Watch How Google Will Kill...

It never happens instantly. The business game is longer than you know.

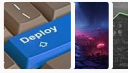
🌟 : [6 min read](#) : [Sep 1](#)

7.6K

236



Lists



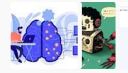
Predictive Modeling w/ Python

[20 stories](#) : [397 saves](#)



Natural Language Processing

[620 stories](#) : [234 saves](#)



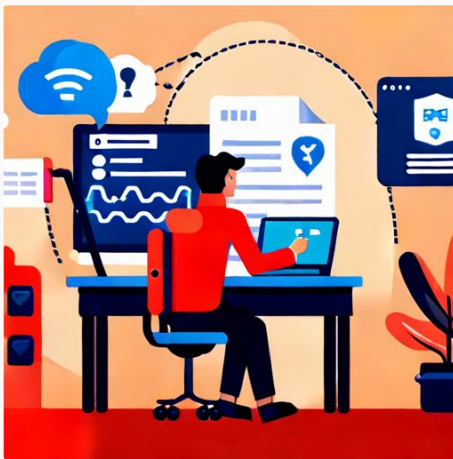
The New Chatbots: ChatGPT, Bard, and Beyond

[13 stories](#) : [123 saves](#)



Practical Guides to Machine Learning

[10 stories](#) : [457 saves](#)



Sachin Kulkarni

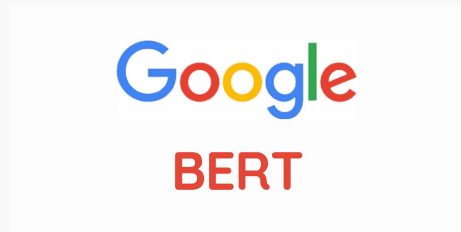
Generative AI with Enterprise Data

Create business value add Enterprise knowledge to Large Language Models

[6 min read](#) : [Jul 25](#)

196

5



Ravvan Shaikh

Mastering BERT: A Comprehensive Guide from Beginner to Advanced...

Introduction: A Guide to Unlocking BERT: From Beginner to Expert

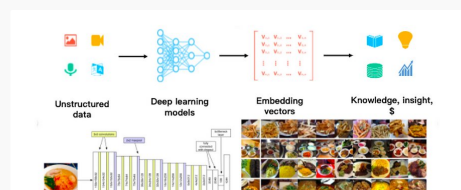
[19 min read](#) : [Aug 26](#)

1K


13



```
Python 3.9.12 Shell [~/Documents/Python/Projects/LLM]
├── install_necessary_libraries.py
├── main.py
├── requirements.txt
├── README.md
├── ...
└── ...
```



```
pip install --q...
Import following libraries
```

 [Maya Akim](#)

Complete Guide to LLM Fine Tuning for Beginners

Fine-tuning a model refers to the process of adapting a pre-trained, foundational model...

5 min read · Aug 13

 33 



 [Jayita Bhattacharyya](#) in [GoPenAI](#)

Primer on Vector Databases and Retrieval-Augmented Generation...

Vector Databases Generation (RAG)
Langchain Pinecone HuggingFace Large...

9 min read · Aug 16

 152 



[See more recommendations](#)