#19 TECHARTIKEL

**Fine-Tuning: Grundlagen, Best Practices und seine Rolle im GenKI-Ökosystem**

13.03.2025

Andreas Schmid



**Inhalt**

[1 Einführung und Funktionsweise 3](#_Toc192662788)

[2 Methoden, Ansätze und technische Realisierung 4](#_Toc192662789)

[3 Vergleich zu weiteren Modelloptimierungsansätzen 6](#_Toc192662790)

[4 Fazit und Ausblick 10](#_Toc192662792)

[5 Quellen 11](#_Toc192662793)

Fine-Tuning: Grundlagen, Best Practices und seine Rolle im GenKI-Ökosystem

**In unserem Blog haben wir bereits Prompt Engineering (**[**1**](https://aracom.de/prompt-engineering-effektive-strategien-fuer-die-interaktion-mit-sprachmodellen/) **&** [**2**](https://aracom.de/prompt-engineering-fortgeschrittene-techniken-fuer-praezise-ki-interaktion/)**) und** [**Retrieval Augmented Generation**](https://aracom.de/retrieval-augmented-generation/) **(RAG) als zwei prominente Vertreter zur Modelloptimierung von großen Sprachmodellen (engl. large language models – LLMs) vorgestellt. Doch wie es so schön heißt: Alle guten Dinge sind drei, und ein weiterer prominenter Ansatz, der im Ökosystem der generativen künstlichen Intelligenz (GenKI) nicht unerwähnt bleiben darf, ist Fine-Tuning.**

**Um diese Lücke zu schließen, klären wir in diesem Techartikel, wie Fine-Tuning funktioniert, welche Best Practices dabei beachtet werden sollten und in welchen Szenarien der Einsatz von Fine-Tuning einen echten Mehrwert bietet.**

# Einführung und Funktionsweise

Unter Fine-Tuning versteht man den Prozess der Anpassung eines vortrainierten LLMs (z.B. GPT-4, Gemini 1.0 oder Llama 2) an einen spezifischen Datensatz oder eine bestimmte Aufgabe, um die Leistung in diesem Bereich zu verbessern. Man spricht hier auch von einer Art Modell-Personalisierung, indem der Trainingsprozess des Vortrainings auf einem kleineren, domänenspezifischen Datensatz fortgesetzt wird [**[1, 2]**](#_Quellen). Abbildung 1 verdeutlicht diesen Prozess.



Abbildung 1: Prozess des Fine-Tunings

Zunächst wird ein LLM mit umfangreichen Trainingsdaten im Zuge des Vortrainings trainiert, woraus ein Basismodell des LLMs resultiert. Dieses Basismodell ist definiert durch seine Millionen (oftmals sogar Milliarden) von Parametern, die durch das Vortraining initialisiert werden. Diese Parameter bestimmen fortan das Modellverhalten, da sie die Gewichtungen und Verbindungen zwischen den Neuronen der einzelnen Schichten des Modells festlegen.

Der Trainingsprozess des Vortrainings wird anschließend mit Domänendaten, bspw. mit internen Geschäftsberichten, Artikeln aus Fachzeitschriften oder mit Dialogbeispielen aus Kundengesprächen, fortgeführt. Hierdurch werden die Parameter des Basismodells angepasst und deren Gewichte und Verzerrungen verändert. Das daraus entstandene Modell ist nun auf die zur Verfügung gestellten Daten abgestimmt und somit „finegetuned“, womit das Fine-Tuning abgeschlossen ist. Abschließend kann das Modell getestet und für den produktiven Einsatz bereitgestellt werden.

# Methoden, Ansätze und technische Realisierung

Genauer betrachtet kann der Prozess des Fine-Tunings mit diversen Methoden durchgeführt werden, die allesamt mit unterschiedlichen Vor- und Nachteilen einhergehen [**[2]**](#_Quellen). Übliche Ansätze sind in Abbildung 2 dargestellt.



Abbildung 2: Methoden und Ansätze

Neben der Wahl eines passenden Ansatzes stellt sich zudem die Frage, wie Fine-Tuning in der Praxis realisiert werden kann. Wie so oft führen auch beim Fine-Tuning viele Wege nach Rom. Einen einfachen Einstieg bieten gängige Clouddienste wie Azure mit der Azure AI Foundry [**[3]**](#_Quellen) oder Amazon Web Services mit der Amazon SageMaker AI [**[4]**](#_Quellen). Diese Plattformen erlauben das Fine-Tuning von ausgewählten LLMs mit wenigen Klicks. Alternativ können zahlreiche Open-Source-Modelle über Hugging Face **[5]**, eine Plattform für KI-Modelle und -Datensätze, heruntergeladen und mithilfe der hauseigenen Trainer-API von Transformers [**[6]**](#_Quellen) finegetuned und evaluiert werden. Eine weitere populäre Open-Source-Lösung ist Unsloth AI **[7]**, die ebenfalls mit dem Hugging Face Ökosystem kompatibel ist und bereits mit moderater Hardwareanforderung lokal eingesetzt werden kann. Zusätzlich ermöglichen Plattformen wie Google Colab [**[8]**](#_Quellen) über ihre Free-Tier-Modelle die kostenlose Anmietung leistungsstarker Hardware, die ebenfalls für den Trainingsprozess geeignet ist. Alternativ können auch kostenpflichtige Paid-Tier-Modelle genutzt werden, die noch mehr Ressourcen und Flexibilität bieten.

# Vergleich zu weiteren Modelloptimierungsansätzen

Wie eingangs bereits erwähnt, sind neben Fine-Tuning auch Prompt Engineering und RAG typische Modelloptimierungsansätze. Um zu verstehen, inwieweit sich diese unterscheiden, ordnet Abbildung 3 die Ansätze in Abhängigkeit der Eigenschaften „Notwendigkeit der Integration zusätzlicher Datenquellen“ und „Notwendigkeit der Modellanpassung“ ein.

#

Abbildung 3: Vergleich der Modelloptimierungsansätze

Während sich Prompt Engineering ausschließlich auf die Optimierung durch präzise Formulierungen und Anweisungen fokussiert und damit die einfachste Möglichkeit zur Modelloptimierung ist, nutzen RAG und Fine-Tuning jeweils unterschiedliche Ansätze, um die vortrainierte Wissensbasis von LLMs zu erweitern. RAG ermöglicht dies über die Integration zusätzlicher Datenquelle über den Kontext, wohingegen Fine-Tuning auf gezielte Modellanpassungen setzt.

Aufgrund der einfachen Integration sollte Prompt Engineering immer verwendet werden, um LLMs weiter zu optimieren. Die Wahl zwischen RAG und Fine-Tuning sollte auf den spezifischen Anforderungen des Szenarios und den inhärenten Eigenschaften der Ansätze basieren. Nachfolgend werden die beiden Ansätze detaillierter miteinander verglichen:

|  |  |  |
| --- | --- | --- |
| **Kategorie** | **RAG** | **Fine-Tuning** |
| **Wissensaktualisierung** | Die Informationsquellen können jederzeit um neue Inhalte erweitert werden. | Der Trainingsprozess muss im Falle einer Aktualisierung erneut durchgeführt werden. |
| **Rechenleistung** | Da Ähnlichkeitssuchen auf großen Datensätzen durchgeführt werden, ist die tägliche Rechenleistung höher als beim Fine-Tuning.  | Fine-Tuning benötigt eine geringere Rechenleistung verglichen mit dem Trainingsprozess eines LLMs. Dennoch ist der einmalige Prozess des Fine-Tunings abhängig von der Größe der Trainingsdaten zeit-, kosten- und rechenintensiv. |
| **Transparenz** | Antworten können zu den Datenquellen zurückverfolgt werden, wodurch eine hohe Transparenz gegeben ist. | Ähnlich wie bei einer Blackbox ist nicht immer klar, warum das LLM auf eine bestimmte Weise reagiert. |
| **Latenz** | Der Prozess der Informationssuche kann zu einer höheren Latenz führen. | Da das Modell nur eigene Trainingsdaten verwendet, ist die Latenz geringer als bei RAG. |
| **Modellanpassung** | Da das zugrundeliegende LLM nicht verändert wird, sind Änderungen des Verhaltens, Schreibstils etc. ausschließlich über Prompt Engineering möglich. | Da der Trainingsprozess des zugrundeliegenden LLMs fortgeführt wird, sind Anpassungen des Verhaltens, Schreibstils oder des spezifischen Domänenwissens möglich. |
| **Halluzinationen** | Da Antworten ausschließlich aus eigenen Dokumenten generiert werden, sind Halluzinationen seltener. Zudem kann über vordefinierte Prompts sichergestellt werden, dass das LLM standardisierte Antworten liefert, wenn keine Informationen gefunden werden. | Halluzinationen können reduziert werden, indem das LLM hinsichtlich spezifischer Domänen trainiert wird. Dennoch kann das Modell halluzinieren, wenn es keine Antwort auf unbekannte Fragen hat. |

Der Vergleich zeigt, dass beide Ansätze unterschiedliche Stärken und Schwächen besitzen. Während RAG mehr Transparenz, ein geringeres Halluzinationsrisiko sowie die einfache Integration neuer Datenquellen ermöglicht, profitiert Fine-Tuning von einer geringeren Latenz und der Möglichkeit der gezielten Modellanpassung, wodurch vor allem das Antwortverhalten, die Tonalität und die Ausgabestruktur personalisiert werden können [**[9, 10]**](#_Quellen). Folgende Fragestellungen können einen ersten Anhaltspunkt bieten, um eine fundierte Entscheidung zwischen den beiden Ansätzen zu treffen:

* Werden regelmäßig neue oder dynamische Daten benötigt? **🡪 RAG**
* Benötigt die Anwendung einen benutzerdefinierten Ton, ein bestimmtes Verhalten oder einen kundenspezifischen Wortschatz? **🡪 Fine-Tuning**
* Ist die Minimierung von Halluzinationen für die Anwendung entscheidend? **🡪 RAG**
* Stehen ausreichend gelabelte Trainingsdaten zur Verfügung? **🡪 Fine-Tuning**

Schlussfolgernd muss die Wahl zwischen RAG und Fine-Tuning je nach Anwendungsfall individuell entschieden werden. Denkbar ist zudem eine komplementäre Lösung unter Einbeziehung beider Ansätze.

# Fazit und Ausblick

Neben Prompt Engineering und RAG ist Fine-Tuning ein weiterer probater Ansatz, der die Optimierung und Spezialisierung von LLMs und GenKI an kundenindividuelle Bedürfnisse ermöglicht. Neue Techniken wie Parameter-Efficient-Fine-Tuning, die Integration in gängige Clouddienste und eine breite Palette an Open-Source-Lösungen machen Fine-Tuning zu einer zugänglichen Methode, die nicht mehr ausschließlich großen Tech-Giganten vorbehalten ist. Nichtsdestotrotz sind eine hohe Datenqualität in Form von sorgfältig kuratierten Trainingsdaten entscheidend, um LLMs präzise und verlässlich zu personalisieren. Ebenso ist die Auswahl eines klar definierten und geeigneten Anwendungsfalls entscheidend, damit Fine-Tuning seinen vollen Mehrwert entfalten kann.

**Abschließend bleibt somit festzuhalten:**

1. Der Einstieg in das Thema Fine-Tuning wird dank zahlreicher Tools zunehmend einfacher.
2. Fine-Tuning zeigt seine Stärken vor allem bei der Anpassung von Tonalität, Stil und Ausgabestruktur.
3. Schwächen bestehen weiterhin bei Halluzinationen und der Einbindung spezifischen Fachwissens, wo RAG meist vorzuziehen ist.
4. RAG, Prompt Engineering und Fine-Tuning stehen nicht im Widerspruch zueinander, sondern ergänzen sich als kombinierbare Ansätze, um optimale Ergebnisse zu erzielen.

# Quellen

[1] O. Ovadia, M. Brief, M. Mishaeli, O. Elisha, "Fine-Tuning or Retrieval? Comparing Knowledge Injection in LLMs", 2024.

[2] [LLM-Feinabstimmung: Best Practices und Anwendungsfälle](https://datasolut.com/feinabstimmung-llms-best-practice/%22%20%5Cl%20%22Was-sind-die-Vorteile-von-Feinabstimmung)

[3] [Azure AI Foundry](https://azure.microsoft.com/de-de/products/ai-foundry)

[4] [Amazon SageMaker AI](https://aws.amazon.com/de/sagemaker-ai/?gclid=CjwKCAiAneK8BhAVEiwAoy2HYa6iQkaiZl9rMFoFxVDUB89wMLo6yuHghKS3vRX8BiWYU0UyUzn_xBoCVpQQAvD_BwE&trk=3ea5c9d1-0497-4ab3-92e6-c583f43ac2f9&sc_channel=ps&ef_id=CjwKCAiAneK8BhAVEiwAoy2HYa6iQkaiZl9rMFoFxVDUB89wMLo6yuHghKS3vRX8BiWYU0UyUzn_xBoCVpQQAvD_BwE:G:s&s_kwcid=AL!4422!3!724218628448!e!!g!!amazon%20sagemaker%20ai!19571721771!171902905415)

[5] [Hugging Face - The AI community building the future](https://huggingface.co/)

[6] [Fine-tuning a model with the Trainer API](https://huggingface.co/learn/nlp-course/chapter3/3)

[7] [Unsloth AI - Open Source Fine-Tuning for LLMs](https://unsloth.ai/)

[8] [Google Colab - Fine-tune a language model](https://colab.research.google.com/github/huggingface/notebooks/blob/master/examples/language_modeling.ipynb)

[9] [Fine-Tuning versus RAG in Generative AI Applications Architecture](https://harsha-srivatsa.medium.com/fine-tuning-versus-rag-in-generative-ai-applications-architecture-d54ca6d2acb8)

[10] [To fine-tune or not to fine-tune](https://ai.meta.com/blog/when-to-fine-tune-llms-vs-other-techniques/)