TRAIBER.NRW
TRANSFORMATION DER AUTOMOTIVE INDUSTRIE IN DER BERGISCHEN REGION

FACTSHEET: LOKALE UND SELBSTGEHOSTETE LLMS

Lokale und selbstgehostete Large Language Modelle (LLMs) werden für Unternehmen zu einem strategischen Baustein. Sie ermöglichen den Einsatz generativer Künstlicher Intelligenz mit voller Kontrolle über sensible Daten, reduzierten Compliance- und Vendor-Lock-in-Risiken sowie besser planbaren Kosten. Dieses Factsheet gibt einen strukturierten Überblick darüber, welche technischen Rahmenbedingungen (Modellauswahl, Speicherformate, GPU-Ressourcen, Software-Architektur) für den lokalen Betrieb relevant sind und wann sich Investitionen in lokale LLMs lohnen können. Behandelt werden typische Einsatzszenarien wie vertrauliche Wissensbots/RAG im Intranet, Code-Assistenz oder Team-Setups mit begrenzter Nutzerzahl sowie passende Hardware-Szenarien von der Workstation bis zur Team-Appliance. Die Inhalte unterstützen Management und technisch affine Entscheidungsträger dabei, Chancen, Grenzen und Handlungsbedarf für die eigene Organisation realistisch einzuschätzen und auf dieser Basis Pilotprojekte, Budgets und Zuständigkeiten zu planen.

HINTERGRUND

Generative Künstliche Intelligenz für Texte und Bilder gilt in vielen Unternehmen als wichtiger Wertschöpfungshebel und soll Effizienz steigern, Routineaufgaben automatisieren und neue digitale Services ermöglichen [1]. Für hochwertige Ergebnisse benötigen Large Language Models (LLMs) passenden Kontext – meist interne Dokumente, vertrauliches Fachwissen und Prozesswissen aus dem Unternehmen. Genau hier entstehen jedoch Fragen nach Datenschutz und der Kontrolle über sensible Informationen. Lokale & selbstgehostete Large Language Modelle (LLMs) setzen an diesem Punkt an: Sie geben **volle Datenhoheit**, ermöglichen **Offline-Betrieb** und verhindern, dass sensible Informationen auf externe Clouds abfließen oder Compliance-Verstöße begangen werden. Gleichzeitig können sie bei planbaren Workloads die **laufenden Kosten senken**, reduzieren das Vendor-Lock-In Risiko bei einem Cloud-Anbieter, lassen sich durch Retrieval-Augmented Generation (RAG) und Fine-Tuning gezielt an Unternehmensbedürfnisse anpassen und liefern **geringe Latenzen** direkt am Entstehungsort der Daten. Typische Einsatzfelder sind vertrauliche Wissensbots bzw. RAG-Lösungen im Intranet, Code-Assistenz auf Entwickler-Workstations oder Team-Setups mit einer überschaubaren Zahl von Nutzenden. Damit werden lokale LLMs insbesondere für mittelständische Unternehmen interessant, die Wert auf Datensouveränität, Planbarkeit und regulatorische Sicherheit legen.

Dieses Fact Sheet richtet sich an Management sowie technisch affine Entscheidungsträger und Praktiker, die bewerten möchten, wann sich Investitionen in lokale LLMs lohnen, und bietet Führungskräften auf höherer Managementebene einen strukturierten Überblick. Im Mittelpunkt stehen Rahmenbedingungen, unter denen selbst- gehostete LLMs sinnvoll sind, die **wichtigsten technischen Grundbegriffe** für den Betrieb auf Workstations oder Team-Appliances (Modellgrößen, Kontextfenster, Speicherbedarf, GPU-Ressourcen) sowie die **typischen Bausteine** im Software-Stack (Backend, Frontend/GUI, RAG, Fine-Tuning, Wissensdatenbanken). Dargestellt werden typische Einsatzszenarien und die dafür erforderlichen Modell- und GPU-Ressourcen – von der einzelnen Workstation bis zur kleinen Team-Appliance. Der Fokus liegt nicht auf einer detaillierten Betriebsanleitung, sondern auf einem gemeinsamen Vokabular, das Gespräche zwischen Management, Fachbereichen und Technik erleichtert. Für das Management werden Anhaltspunkte zur Strukturierung von Investitionsentscheidungen gegeben (z. B. grobe Abschätzung von Hardware-Budgets und Projektumfang), während die technische Zielgruppe eine „Landkarte“ erhält, um existierende oder geplante LLM-Initiativen im Unternehmen einzuordnen. Die dargestellten Inhalte eignen sich als Gesprächsgrundlage in Strategie-Workshops, frühen Scoping-Phasen oder beim Austausch mit externen Partnern.

MODELAUSWAHL

Für die Auswahl des richtigen lokal gehosteten Modells gibt es einige zentrale Begriffe, die unerlässlich sind um sich mit dem Thema zu befassen. In der folgenden Liste sind die wichtigsten auf Modellebene beschrieben:

- **Parameteranzahl:** Anzahl der Gewichte im Modell in Milliarden (engl. *billion*) bzw. als Einheit **[B]**
- **Token:** Basiseinheit für Text-Kodierung, um Inhalte effizient in LLMs verarbeiten zu können. 1 Token entspricht ungefähr 3–4 Zeichen im Text
- **Präzision:** Rechengenauigkeit/ Datentyp der Gewichte z. B. FP16 (16 Bit), INT8 (8 Bit), INT4 (4 Bit)
- **Quantisierung:** Reduzierung der Präzision zur Senkung von Speicher- und Rechenbedarf bei Qualitätsverlust (z. B. von FP16 zu INT4)
- **Modelgröße:** Gesamter Speicherbedarf des Modells im RAM/VRAM, korreliert mit Parameteranzahl

Etwas vereinfacht zusammengefasst: *Parameteranzahl* und *Modelgröße* bestimmen grob, wie leistungsfähig ein Modell ist und wie viel Hardware es benötigt. Das *Kontextfenster* legt fest, wie lange Dokumente oder Chatverläufe ein Modell sinnvoll verarbeiten kann. *Quantisierung* und *Präzision* sind Stellschrauben, mit denen man ein Modell so anpasst, dass es noch ausreichend gute Qualität liefert, aber in das vorhandene GPU-Budget passt.

Einordnung nach Paramtergröße

Die Parametergröße ist typischerweise ein einfacher Indikator über die Antwortqualität und Bedarf an GPU-Speicher eines spezifischen Modells und damit die laufenden Inferenzkosten. Die folgende Unterteilung bietet eine grobe Orientierung, welche Größenordnung für typische Unternehmensszenarien realistisch ist, von einfachen Prototypen bis hin zu breit genutzten Assistenzsystemen:

- **Small (~7-16B):** Eingeschränkte Allround-Qualität, aber oft ausreichend für einfache Assistenten, interne Prototypen oder klar abgegrenzte Spezialaufgaben
- **Medium (~27-33B):** Solide Allround-Nutzung mit besserer Antwortqualität und mehr Kontexttiefe; geeignet für viele Wissensbots und Office-Unterstützung
- **Large (~70-73B):** Hohe Allround-Qualität, geeignet für anspruchsvollere Aufgaben, komplexere Dialoge und breitere Nutzung im Unternehmen
- **XL/Ultra (>200B):** Sehr hohe Qualität mit Ergebnissen vergleichbar zu vielen Cloud-Lösungen; nur mit Cluster-Hardware wirtschaftlich zu betreiben

Vereinfacht gesagt: Je größer das Modell, desto höher sind typischerweise Qualität und Hardwareanforderungen. Für viele mittelständische Szenarien reichen Small- oder Medium-Modelle mit guter Anpassung und Architektur bereits aus. Eine weiterführende Entscheidungshilfe für die anwendungsspezifische Modellauswahl ist in [3] zu finden.

Weiterführende Begriffe

Neben der Kategorisierung über die Parameteranzahl können Modelle auch zusätzliche Fähigkeiten haben, die sie gegenüber anderen Modellen derselben Größe hervorheben. Solche Eigenschaften beeinflussen zum Beispiel, wie gut ein Modell komplexe Aufgaben strukturiert bearbeiten kann oder wie effizient vorhandene Rechenressourcen ausgenutzt werden. Die folgenden Begriffe markieren wichtige Weiterentwicklungen, die bei der Bewertung moderner, lokal einsetzbarer LLMs berücksichtigt werden sollten:

- **Reasoning:** Fähigkeit eines Modells, Schlussfolgerungen zu ziehen und mehrstufige Denkprozesse auszuführen, um komplexe Aufgaben logisch, planvoll und nachvollziehbar zu lösen
- **Mixture-Of-Experts-Modelle (MoE):** Aktiviert pro Eingabe nur relevante spezialisierte Teilnetze (Experten), wodurch bei gleicher Rechenleistung eine höhere Kapazität und Spezialisierung möglich ist

Für die Praxis bedeutet dies: *Reasoning* Modelle sind insbesondere dann interessant, wenn komplexe Entscheidungen, Analysen oder Planungsaufgaben unterstützt werden sollen. *Mixture-of-Experts*-Modelle eignen sich vor allem für Szenarien mit vielen unterschiedlichen Aufgabenbereichen, in denen eine hohe Kapazität bei vertretbarem Rechenaufwand benötigt wird.

Speicherformate von Modellen

Aktuelle Standard-Speicherformate für lokale LLMs sind **GGUF** (GPT-Generated Unified Format) und **safetensors**, die für schnelles und sicheres Laden optimiert sind. Die meisten Umgebungen können GGUF und safetensors Modelle laden, aber manche Umgebungen wie z.B. llama.cpp können nur GGUF laden. Von bin- und pth-Formaten ist **abzuraten**, da beim Laden beliebiger Python-Code ausgeführt werden kann, was ein Sicherheitsrisiko darstellt [4]. Für das **Management** reicht hier oft die Leitfrage an die technischen Verantwortlichen: *„Nutzen wir für lokale Modelle etablierte, sichere Formate wie GGUF oder safetensors?“*. Eine positive Antwort ist ein guter Indikator dafür, dass beim Modellhandling Sicherheitsaspekte berücksichtigt wurden.

HARDWARE & USE-CASES

Einsatzszenarien und Hardwareempfehlungen

Für den lokalen Einsatz von LLMs hängt die geeignete Hardware stark von der Nutzeranzahl, der erwarteten gleichzeitigen Nutzung und der Modellgröße ab. Einzelanwender kommen meist mit einer Workstation aus Teams oder Abteilungen benötigen bereits dedizierte Serverstrukturen. Für zentrale Services oder besonders große Modelle ist ein Clusterbetrieb notwendig, was hier aber nicht abgedeckt werden kann. Die folgenden Angaben verstehen sich als **Orientierung** für typische Szenarien und sind keine festen Vorgaben. Sie helfen dabei, eine realistische Größenordnung für geplante Pilotprojekte und spätere Erweiterungen abzuschätzen, bevor konkrete Angebote eingeholt werden.

Workstation (z. B. 1-5 Nutzende)

- **Use Case:** Für eine geringe Zahl gleichzeitiger Abfragen geeignet, z. B. für Prototyping oder persönliche Assistenzanwendungen wie Textgenerierung, Code- oder Office-Unterstützung
- **GPU-Leistung:** 1-2 GPUs mit je 24–48 GB VRAM
- **Empfehlung:** 1 GPU als Basis; 2 GPUs, wenn Modelle ~33–70B genutzt, längere Kontexte gefahren oder multimodal verarbeitet werden (LMM)

Kleiner Server / Team-Appliance (z. B. 5–25 Nutzende)

- **Use Case:** Geeignet für Teams oder Abteilungen, die LLMs regelmäßig und parallel verwenden für beispielsweise Zugriff auf interne Wissensbestände, z. B. Unternehmensdokumente oder Projektdaten oder Mehrbenutzerbetrieb von Text- oder multimodalen Modellen
- **GPU-Leistung:** 48–80 GB VRAM pro GPU
- **Empfehlung:** 2 GPUs als Minimum für stabilen Mehrbenutzerbetrieb (max. 5); 4 GPUs als Standard für 10–25 gleichzeitige Sessions oder 70B in INT4 mit Reserven; 6–8 GPUs für 15–25 Nutzende

Für erste Pilotprojekte reicht meist eine gut ausgestattete Workstation, für Teamweite Nutzung ist ein kleiner Server mit mehreren GPUs erforderlich. Clusterlösungen sind erst bei Unternehmensweiten, stark skalierenden Anwendungen oder sehr großen Modellen sinnvoll.

Faustregeln für VRAM-Bedarf

Das gesamte Model und das Kontextfenster passen idealerweise in den Arbeitsspeicher (VRAM) der verfügbaren GPUs passen [3]. Es lassen sich zwar auch Teile des Models oder Kontext auf die CPU und RAM auslagern, aber das führt zu einer längeren Antwortzeit. Um den VRAM-Bedarf zu berechnen kann man folgende Faustregeln verwenden:

- Eine 4-Bit-Quantisierung benötigt etwa 0,5 GB pro 1 B Parameter, 8-Bit etwa 1 GB, FP16 rund 2 GB
- Pro 1k Tokens Kontext kommen 0,2–0,6 GB dazu

Damit benötigen 7 B-Modelle mit einem Kontextpuffer rund 8–12 GB VRAM, 13B-Modelle 12–16 GB, 33 B-Modelle 24–32 GB und 70B-Modelle etwa 48–80 GB.

BACKEND-SERVER & FRONTEND-GUIS

Selbstgehostete LLM-Systeme sind häufig architektonisch in einen Backend-Server und ein Frontend unterteilt. Auf dem Backend-Server werden die Modelle geladen und ausgeführt, während Nutzerende im Frontend, meist über ein *Graphical User Interface* (GUI) ihre Anfragen und Prompts eingeben. Daraus ergibt sich dieser Aufbau:

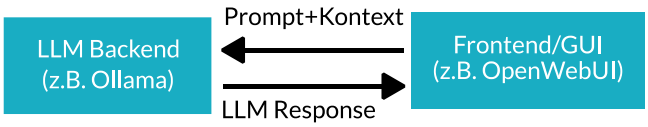

Abbildung 1: Vereinfachter Aufbau eines LLM-Systems

Die Kommunikation zwischen Back- und Frontend erfolgt typischerweise über eine HTTP(S)-Schnittstelle, deren Format an den Standard der OpenAI-API angelehnt ist. Dadurch sind Back- und Frontend weitestgehend unabhängig voneinander wählbar. Für Unternehmen hat diese Trennung einen **praktischen Vorteil**: Das Backend kann als zentraler, sicher betriebener LLM-Service im Rechenzentrum laufen, während verschiedene Frontends (z. B. Weboberflächen, Fachanwendungen oder Integrationen in bestehende Systeme) unabhängig davon entwickelt und ausgetauscht werden können. Dadurch bleiben Investitionen in die Infrastruktur langfristig nutzbar, auch wenn sich Oberflächen oder Tools ändern. Eine Auswahl für mögliche Backend-Server und Frontend-GUIs sind hier aufgeführt:

Backend-Server:

- **Ollama:** Lokaler LLM-Server mit einfacher Installation und Model-Management. Ideal für Workstations; bietet OpenAI-kompatible API und gute GUI- Integration
- **Llama.cpp:** Leichtgewichtig für CPU- und GPU-Systeme (GGUF-Formate). Optimiert für geringe Ressourcen und breite Plattformunterstützung
- **vLLM:** Für Multi-GPU- und Cluster-Umgebungen. Bietet effizientes Speicher- und Batch-Management und eignet sich für hohe Anfragevolumina

Frontend/GUI:

- **OpenWebUI:** Lokaler LLM-Server mit einfacher Installation und Model-Management. Ideal für Workstations; bietet OpenAI-kompatible API und gute Integration mit GUIs
- **text-generation-webui:** Leichtgewichtig für CPU- und GPU-Systeme (GGUF-Formate). Optimiert für geringe Ressourcen und breite Plattformunterstützung
- **LibreChat:** Für Multi-GPU- und Cluster-Umgebungen. Bietet effizientes Speicher- und Batch-Management und eignet sich für hohe Anfragevolumina

Für Single-User Workstations können All-In-One Lösungen wie z. B. GPT4All Desktop, LM-Studio, Ollama App (Ollama mit integrierter GUI) eine gute Option sein, da sie Backend und Frontend in einer Anwendung kombinieren.

EIGENE MODELLE & WISSEN

Unternehmen profitieren besonders dann von lokalen LLMs, wenn diese auf eigenes Fachwissen und unternehmensspezifische Daten abgestimmt sind. Zwei zentrale Ansätze ermöglichen dies: das Fine-Tuning eines bestehenden Modells zur Anpassung an domänenspezifische Ausdrucksweisen und Regeln, sowie die Erweiterung des Wissenskontexts durch externe Wissensdatenbanken (Retrieval-Augmented Generation, RAG). Beide Methoden erhöhen die Relevanz und Zuverlässigkeit der Antworten, unterscheiden sich jedoch in Aufwand, Flexibilität und Wartbarkeit.

Fine-Tuning

Beim Fine-Tuning wird ein bestehendes Modell mit zusätzlichen, meist domänenspezifischen Beispielen nachtrainiert. Dadurch lernt es, firmenspezifische Terminologien, Schreibstile oder Entscheidungslogiken besser anzuwenden. Leichtgewichtige Verfahren wie LoRA (Low-Rank Adaptation) oder QLoRA reduzieren dabei den Ressourcenbedarf erheblich, da nur ein kleiner Teil der Modellgewichte angepasst wird. Fine-Tuning eignet sich insbesondere, wenn:

- Ein konsistenter Schreibstil erlernt werden soll,
- formale Regeln wie Berichtsstandards gelten,
- proprietäre Terminologien zuverlässig abgebildet werden müssen,
- oder konstantes Domänen-spezifisches Wissen bekannt sein muss.

Für stark veränderliche Wissensdomänen ist Fine-Tuning hingegen weniger geeignet, da jedes neue Wissen ein erneutes Training erfordert. Hier bietet sich stattdessen eine RAG-basierte Architektur an.

Wissensdatenbanken

Anstatt das Modell selbst zu verändern, erweitern Wissensdatenbanken den Kontext der Antwortgenerierung. Diese Methode, bekannt als Retrieval-Augmented Generation (RAG), kombiniert ein lokales LLM mit einer Suchkomponente, die aus einer externen Datenquelle (z. B Unternehmensdokumente, Datenbanken oder Projektreports) relevante Textpassagen abruft [5]. Die abgerufenen Inhalte werden als zusätzlicher Kontext in den Prompt integriert, sodass das Modell auf aktuelle und geprüfte Informationen zugreift, ohne neu trainiert zu werden. Eine hochwertige RAG-Implementierung zeichnet sich aus durch:

- präzise Dokumentaufbereitung mit Chunking und hoher Embedding-Qualität,
- semantische Suche mit Vektordatenbanken,
- und eine strukturierte Prompt-Zusammenführung.

RAG-Systeme sind besonders wartungsfreundlich und bieten robuste, nachvollziehbare Antworten – ideal für interne Wissensbots oder technische Supportsysteme. Eine Übersicht eines solchen Systems ist in Abbildung 2 dargestellt.

Übersicht für Firmenwissen

Die Entscheidung, welches Verfahren im eigenen Unternehmenskontext am sinnvollsten ist, hängt von mehreren organisationsspezifischen Faktoren ab, unter anderem vom technischen Know-how im Bereich LLMs, der Qualität und Struktur der verfügbaren Unternehmensdaten sowie von der lokal verfügbaren Rechenleistung für Experimente und Betrieb. Aus Entscheidungssicht lässt sich die Wahl jedoch häufig auf drei Grundvarianten verdichten und schafft damit eine Grundlage, ob eher in Fine-Tuning, RAG oder eine kombinierte Vorgehensweise investiert werden sollte:

- **Fine-Tuning**: Wenn ein konsistenter Schreibstil, feste Regeln und stabile, wenig veränderliche Domäneninhalte im Vordergrund stehen (z. B. Berichtsstandards, Fachterminologie)
- **RAG**: Wenn sich Inhalte häufig ändern (z. B. Projekte, Produkte, Richtlinien) und aktuelle, prüfbare Informationen aus Dokumenten benötigt werden
- **Kombination**: wenn einerseits ein bestimmter Sprachstil eingeprägt werden soll, andererseits auf große, sich verändernde Wissensbestände zugegriffen wird

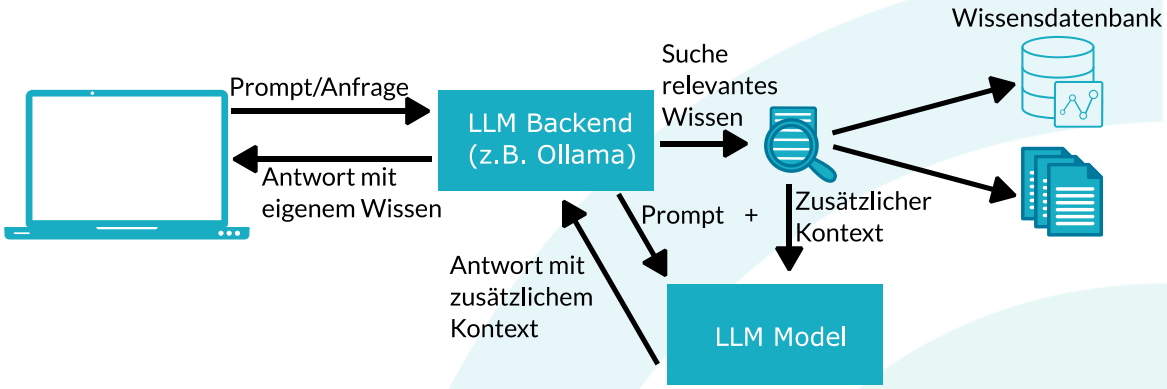

Abbildung 2: Der Aufbau und Ablauf eines RAG-Systems, um einen Prompt mit Unternehmensspezifischen Wissen zu ergänzen.

ZUSAMMENFASSUNG

Warum lokale LLMs für Unternehmen relevant sind?

Lokale bzw. selbstgehostete Large Language Modelle (LLMs) ermöglichen den Einsatz generativer Künstlicher Intelligenz, ohne dass sensible Unternehmensdaten an externe Cloud-Anbieter übertragen werden müssen. Das erhöht die Datenhoheit, reduziert Compliance-Risiken und erleichtert die Einhaltung interner Richtlinien sowie regulatorischer Vorgaben. Gleichzeitig können bei planbaren Workloads die laufenden Kosten gesenkt und Antwortzeiten reduziert werden. Insbesondere wenn das Modell nahe an den Daten und den Nutzenden betrieben wird.

Zentrale technische Kenngrößen

Parameteranzahl und *Modelgröße* geben grob an, wie leistungsfähig ein Modell ist und wie viel Hardware dafür benötigt wird. Das *Kontextfenster* bestimmt, wie viele Dokumente oder wie lange Chatverläufe sinnvoll verarbeitet werden können. Durch *Quantisierung* lassen sich Modelle so verkleinern, dass sie auf vorhandene GPU-Ressourcen passen, ohne dass die Qualität für viele Anwendungsfälle unbrauchbar wird.

Orientierung bei Hardware und Einsatzszenarien

Für Einzelpersonen oder kleine Teams reicht häufig eine leistungsfähige Workstation mit ein bis zwei GPUs, etwa für interne Assistenten oder erste Pilotprojekte. Für ganze Abteilungen oder zentrale Dienste mit überschaubarer Nutzenden Anzahl werden kleine Server- bzw.\ Team-Appliances mit mehreren GPUs relevant, um mehrere Nutzende und Anwendungen stabil zu bedienen. Clusterlösungen mit vielen GPUs sind in der Regel erst dann notwendig, wenn sehr große Modelle oder unternehmensweite, stark skalierende Anwendungen geplant sind.

Architektur: Trennung von Backend und Frontend

Üblicherweise sind LLM-Systeme unterteilt in einen zentralen Backend-Server, auf dem die LLMs betrieben werden, und ein oder mehrere Frontends/GUI-Lösungen, über die ein Nutzender auf die Modelle zugreifen. Die technische Komplexität und Sicherheit werden weitestgehend im Backend gebündelt, während Oberflächen für unterschiedliche Zielgruppen (z. B. Fachabteilungen, IT, Entwicklung) flexibel gestaltet und bei Bedarf ausgetauscht werden können. Dadurch bleiben Investitionen in die Infrastruktur langfristig nutzbar, auch wenn sich Anforderungen oder Tools im Laufe der Zeit ändern.

Eigene Inhalte einbinden: Fine-Tuning, RAG oder Kombination

Damit Unternehmenswissen in die Arbeit mit lokalen LLMs integriert werden kann, gibt es zwei grundlegende Ansätze. *Fine-Tuning* eignet sich, um Modelle an domänenspezifische Sprache, Regeln und Schreibstile anzupassen, die sich nur selten ändern (z. B. Berichtsstandards, Fachterminologie). *Retrieval-Augmented Generation (RAG)* nutzt dagegen externe Wissensdatenbanken und Dokumente, um aktuelle, prüfbare Informationen zur Laufzeit in Antworten einzubinden, ohne das Modell selbst neu trainieren zu müssen. Für viele Unternehmen ist eine *Kombination* beider Ansätze sinnvoll: das Modell lernt den passenden „Ton“, während Inhalte aus einer gepflegten Wissensbasis stammen.

Nutzen des Factsheets für Management und Technik

Insgesamt bietet das Factsheet eine gemeinsame Grundlage für Gespräche zwischen Geschäftsführung, Bereichsleitung und technischen Teams. Es liefert keinen vollständigen Implementierungsleitfaden, sondern eine strukturierte Übersicht über Chancen, Grenzen und Rahmenbedingungen beim Einsatz lokaler LLMs. Entscheidungsträger erhalten damit eine Orientierung, in welchen Fällen lokale LLMs strategisch interessant sind und welche Größenordnungen an Hardware realistisch sind, bevor konkrete Pilotprojekte gestartet werden.

QUELLEN:

[1] KPMG, Generative KI in der deutschen Wirtschaft 2025, 2025
[2] Simon, J., Charlaix, E., Zafrir, O., Margulis, I., Boudoukh, G., Wasserblat, M., A Chatbot on your Laptop: Phi-2 on Intel Meteor Lake, Hugging Face, Online, 14.11.2025
[3] Vila, Daniel, How to Choose the Best Open Source LLM for Your Project in 2025, Hugging Face, Online, 14.11.2025
[4] Hugging Face, Pickle Scanning, Hugging Face Documentation, Online, 14.11.2025
[5] Yu, H., Gan, A., Zhang, K., Tong, S., Liu, Q., Liu, Z. (2025). Evaluation of Retrieval-Augmented Generation: A Survey. In: Zhu, W., et al. Big Data. BigData 2024. Communications in Computer and Information Science, vol 2301. Springer, Singapore.

HERAUSGEBER

TRAIBER.NRW

GESCHÄFTSSTELLE TRAIBER.NRW

c/o Bergische Universität Wuppertal
TMDT - Institute for Technologies and Management of Digital Transformation

Lise-Meitner-Str. 27, 42119 Wuppertal
Telefon: 0202 439 1164
E-Mail: koordination@traiber.nrw
www.traiber.nrw

INHALTLICHE VERANTWORTUNG

STEPHAN SANDFUCHS
DIAKO FAROOGHI
JÖRG FROCHTE
Hochschule Bochum
AKIS – Interdisziplinäres Institut für Angewandte KI und Data Science Ruhr

Gefördert durch:
Bundesministerium für Wirtschaft und Energie
aufgrund eines Beschlusses des Deutschen Bundestages