EduLLama

Feinabstimmung und Distillation von LLama zu Bildungszwecken

Fabian Banovic[1], *Lukas* Bruckner[1], *Joschua* Jakubek[1],*Marcus* Wirth[2]

[1]*Eviden* [2]*Schaeffler*

Abstract. LLM's geben i.d.R. wenig präzises Domänenwissen wieder, oder gar falsche Informationen. Das macht sie weniger nützlich in Domänen, in denen Spezialwissen und präzise Antworten erforderlich sind. In diesem Papier untersuchen wir, ob mithilfe von Finetuning effektiv eine ausreichende Genauigkeit für Domänenwissen erzielt werden kann und ob es grundsätzlich möglich ist, anhand eines Finetuning-Lehrermodells ohne weiteren Grunddatensatz eine Distillation auf ein Schülermodell durchzuführen. Diese Frage wird mit einer vergleichenden Evaluierung des Llama-7b Basismodells und einer Finetuning Version dessen, sowie einem explorativen Versuch der Distillation untersucht. Im direkten Vergleich konnte das Finetuning Modell, anhand von GPT-4, das Basismodell in 56% der Antworten schlagen, gemessen an der Kosinusähnlichkeit jedoch nur in 34%. Im Bereich der Distillation des Finetuning Modells konnte ein Weg gefunden werden, mit dem ein Schülermodell nur mit den Antworten des Lehrermodells auf einen Fragenkatalog trainiert werden kann. Die Ergebnisse deuten darauf hin, dass mit mehr Daten und einer besseren Datenvorbereitung das Finetuning auf eine Domäne in ausreichender Genauigkeit möglich ist. Des weiteren ist mit weiterem Entwicklungsaufwand des Distillation-Trainers und höherer Rechenkapazität eine zielführender Distillationprozess zu erwarten.

1 Motivation

Wie in den letzten Monaten deutlich zu erkennen ist, gewinnen Chatbots, wie beispielsweise ChatGPT, immer mehr an Popularität. Solche Chatbots sind meist sehr breit aufgestellt und können im Wesentlichen auf alle Fragen antworten. Kommt es nun zu fachspezifischen Angelegenheiten, so kann es sein, dass einige Antworten nicht immer den Erwartungen des Nutzers entspricht. An dieser Stelle setzt die nachfolgende Arbeit an. Sie beschäftigt sich mit dem Thema, wie diese Modell mit fachspezifischen Wissen ausgestattet werden können, um die eben dargestellte Problematik lösen zu können. Am Ende der Arbeit soll die Frage beantwortet werden, welche Möglichkeiten es gibt, ein Large Language Model für einen spezifischen Kontext anzupassen und wie dieses letztendlich performt. Dabei liegt der Fokus auf der Art und Weise wie das Finetuning stattfindet. Um das Ergebnis der Arbeit bewerten zu können, soll zum Schluss deutlich gemacht werden, wie die verwendeten Modelle verglichen und bewertet wurden.

Des Weiteren werden LLMs zunehmend größer, was für viele Anwendungsfälle nicht unbedingt benötigt werden würde. Eine potenzielle Ausführung auf Edge Devices liegt dadurch ebenfalls zunehmend außer Reichweite. Durch den Prozess der sogenannten *Distillation* kann das Wissen solcher Modelle zu gewissen Teilen in kleinere Modelle transferiert werden, wodurch man spezialisierte leichtgewichtige Modelle für Tasks einsetzen kann. In dieser Arbeit soll der Ansatz für eine besonders ressourcenschonende und datensparende *Distillation* zum Trainieren auf eine Domäne spezialisierter Modelle erarbeitet werden.

2 Datensatz

2.1 Herkunft und Extraktion der Daten

Die für das Projekt verwendeten Daten stammen aus Vorlesungsfolien im PDF-Format. Um die Dateiinhalte für das Finetuning vorzubereiten wurden einige Schritte durchgeführt. Zunächst wurden alle Dateien mithilfe von PyMuPDF, einem Python Paket spezialisiert auf die Datenextraktion aus PDF-Dateien[1], verarbeitet. Eine daraus resultierende Liste, enthält alle Seitenzahlen die Bilddateien enthalten. Diese Liste kann, sofern Bilder, wie beispielsweise Logos oder ähnliche, nicht extrahiert werden sollen, angepasst werden. Die jeweiligen Dateien wurden daraufhin, mithilfe dieser Liste, an den entsprechenden Stellen aufgeteilt. Resultierend daraus, konnten nun die dazwischenliegenden Texte verarbeitet werden. Dafür wurde auf der einen Seite Py2PDF, ein Tool für die Textextraktion [2], für Dateien verwendet, welche direkt ausgelesen werden konnten. Auf der anderen Seite wurde Tesseract, ein Tool das Object Character Recognition (OCR) einsetzt, um Text aus PDFs zu lesen [3], für Dateien verwendet, in denen der Text eingescannt wurde und damit eine direkte Extraktion nicht möglich macht. Aufgrund dessen, dass Bilder und Grafiken, für uns Menschen, das Verständnis von Themen visualisieren oder gar erweitern, ist es wichtig diese mit in den Datensatz aufzunehmen. Damit die Bilder, welche vorher identifiziert worden sind, genutzt werden können, wurden diese an die API von GPT-4 geschickt, um eine Beschreibung des Inhaltes in Textform zu erhalten. Da einige Inhalte auf deutsch waren und Llama 2 besser mit englischem Text umgehen kann, wurden diese mithilfe der GPT-API übersetzt, wofür die Genehmigung der jeweiligen Ersteller eingeholt wurde. Endergebnis waren einzelne Textdateien der ehemaligen PDF-Dokumente.

2.2 Vorverarbeitung

Wie eben erwähnt, sind die Informationen zum derzeitigen Zeitpunkt in Textdateien abgespeichert. Um die Daten als Input für die Modelle verwenden zu können, wurden sie zunächst eingelesen. Das Ziel ist es die jeweiligen Vorlesungsinhalte, der einzelnen Dateien, so zu preparieren, dass der daraus resultierende Datensatz aus den Inhalten von jeweils einer Vorlesungsfolie entspricht. Damit dies umgesetzt werden konnte, wurden die Seitenzahlen, sofern diese vorhanden waren, benutzt. In anderen Dateien, welche weder Seitenzahlen noch son-

stige wiederkehrende Merkmale enthielten, sondern nur aus Auflistungen, wurde sich dazu entschieden, einzelne Textblöcke als Input zu nehmen. Daraus folgte, dass es einige wenige Inputs gab, die mehrere Tausende Wörter enthielten. Um dieses Problem zu lösen, wurde sich an der durchschnittlichen Wortanzahl aller Einträge orientiert, um die zu großen Texte in mehrere kleine aufzuteilen. Das Ergebnis all dieser Operationen war eine Liste mit allen Einträgen, welche letztendlich genutzt wurde, um den finalen Datensatz zu erstellen.

2.3 Effizientes Finetuning mit QLoRA

Diese Projekt wurde mithilfe von kostenlos zur Verfügung stehenden Ressourcen auf der Platform "Kaggle" durchgeführt. Die Hardware, die zur Verfügung gestellt wurde, ist eine CPU mit 2 Intel Xeon 2.00GHz vCPU's, 16 GB RAM und einer NVIDIA P100 GPU mit 16GB VRAM. Da diese Ressourcen nicht ausreichend sind für ein volles Finetuning von Llama 2 7B, wurde für das QLoRA Finetuning verwendet.

Das Finetuning mit QLoRA basiert auf der Idee von **LoRA (Low-Rank Adaptation)**. Mithilfe von LoRA kann das Finetuning von LLM's (Large Language Models) entscheidend beschleunigt werden. Die vorhandenen Methoden für das Finetuning ohne LoRA haben den Nachteil, dass Modelle entweder erweitert, oder als Ganzes erneut trainiert werden müssen. Durch LoRA werden bei vergleichbarer Leistung die anpassbaren Modellparameter, und somit der Rechenaufwand als auch die Anforderungen an den Grafikspeicher drastisch gesenkt. Erzielt wird dieser Effekt durch das "Einfrieren" der vortrainierten Parameter und die Einführung neuer Parameter für das Finetuning. Dieses Vorgehen basiert auf der Annahme, dass das Finetuning einer deutlich niedrigeren intrinsische Dimension unterliegt, verglichen mit dem Pretraining. Die Matrizen der Modellgewichte eines Layers im LLM, besitzen aufgrund ihrer Informationsdichte einen hohen oder vollen Rang. Im Prozess des Finetunings werden jene Matrizen durch Addition an das neue Ziel angepasst. (1) Die Gewichtsmatrix W_0 aus dem Pretraining wird in die Rechnung einbezogen, bleibt aber unverändert, oder auch "eingefroren". Es wird nur die Matrix ΔW im Finetuning angepasst. Diese besitzt aber ebenfalls einen hohen oder vollen Rang und kann unter der Annahme der niedrigen intrinsischen Dimension weiter vereinfacht werden. (2) Die positive Ganzzahl r definiert als Hyperparameter den Rang der Matrizen von A und B. Während des Finetunings werden nur die Parameter dieser Matrizen angepasst, da diese Implizit ΔW repräsentieren. Mithilfe von r lässt sich die Komplexität, und damit Genauigkeit und Rechenaufwand, je nach Kontext steuern. LoRA führt außerdem den Faktor $\frac{\alpha}{r}$ ein, welcher ähnlich der Learning Rate die Anpassung während des Finetunings dämpft. Nach Abschluss des Finetunings wurden die ursprünglichen Parameter nicht verändert, sondern ein sogenannter angepasster Adapter an jeden Layer angehängt. In einem Forward-Pass wird der Eingangsvektor sowohl mit den Parametern des Pretrainings als auch mit dem Adapter multipliziert und beide Ergebnisse als Output aufsummiert. (3) LoRA kann auf jeden beliebigen hidden layer in jedem beliebigen Neuronalen Netzwerk im Finetuning angewandt werden. Dadurch wird die Anzahl der trainierbaren Parameter im Finetuning drastisch reduziert und die Möglichkeit geschaffen, austauschbare Adapter eines LLM's für verschiedene Aufgaben zu erhalten. [4]

Auf Basis des vorangegangenen Konzepts werden durch **QLoRA (Quantized Low-Rank Adaptation)** weitere entscheidende Fortschritte für das Finetuning veröffentlicht, welche dieses auf der uns zur Verfügung stehenden Hardware ermöglichen. Das QLoRA Finetuning führt zusätzlich zu LoRA drei weitere Schritte ein: 1. den Datentyp 4-Bit-NormalFloat, 2. die doppelte Quantisierung und 3. die Auslagerung des Optimizers. Der 4-Bit-NormalFloat ermöglicht, basierend auf der Annahme einer Normalverteilung der Parameterwerte zwischen -1 und 1 mit dem Mittelwert 0, den Informationsverlust bei der Quantisierung von Parametern auf jenen 4-Bit Datentyp zu minimieren. Dadurch wird eine effektiv verlustfreie Dequantisierung zu ermöglicht. Der 4-Bit-NormalFloat minimiert den Informationsverlust, indem die Quantisierungskonstanten in Form von Quantilen an die Normalverteilung der Parameterwerte angepasst werden. Der benötigte Speicher eines LLM's setzt sich aus der Anzahl der Parameter multipliziert mit der Größe des Datentyps zusammen. Kann der Datentyp verlustfrei, beispielsweise von 32-Bit-Float zu 4-Bit-NormalFloat quantisiert werden, so wird der benötigte Speicher immens gesenkt. Die doppelte Quantisierung verringert den Fußabdruck des 4-Bit-NormalFloat im Speicher nochmals. Die aus der Quantisierung zu diesem Datentyp entstandenen Quantisierungskonstanten benötigen ebenfalls Speicherplatz. Um deren Speicherverbrauch weiter zu verringern, werden sie erneut einer Quantisierung unterzogen. Dieser Schritt unterliegt ebenfalls keinem nennenswertem Informationsverlust. Die Parameter der LoRA Matrizen verbleiben immer in einem Rechendatentyp, zum Beispiel 16-Bit-BrainFloat, und die Pretraining Parameter grundsätzlich im Speicherdatentyp, dem 4-Bit-NormalFloat. Nur wenn die jeweiligen Pretraining Parameter in der Backpropagation benötigt werden, müssen sie für diesen Schritt in den Rechendatentyp Dequantisiert werden, damit die Matrixmultiplikation mit höherer Präzision durchgeführt werden kann. Es hat sich gezeigt, dass der dabei auftretende Quantisierungsverlust durch das Finetuning implizit ausgeglichen wird. Zuletzt können mit QLoRA, auftretende Spitzen in der Speicherallokierung durch den Optimizer ausgelagert werden. Hierfür wird eine Art geteilter Speicher zwischen der GPU und der CPU genutzt, in welchem der Zustand des Optimizers gespeichert wird, falls der GPU-Speicher überallokiert wird. Diese Situation kann durch große Sprünge in der Gradientenberechnung auftreten, beispielsweise wenn ein verhältnismäßig umfangreicher Batch den Backward-Pass durchläuft. [5]

Für das Finetuning von Llama ist die Vorlage der QLoRa Autoren verwendet worden. Hierbei waren einige Aspekte anzupassen, hauptsächlich waren alle Sublayer in der Llamastruktur zu definieren, an die die LoRa Matrizen angehängt werden sollen. Ein Problem stellte zu Beginn die

fehlende Konvergenz des Losses dar. Dies ist durch Erhöhung der Inputlänge auf 4 Folien erreicht worden. (4)

3 Distillation

3.1 Related work

Der konkrete Begriff der *Distillation* ist 2015 von Hinton et. al. [6] aufgebracht worden. Grundlegend wird die *Distillation* hier als Prozess definiert, bei welchem das in einem gegebenen Modell mit relativ vielen Parametern enthaltene Wissen in ein deutlich kleineres Modell übertragen wird. Dies erfolgt über die Verwendung von *soft targets* für das Training des kleineren Modells, welche sich aus der vorhergesagten Outputverteilung des großen Modells ergeben. Es wird also stets der gleiche Input, beispielsweise eine Zahl aus MNIST, beiden Modellen übergeben und die Vorhersage des großen Modells dient als Zielwert für die Berechnung des Loss für das kleine Modell. Im Gegensatz zu den im normalen Training verwendeten *hard targets*, welche zum Beispiel nur 1 sind für die richtige Klasse und sonst 0, spiegeln die *soft targets* zudem auch die Abstände zwischen den Klassenwahrscheinlichkeiten wider. So bieten sie eine deutlich reichhaltigere Trainingsbasis und die Anzahl der Trainingsdaten kann reduziert werden ohne dabei an Generalisierungsfähigkeit einzubüßen. Hinton et. al. [6] schlagen auch eine Variante vor, bei der eine kombinierte Lossfunktion mit gewichtetem Loss aus *soft targets* und *hard targets* eingesetzt wird, was die Ergebnisse weiter verbessert. Für die hier vorgestellte *Distillation* von Llama wird jedoch bewusst auf die Nutzung von *hard targets* verzichtet, da ein effizientes Distillieren mit wenigen Grunddaten und geringerer Ressourcennutzung das primäre Ziel darstellt, was noch genauer betrachtet wird.
Eine weitere wichtige Erkenntnis ist, dass eine höhere *temperature* des Modells für den Prozess der Distillation zu besseren Ergebnissen führt, Hinton et. al. [6] definieren das Anheben der *temperature* sogar als wesentlich.

Erstmals im Kontext von LLMs ist das Konzept der *Distillation* in bedeutenswerter Weise 2020 von Sanh et. al. [7] in dem Paper *DistilBERT* aufgegriffen worden. Hierbei wird ein um 40% kleineres Modell auf Basis von BERT distilliert, wobei 97% der Performance über verschiedene Tasks hinweg erhalten bleibt. Ein zu nennender Aspekt aus dem Paper ist zudem die Initialisierung der Layer des kleineren Modells mit den Gewichten des großeren Modells. Die Autoren haben hierbei jeden zweiten Layer benutzt, um das kleinere Modell aufzubauen. Auch Hinton et. al. [6] beschreiben bereits diese Idee in ihrem Paper in einem leicht anderen Kontext. Sie weisen aber zudem darauf hin, dass die Nutzung der Gewichte des *teacher-Modells* die benötigten Trainingsdaten für den *student* weiter reduzieren kann und bereits durch diesen Prozess ein wesentlicher Wissenstransfer stattfinden kann.
Die nach bestem Wissen einzige derzeit relevante Veröffentlichung über das Thema *Distillation* in Bezug auf das Llama Modell ist das Baby Llama Paper [8]. In diesem Paper wird ein kleineres Llama Modell mit weniger Layern und anderen Reduktionen verwendet, welches dann unter Nutzung eines *teacher-Ensembles* aus Llama und GPT-2 auf 10M Wörtern mit Hilfe der *Distillation* trainiert. Das methodische Vorgehen sowie die Ergebnisse in diesem Papier sind dabei an manchen Stellen kritisch zu betrachten. Beispielsweise ist für die Berechnung des *Distillation Loss* in *Baby Llama* der in [9] für einen *tiny-bert* use case geschriebene Trainer vollständig kopiert worden.

3.2 Distilling Llama

In dieser Arbeit wird ebenfalls auf dem in [9] definierten *DistillationTrainer* aufgebaut, jedoch wird dieser in wesentlichen Teilen modifiziert und in zahlreichen Details auf das primäre Ziel dieses Projekts zugeschnitten. Im erfolgreichen Aufsetzen dieses Trainers liegt der wesentliche Beitrag des *Distillationteils* dieser Arbeit, ein derartiger Trainer ist in dieser Form für Llama nach bestem Wissen noch nie veröffentlicht und damit eine völlige Neuschaffung.

Den Startpunkt der *Distillation* in dieser Arbeit bildet das Aufbauen eines verkleinerten Llama Modells mit weniger Layern, hierbei werden 11 Layer mitsamt der Gewichte aus den 32 Layern von Llama extrahiert, als Ergebnis liegt *Llama 1.4B* vor. In der praktischen Umsetzung gestaltet sich dieses Extrahieren insofern schwierig, dass jeder Layer auf den Index des vorherigen zugreift und es zu Fehlern kommt, wenn dieser Index nicht übereinstimmt. Da dieser Index nicht direkt angepasst werden kann, musste dieses Problem umgangen werden, indem die ersten 11 Layer geladen werden und dann nur die Gewichte der gewünschten Layer geladen werden.

Als zweiten Schritt wird das bereits vorgestellte QLoRa Finetuning auf *Llama 1.4B* angewandt. Dieser Schritt ist durchgeführt worden, um sicherzustellen, dass durch Nachtrainieren mit nur wenigen Daten bereits eine Wiederherstellung von ganzen Wörtern als Textoutput möglich ist, was zu einem zufriedenstellen Grad erfolgreich verlaufen ist. Im Gesamtprozess könnte dieser Schritt jedoch potentiell auch weggelassen werden.

Nach dieser Vorbereitung erfolgt nun der eigentliche Prozess der *Distillation* unter Nutzung des eigens entwickelten *DistillationTrainer*. Hierfür ist zunächst das allgemeine Ziel der *Distillation* in dieser Arbeit zu definieren, welches sich von derzeitiger Literatur unterscheidet. Da als Hauptziel ein ressourcen- sowie datensparendes Training festgesetzt wird, wird keine *Distillation* auf großen Mengen an Text wie bei Pre-Training Ansätzen durchgeführt. Stattdessen soll die Datengrundlage allein aus einem Fragenkatalog zu Themen aus einer Domäne, beispielsweise Vorlesungsbereichen, bestehen. Diese Fragen werden dann parallel an *teacher* und *student* gegeben und der Loss wird aus der Differenz der Antworten berechnet. Beim klassichen Trainieren auf Text wird jeweils der Loss bei der Vorhersage des nächsten Tokens herangezogen. Auch bei Klassifikationen muss nur jeweils ein Wahrscheinlichkeitsverktor verglichen werden, wofür auch der *DistillationTrainer* aus [9] konzipiert worden ist. Mit diesem Trainer würde bei den Fragen lediglich ein

Forwardpass durchgeführt und so der Loss bestimmt werden, also nur das erste token der Antwort generiert werden. Um dies zu vermeiden müssen alle Wahrscheinlichkeitstensoren (*probits*) der Anwort in einem Durchlauf generiert und dann iterativ zwischen beiden Modellen verglichen werden, aus der Summe ergibt sich der Loss. Dieser Aspekt der Generierung aller *probits* und das Zurückgeben eines Loss Wertes in einem für die transformers library passenden Format hat sich als eine enorme Schwierigkeit erwiesen. Weitere Details zu Herausforderungen bei der Implementierung und eine Sammlung zahlreicher Versuche Probleme zu lösen sind in den entsprechenden Notebooks dokumentiert.

4 Evaluation

Als Grundlage für die Evaluation wurden für fünf genutzte Vorlesungen insgesamt 100 Fragen mit jeweiligen Musterantworten entwickelt. Diese Fragen und Antworten basieren auf den über die Semester erstellten Karteikarten welche noch durch ChatGPT in ein einheitliches Format gebracht wurden. Zur Auswertung wurden zwei Methoden gewählt. Einerseits eine neuere qualitative Methode aus dem Paper "MT-Bench", welche auch als Grundlage für die Bewertung des Huggingface Chatbot-Leaderboards dient[10] und andererseits eine klassische Methode unter Verwendung von Kosinus-Abständen [11]. Bei der qualitativen Auswertung werden Antworten von zwei LLMs im Kontext einer Musterantwort verglichen, um zu bewerten, welche Antwort am besten und am nächsten an den Vorlesungsinhalten liegt. Obwohl diese Auswertung klassischerweise durch Menschen erfolgen könnte, wäre dies sehr kosten- und arbeitsintensiv. Hierbei kommt die Methode aus dem Paper "MT-Bench" zum Einsatz, wobei GPT-4 als Richter fungiert und bewertet, welche Antwort die bessere ist. Die Verwendung dieser Methodik wird durch wissenschaftliche Auswertungen gestützt, die eine 85 % Übereinstimmung zwischen menschlichen Bewertungen und den Antworten von GPT-4 zeigen. Diese Übereinstimmungsrate ist etwas höher als die zwischen menschlichen Antworten, die bei 81% liegt. Das Paper weist jedoch auch auf Schwächen der Auswertungsmethoden hin, wie etwa die leichte Bevorzugung der zuerst gegebenen Antwort, die leichte Bevorzugung längerer Antworten und die fehlende intrinsische Logik und Mathematikfähigkeit der Modelle, welche zu schlechteren Bewertungen führen können [10]. Um diese Probleme zu minimieren, wurden die Antworten der LLMs in zufälliger Reihenfolge an GPT-4 gesendet, um eine Gleichverteilung der zuerst gegebenen Antwort zu gewährleisten. Zusätzlich wurde die maximale Antwortlänge der LLMs auf die Maximalänge der Musterantworten für das jeweilige Fachgebiet beschränkt, um große Abweichungen in den Antwortlängen zu verhindern. Des Weiteren wurde die Musterantwort als Kontext mit an GPT-4 gesendet, um dem Modell einen Bewertungskontext zu geben. Ein weiteres Feature ist, dass GPT-4 seine Antwort begründet, um die Ergebnisse überprüfbar zu machen. Bei unterschiedlichen Bewertungen zwischen Menschen und LLMs wurden die Begründungen zu 75% als vernünftig angesehen und sogar 34% der Probanden waren bereit ihre Entscheidung zu ändern [10]. Als weitere Metrik zur Evaluierung wurde die Cosinusähnlichkeit gewählt. Dabei wurden die Antworttexte sowie die Musterantworten zuerst mit dem Word2Vec-Algorithmus "FastText", der auf englischen Wikipedia-Seiten und englischem Web-Crawling basiert [12], in Vektoren umgewandelt. Anschließend wurde die Kosinusähnlichkeit der jeweiligen LLM-Antwort mit der Musterantwort verglichen, um zu überprüfen, ob ähnliche Informationen vorhanden sind.
Nach Abschluss des Finetunings wurden die Antworten des neu gewonnenen Modells bei 56% der Testfragen durch GPT-4 als die bessere Antwort eingestuft. Jene Antworten wurden anhand der Kosinusähnlichkeit zur Standardantwort nur in 34% der Fälle als Sieger des Vergleichs bewertet. Das durchschnittliche Ergebnis dieses Verfahrens wird von einigen starken Ausreißern beeinflusst, während sich der Medianwert der Kosinusähnlichkeiten beider Modelle nur leicht unterscheidet. (5, 6, 7, 8)

5 Fazit

Rückblickend lassen sich Punkte herausarbeiten, die kritisch betrachtet werden können. Für einige Vorlesungen, zeigten die Ergebnisse zufriedenstellende Ergebnisse, für andere jedoch nicht. Durch eine nähere Betrachtung konnte festgestellt werden, dass die Inputs, für jene, die besser abgeschnitten haben, mehr Text und damit mehr Kontext enthielten. Daraus lässt sich folgern, dass der Input für die einzelnen Dateien individuell gestaltet werden muss. Ein wesentlicher Punkt ist jedoch die generelle Textbereinigung von Textstellen mit schwachem Informationsgehalt. Wir gehen davon aus, dass aufgrund von Overfitting Gleichungen und Code in die Antworten des finetuning Modells einbezogen wurden. Da solche Inhalte nicht in den Standardantworten enthalten sind, führt das zwangsläufig zu einer schlechten Kosinusähnlichkeit.

Mit dem Aufsetzen eines funktionierenden *DistillationTrainer* unter Überwindung zahlreicher technischer Herausforderungen ist ein relevanter Beitrag nicht nur in einem projektbezogenen Kontext, sondern auch auf dem allgemeinen Gebiet der *Distillation* geschaffen worden. Denn dadurch wird eine *Distillation* unter reiner Nutzung des *teachers* ohne grundlegenden Datensatz möglich. Aufgrund mangelnder Ressourcen ist die Evaluation des Ansatzes nur gering ausgefallen, da ein signifikantes Training nicht möglich war, Ergebnisse zeigen zwar ganze Wörter mit teils richtigem Kontext, jedoch noch keine sinnvollen Satzstrukturen. Mit tieferer Einarbeitung in technische Aspekte des Trainers sowie Optimierung und längerer Durchführung des Trainings werden jedoch reale Potentiale auf Basis theoretischer Grundlagen gesehen.

Appendix

$$W = W_0 + \Delta W$$

Figure 1. LoRA Basisformel

$$\Delta W \epsilon \mathbb{R}^{d \times k} = B \epsilon \mathbb{R}^{d \times r} \times A \epsilon \mathbb{R}^{r \times k}$$

Figure 2. LoRA Niedrigrangzerlegung

$$h = W_0 x + \Delta W x = W_0 x + BAx$$

Figure 3. LoRA Forward-Pass

Figure 4. Die Grafik zeigt den, mit Weigts and Biases gemessenen, *train loss* von den Inputs mit einer halben und mit vier Seiten. [Eigene Grafik]

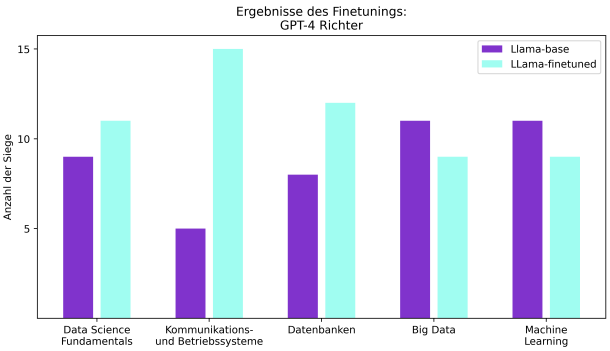

Figure 5. Zeigt die Anzahl der Siege der Modelle basierend auf GPT-4 als Richter. [Eigene Grafik]

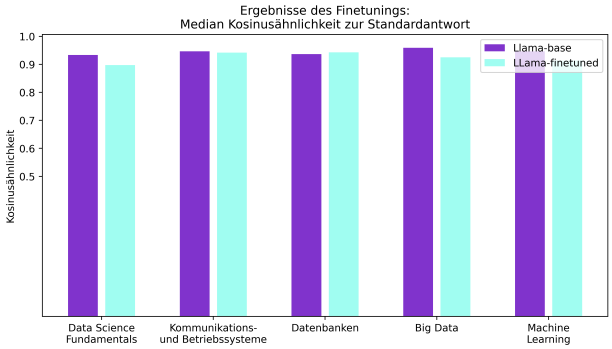

Figure 6. Zeigt den Median der Kosinusähnlichkeit beider Modelle zur Standardantwort. [Eigene Grafik]

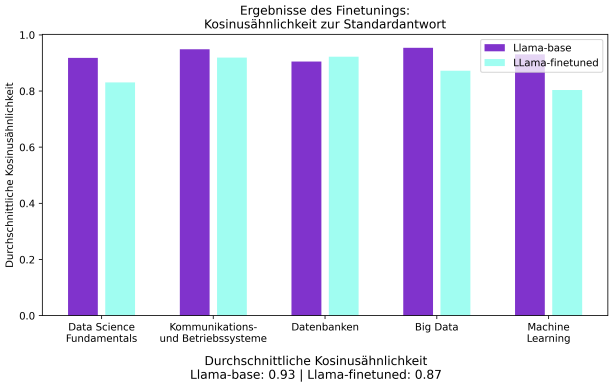

Figure 7. Zeigt die durchschnittliche Kosinusähnlichkeit beider Modelle. [Eigene Grafik]

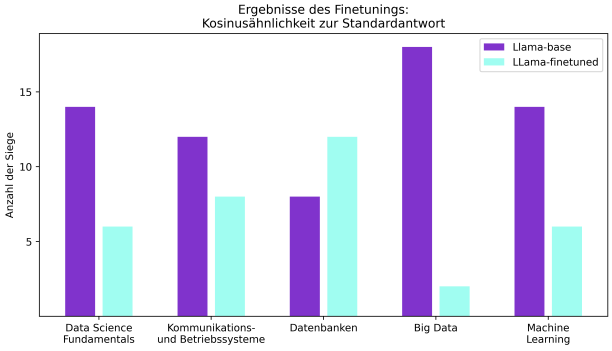

Figure 8. Zeigt die Anzahl der Siege der Modelle basierend auf der Kosinusähnlichkeit. [Eigene Grafik]

References

[1] *Pymupdf 1.23.14 documentation* (15.01.2024), `https://pymupdf.readthedocs.io/en/latest/`

[2] *Welcome to pypdf2 — pypdf2 documentation* (06.01.2023), `https://pypdf2.readthedocs.io/en/3.0.0/`

[3] GitHub, *tesseract-ocr/tesseract: Tesseract open source ocr engine (main repository)* (20.01.2024), `https://github.com/tesseract-ocr/tesseract?tab=readme-ov-file#tesseract-ocr`

[4] E.J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen (2021), publisher: arXiv Version Number: 2

[5] T. Dettmers, A. Pagnoni, A. Holtzman, L. Zettlemoyer (2023), publisher: arXiv Version Number: 1

[6] G. Hinton, O. Vinyals, J. Dean, *Distilling the knowledge in a neural network* (2015), 1503.02531

[7] V. Sanh, L. Debut, J. Chaumond, T. Wolf, *Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter* (2020), 1910.01108

[8] I. Timiryasov, J.L. Tastet, *Baby llama: knowledge distillation from an ensemble of teachers trained on a small dataset with no performance penalty* (2023), 2308.02019

[9] GitHub, *philschmid/knowledge-distillation-transformers-pytorch-sagemaker* (23.01.2024), https://github.com/philschmid

[10] L. Zheng, W.L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing et al., arXiv preprint arXiv:2306.05685 (2023)

[11] P.Y. Yuan, A.M. Du, C. Wang, *Using Word2vec to Match Knowledge Points and Test Questions: A Case Study*, in *2020 IEEE 2nd International Conference on Computer Science and Educational Informatization (CSEI)* (2020), pp. 272–276

[12] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, A. Joulin, *Advances in Pre-Training Distributed Word Representations*, in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)* (2018)

.

- Fabian Banovic
  - Motivation (Bericht)
  - Preprocessing (Code+Bericht)
  - Finetuning (Code)
  - Fazit (Bericht)
- Lukas Bruckner
  - Textextraktion (Code+Recherche)
  - Evaluation (Code+Bericht+Recherche)
  - Präsentation
  - Readme
- Joschua Jakubek
  - Distillation (Recherche)
  - Finetuning (Recherche + Code)
  - Evaluation Plots (Exploration + Code)
  - Abstract + Effizientes Finetuning mit QLoRA (Bericht)
- Marcus Wirth
  - Finetuning (Code + Bericht + Recherche)
  - Distillation (Code + Bericht + Recherche)
  - Motivation (Bericht)
  - Fazit (Bericht)