



FHNW Hochschule für Technik

Bachelor Thesis

Unsicherheitsabschätzung in Large Language Models mittels Reasoning-Strategien

Autor: Mauro Hirt

mauro.hirt@students.fhnw.ch

mauro.hirt@hotmail.com

Betreuer:

Daniel Perruchoud

daniel.perruchoud@fhnw.ch

Stephan Heule

stephan.heule@fhnw.ch

Windisch, August 2025

Abstract

Large Language Models (LLMs) werden zunehmend in risikosensitiven Kontexten eingesetzt; verlässliche Unsicherheitsabschätzung auf Antwortebene ist darum zentral. Diese Arbeit untersucht, wie Reasoning-Strategien (*Chain-of-Thought (CoT)*; *Chain-of-Draft (CoD)*; *Tree-of-Thought (ToT)*) zur Quantifizierung epistemischer Unsicherheit genutzt werden können. Verglichen werden Self-Evaluation-Verfahren (*P-True*, *Self-Probing*) und tokenwahrscheinlichkeitsbasierte Aggregationsmetriken (*Probas-Mean*, *Probas-Min*, *Token-SAR*) — jeweils als Final-Answer-only-Baselinsen und in reasoning-bewussten Varianten mit Keyword-Extraktion und Importance-Scoring entlang der Reasoning-Kette. Bewertet wird mit AUROC, ECE und Reliability Curves auf HOTPOTQA, 2WIKIMHQA, GSM8K, SVAMP und ASDIV (Llama-3.1-8B). Im ToT-Setting (GAME of 24, Breadth-First Search (BFS); DeepSeek-V3-Chat) wird die publizierte Evaluate-Heuristik durch tokenwahrscheinlichkeitsbasierte Scores ersetzt; dabei dienen Unsicherheitsquantifizierung (UQ)-Signale ausschließlich der internen Pfadauswahl zur Genauigkeitssteigerung und erzeugen keinen finalen Unsicherheitswert.

Ergebnisse: (1) Reasoning-bewusste, keyword-gewichtete Aggregationen verbessern Diskrimination und Kalibration konsistent gegenüber Final-Answer-Baselinsen; besonders stark sind *Probas-Min* und *Token-SAR*. Keyword-Selektion reduziert Längen-/Formatbias, Importance-Gewichte erhöhen die Sensitivität für entscheidungsrelevante Tokens; lokale Unsicherheiten in der Begründung werden früher sichtbar. (2) CoD erhält die Qualität der Unsicherheitsabschätzung weitgehend bei typischerweise geringerem Tokenbudget und –2.84 Prozentpunkten Accuracy gegenüber CoT. (3) Self-Evaluation bringt nur begrenzt Mehrwert: *P-True* zeigt einen engen Konfidenzbereich (limitierte Diskrimination), *Self-Probing* eine schwache Kopplung zwischen Confidence und tatsächlicher Korrektheit. (4) Im ToT-Experiment ergibt sich ein zweistufiger Kostenbefund: Zunächst senkt der Modellwechsel von GPT-4 auf DeepSeek die Kosten der ToT-Baseline um etwa 99 %; darauf aufbauend reduzieren die UQ-basierten Branch-Scores die DeepSeek-Baselinekosten nochmals um den Faktor 3–5, gehen jedoch mit deutlichen Genauigkeitseinbußen einher (von 72 % auf maximal 24 %; Single-Solution, *Probas-Mean*, $T \approx 1,8$).

Vorwort

Die Wahl des Themas entsprang einer zunehmenden Faszination für LLMs, die sich in den letzten Jahren deutlich vertieft hat. Das technologische Potenzial dieser Modelle ist noch kaum abschätzbar – gerade hierin liegt der Reiz. Konzepte wie technologische Singularität, sich selbst verbessernde Intelligenz oder Fragen des Alignment sind sowohl technisch als auch philosophisch und gesellschaftlich höchst relevant. Mit dem erwarteten Wachstum ihres Einsatzes gewinnt insbesondere die verlässliche Quantifizierung von Unsicherheit an Bedeutung, um LLMs verantwortungsvoll und ethisch einsetzen zu können. Diese Arbeit bot die Gelegenheit, eingehender in dieses Forschungsfeld einzutauchen und das eigene Wissen im Bereich des Maschinellen Lernens zu erweitern.

Für die wertvolle Unterstützung, die regelmässigen Fachgespräche und das stets angenehme Miteinander danke ich meinen Betreuern, **Daniel Perruchoud** und **Stephan Heule**, herzlich. Besonders dankbar bin ich auch für die Möglichkeit, während der Bearbeitungszeit einen Arbeitsplatz am Institut nutzen zu dürfen – dies hat die Durchführung der Experimente wesentlich erleichtert. Die Kombination aus fachlicher Begleitung und menschlicher Offenheit wurde sehr geschätzt.

Inhaltsverzeichnis

Vorwort	iii
Abbildungsverzeichnis	vii
Tabellenverzeichnis	viii
Abkürzungsverzeichnis	ix
1. Einleitung	1
1.1. Motivation	1
1.2. Problemstellung und Zielsetzung	1
1.3. Forschungsfragen	1
1.4. Aufbau der Arbeit	2
2. Grundlagen	3
2.1. Notation	3
2.2. Systematisierung von Unsicherheitsmethoden	4
2.2.1. Zielgrösse: Epistemische vs. Aleatorische Unsicherheit	4
2.2.2. Zugriffsebene: Blackbox, Greybox, Whitebox	4
2.2.3. Granularität: Prompt-Level vs. Antwort-Level	4
2.2.4. Einordnung der in dieser Arbeit verwendeten Methoden	5
2.3. Reasoning-Strategien	5
2.3.1. Chain-of-Thought (CoT)	5
2.3.2. Chain-of-Draft (CoD)	6
2.3.3. Tree-of-Thought (ToT)	6
2.4. Unsicherheitsmetriken	8
2.4.1. Self-Evaluation	9
2.4.2. Aggregationsmetriken	10
2.5. Bewertungsmetriken für Unsicherheitswerte	12
2.5.1. Kalibrationsmetriken	12
2.5.2. Diskriminationsmetrik	13
3. Methodik	15
3.1. Experimenteller Rahmen	15
3.2. Materialien	16
3.2.1. Datengrundlagen	16
3.2.2. LLM-Backends	17

Inhaltsverzeichnis

3.3.	Implementierung CoT/CoD Experiment	18
3.3.1.	Inference Prompt	18
3.3.2.	Keyword Extraction & Importance Scoring	19
3.3.3.	Confidence-Scores	20
3.3.4.	Ground-Truth-Bewertung	23
3.4.	Tree-of-Thought (ToT) Experiment	24
3.4.1.	Modellumstellung (GPT-4 → DeepSeek-V3-Chat)	24
3.4.2.	Baseline-Generate-Prompt	24
3.4.3.	Antwortvalidierung	26
3.4.4.	Austausch der <i>Evaluate</i> -Metrik	27
3.5.	Auswertungsprotokoll	29
3.5.1.	Geltungsbereich und Datengrundlage	29
3.5.2.	Bewertungsmetriken	29
3.5.3.	Workflow und Visualisierung	30
3.5.4.	Experimentsspezifische Unterschiede	31
3.6.	Reproduzierbarkeit	31
4.	Experimente und Ergebnisse	32
4.1.	Quantitative Resultate	32
4.1.1.	Accuracy	32
4.1.2.	Diskriminationsfähigkeit	33
4.1.3.	Kalibrationsergebnisse	37
4.2.	Laufzeit- und Kostenanalyse	42
4.2.1.	CoT / CoD – LLaMA-3 (Cluster)	42
4.2.2.	GPT-4o-mini API – Kostenübersicht	42
4.2.3.	Kostenvergleich der ToT-Experimente	43
4.3.	Qualitative Resultate	43
4.3.1.	Fallstudien: <i>Chain-of-Draft</i> (CoD)	43
4.3.2.	Stichprobenprüfung: String-Matching-Verifier (ASDiv) und LLM-Judge (HotpotQA)	44
4.4.	Zusammenfassung der Ergebnisse	45
5.	Diskussion	47
5.1.	Interpretation der Hauptergebnisse	47
5.2.	Vergleich mit Vorarbeiten	49
5.3.	Limitationen und Validitätsbedrohungen	49
5.4.	Praktische Implikationen	50
6.	Fazit und Ausblick	51
6.1.	Zusammenfassung der wichtigsten Erkenntnisse	51
6.2.	Beantwortung der Forschungsfragen	51
6.3.	Ausblick auf zukünftige Arbeiten	52
Literaturverzeichnis		53

Inhaltsverzeichnis

A. Appendix	56
A.1. Beispielaufgaben aus jedem Datensatz	56
A.2. Komplette Instruktionstemplates für Self-Evaluation Methoden	56
A.2.1. Self-Probing	56
A.2.2. P-True	58
A.3. Reproduzierbare Umgebung und Ausführung	59
A.3.1. Python-Abhängigkeiten	59
A.3.2. Singularity Build Skript	60
A.3.3. Dockerfile	61
A.4. Reliability Curves	62
A.4.1. Chain-of-Thought (CoT)	62
A.4.2. Chain-of-Draft (CoD)	67
A.5. Erweiterte Accuracy-Auswertungen	69
A.6. Brier Score	70
A.6.1. Brier Score (CoD)	70
A.6.2. Brier Score (CoT)	71
Ehrlichkeitserklärung	72

Abbildungsverzeichnis

2.1. Beispielhafte ToT-Suche (BFS, $b = 5$) für eine Game-of-24-Aufgabe [1]	8
2.2. Beispiel einer Reliability Curve [2]	13
3.1. Experimentelle Pipeline für CoT und CoD.	15
4.1. AUROC-Leaderboard für das CoT-Experiment und Vergleich mit CoT-UQ-Paper-Ergebnissen. Höhere Werte sind besser.	35
4.2. AUROC Leaderboard für das CoD Experiment. Höhere Werte sind besser. .	36
4.3. Vergleich Baseline Aggregationsmetriken CoD vs. CoT. Höhere Werte sind besser.	37
4.4. ECE-Leaderboard für das CoT-Experiment. Niedrigere Werte sind besser. .	38
4.5. ECE-Leaderboard für das CoD-Experiment. Niedrigere Werte sind besser. .	39
4.6. Reliability-Vergleich für das CoD-Experiment (ASDiv als Beispiel): (a) Aggregated Probabilities, (b) P_{true} , (c) Self-Probing.	40
4.7. AUROC vs. ECE Scatter für das CoT-Experiment (Aggregationsmetriken). .	41
A.1. Reliability-Vergleich der CoT -Varianten für HotpotQA	62
A.2. Reliability-Vergleich der CoT -Varianten für SVAMP	63
A.3. Reliability-Vergleich der CoT -Varianten für GSM8K	64
A.4. Reliability-Vergleich der CoT -Varianten für 2WikiMHQA	65
A.5. Reliability-Vergleich der CoT -Varianten für ASDiv	66
A.6. Reliability-Vergleich der CoD -Varianten für HotpotQA	67
A.7. Reliability-Vergleich der CoD -Varianten für SVAMP	68
A.8. Reliability-Vergleich der CoD -Varianten für GSM8K	69

Tabellenverzeichnis

2.1.	Verwendete Symbole	3
2.2.	Überblick über die in dieser Arbeit untersuchten Unsicherheitsmetriken (Abschnitt 2.4) nach der Taxonomie aus Abschnitt 2.2.	5
2.3.	Benchmark-Ergebnisse auf GSM8K (Few-Shot): Vergleich CoT → CoD [3].	6
2.4.	Erfolgsraten auf dem Game-of-24-Benchmark (GPT-4).	8
3.1.	Überblick über die verwendeten Datensätze.	17
4.1.	Durchschnittliche Accuracy (5 Läufe) von CoT und CoD; Δ entspricht der Differenz (CoT–CoD).	32
4.2.	Erfolgsraten auf dem <i>Game-of-24</i> -Benchmark – Vergleich GPT-4 und DeepSeek (ToT-Baselines).	33
4.3.	Accuracy der unsicherheitsbasierten ToT-Varianten bei $b = 5$	33
4.4.	Vergleich der durchschnittlichen Laufzeit pro Run für CoT- und CoD-Experimente	42
4.5.	Kumulierte GPT-4o-mini-API-Nutzung (Tarif: \$0.15 / 10^6 Input-Tokens, \$0.60 / 10^6 Output-Tokens; Stand Juli 2025)	42
4.6.	Kosten des ToT-Experiments (DeepSeek-API).	43
4.7.	Kosten des ToT-Experiments mit GPT-4 laut [4].	43
4.8.	Reasoning-Kette, extrahierte Keywords und Token-Wahrscheinlichkeiten (Fallstudie 1)	44
4.9.	Reasoning-Kette, extrahierte Keywords und Token-Wahrscheinlichkeiten (Fallstudie 2)	44
A.1.	Beispielaufgaben aus jedem Datensatz	56
A.2.	Erweiterte Genauigkeitsstatistiken (CoT). Fünf Läufe pro Datensatz; Angaben in Prozent. 95 %-KI nach Student- t ($df = 4$).	70
A.3.	Erweiterte Genauigkeitsstatistiken (CoD). Fünf Läufe pro Datensatz; Angaben in Prozent. 95 %-KI nach Student- t ($df = 4$).	70
A.4.	Brier Score (CoD): Baseline vs. Keyword-Variante (Mittel über 5 Läufe). Niedriger ist besser.	70
A.5.	Brier Score (CoT): Baseline vs. Keyword-Variante (Mittel über 5 Läufe). Niedriger ist besser.	71

Abkürzungsverzeichnis

LLM Large Language Model

CoT Chain-of-Thought

CoD Chain-of-Draft

ToT Tree-of-Thought

DFS Depth-First Search

BFS Breadth-First Search

Token-SAR Token-Level Shifting Attention to Relevance

ECE Expected Calibration Error

AUROC Area Under the Receiver Operating Characteristic Curve

TPR True Positive Rate

FPR False Positive Rate

UQ Unsicherheitsquantifizierung

1. Einleitung

1.1. Motivation

Large Language Models (LLMs) werden zunehmend in sensiblen Domänen eingesetzt – von Wissensabfragen über Mathematik bis hin zu Planungsaufgaben. Mit dieser Verbreitung steigt der Bedarf, die Verlässlichkeit ihrer Antworten transparent zu bewerten. Ein zentrales Element ist die Quantifizierung von Unsicherheit: Nur wenn Modelle signalisieren, wann sie sich irren könnten, lassen sich Risiken angemessen steuern, z. B. durch Rückfragen, Eskalation an Menschen oder automatisches Verwerfen unsicherer Antworten. Reasoning-Strategien wie *Chain-of-Thought* (CoT), *Chain-of-Draft* (CoD) und *Tree-of-Thought* (ToT) eröffnen zudem die Möglichkeit, intern genutzte Zwischenschritte explizit zu machen und diese für die Unsicherheitsabschätzung nutzbar zu machen.

1.2. Problemstellung und Zielsetzung

Bestehende Unsicherheitsmetriken für LLMs fokussieren häufig auf Prompt-Level-Aggregation oder rein tokenstatistische Größen der finalen Antwort. Dadurch bleiben semantische Unterschiede innerhalb der Reasoning-Kette ungenutzt, und Kalibrationsprobleme bestehen fort. Ziel dieser Arbeit ist es, Antwort-Level-Unsicherheit unter Einbeziehung von Reasoning-Strategien systematisch zu untersuchen. Konkret sollen:

- Selbstbewertungsmethoden (*P-True*, *Self-Probing*) und Aggregationsmetriken (*Probas-Mean*, *Probas-Min*, *Token-SAR*) unter CoT und CoD verglichen werden,
- der Effekt semantischer Gewichtung der Reasoning-Kette durch Keyword-Extraktion evaluiert werden,
- der Effekt des Einbezuges verschiedener Kontexte der Reasoning-Kette auf die Self-Evaluation Methoden evaluiert werden,
- und im ToT-Setting geprüft werden, ob heuristische Zwischenbewertungen kostengünstig durch tokenwahrscheinlichkeitsbasierte Scores ersetzt werden können.

1.3. Forschungsfragen

F1 (Aggregation): Verbessern Reasoning-Ketteninformationen (Keywords und Importance-Scores) die Aggregationsmetriken (*Probas-Mean*, *Probas-Min*, *Token-SAR*) gegenüber Final-Answer-only-Baselines — gemessen an Area Under the Receiver Ope-

1. Einleitung

rating Characteristic Curve (AUROC), Expected Calibration Error (ECE) und Kalibrationskurven?

F2 (Self-Evaluation): Verbessern Reasoning-Kontexte (Allsteps/Keystep/Allkeywords/Keywords) die Self-Evaluation-Methoden (*P-True, Self-Probing*) gegenüber deren Final-Answer-only-Baselines — hinsichtlich AUROC, ECE und Kalibrationskurven?

F3 (Effizienz CoD vs. CoT): Erhält CoD die UQ-Qualität (AUROC/ECE) von CoT bei reduziertem Tokenbudget und geringerer Latenz, und wie groß ist der Accuracy-Trade-off?

F4 (ToT: Kosten–Genauigkeit): Welchen Kosten–Genauigkeit-Trade-off erzeugt im ToT-BFS (Game of 24) der Ersatz der `evaluate`-Heuristik durch tokenwahrscheinlichkeitsbasierte Scores (*Probas-Mean/Min*), und welches Generierungsregime (Single vs. Multiple; Temperatur) schneidet am besten ab?

1.4. Aufbau der Arbeit

Kapitel 2 beschreibt Notation, Taxonomie der Unsicherheitsmethoden, Reasoning-Strategien sowie die verwendeten Metriken. Kapitel 3 erläutert die Methodik, Datensätze, Backends und Implementierungsdetails. Kapitel 4 berichtet die Ergebnisse zu Genauigkeit, Diskrimination, Kalibration sowie Laufzeit und Kosten. Kapitel 5 diskutiert die Befunde, ordnet sie im Kontext der Literatur ein und adressiert Limitationen. Kapitel 6 fasst die Erkenntnisse zusammen, beantwortet die Forschungsfragen und skizziert künftige Arbeiten.

2. Grundlagen

2.1. Notation

Tabelle 2.1.: Verwendete Symbole

Symbol	Beschreibung
q	Frage
σ	Instruktions-Template mit Platzhaltern
θ	Substitutionsmapping $\{ \langle \text{QUESTION} \rangle \mapsto q, \langle \text{RESPONSE} \rangle \mapsto r, \langle \text{KONTEXT}_x \rangle \mapsto C_x \}$
$\sigma[\theta]$	Instruktions-Prompt nach Platzhalterersetzung
$\pi = \sigma[\theta]$	Gesamter Prompt, der an das Modell gesendet wird
r	Detokenisierte Modellantwort (Klartext inklusive Reasoning-Chain)
C_x	Kontextausschnitt $C_x \subseteq r$ (mit $x \in \{\text{bl}, \text{all}, \text{key}, \text{allkw}, \text{keykw}\}$) als Teilmenge der detokenisierten Modellantwort r
T	Anzahl der Antworttoken, $T \in \mathbb{N}_{\geq 1}$
$\tau = (\tau_1, \dots, \tau_T)$	Tokenfolge der Antwort
$\mathbf{p} = (p_1, \dots, p_T)$	Token-Wahrscheinlichkeiten (Softmax-Output), $0 \leq p_t \leq 1$
$\mathcal{J} = \{1, \dots, T\}$	Grundmenge aller Token-Indizes
$\mathcal{I} \subseteq \mathcal{J}$	Teilmenge der Token-Indizes (z. B. Keyword-Token)
w_t	Importance-Gewicht (Importance-Score) des Tokens t , diskret $1 \dots 10$
$\text{sim}(r, r \setminus \tau_t)$	Ähnlichkeit zweier Antwortrepräsentationen, bestimmt mit <code>roberta-large</code> Cross-Encoder, $0 \leq \text{sim} \leq 1$
$r \setminus \tau_t$	Antwort ohne den Token-String τ_t
v_t / \tilde{v}_t	Semantische Wichtigkeit bzw. normalisierte Variante
c	Confidence-Score der Antwort, $0 \leq c \leq 1$
a	Accuracy, $0 \leq a \leq 1$
y	Ground-Truth-Label, $y \in \{0, 1\}$
\hat{y}	Modellvorhersage, $\hat{y} \in \{0, 1\}$
N	Anzahl der Fragen im Datensatz
\mathcal{D}	Datensatz: $\mathcal{D} = (q^{(i)}, y^{(i)}, \hat{y}^{(i)}, r^{(i)})_{i=1}^N$

2.2. Systematisierung von Unsicherheitsmethoden

In diesem Abschnitt werden bestehende Methoden zur Unsicherheitsquantifizierung nach verschiedenen Dimensionen klassifiziert.

2.2.1. Zielgrösse: Epistemische vs. Aleatorische Unsicherheit

- **Aleatorische Unsicherheit:** Entsteht durch inhärente Zufälligkeit im Datenraum. Sie ist irreduzibel, z.B. bei ambigen oder mehrdeutigen Aufgaben[5].
- **Epistemische Unsicherheit:** Resultiert aus mangelndem Wissen oder Modellsicherheit, etwa durch unzureichende Trainingsdaten. Sie kann durch zusätzliche Information reduziert werden[5].

In der Praxis ist die Unterscheidung zwischen epistemischer und aleatorischer Unsicherheit meist nicht zuverlässig möglich; beobachtbar ist primär ihre Summe[5].

2.2.2. Zugriffsebene: Blackbox, Greybox, Whitebox

In der Forschung werden drei Zugriffsebenen unterschieden, welche den Grad des Einblicks in die internen Zustände eines LLM bestimmen[6].

Black-Box Es liegen ausschliesslich Ein- und Ausgaben vor; Logits, Gewichte und verdeckte Aktivierungen bleiben vollständig verborgen[6].

Grey-Box Neben den Ein- und Ausgaben sind eingeschränkt interne Signale (in unserem Fall Tokenwahrscheinlichkeiten) zugänglich, während die Modellgewichte weiterhin unzugänglich bleiben[6].

White-Box Sämtliche Modellgewichte sowie alle Zwischenaktivierungen können eingesehen werden, wodurch detailliertere Analysen ermöglicht werden[6].

Die Zuordnung dieser Zugriffsebenen ist in der Literatur nicht einheitlich. In einigen Arbeiten wird lediglich zwischen Black-Box- und White-Box-Zugriff unterschieden; Methoden mit eingeschränktem Einblick, wie etwa Grey-Box-Verfahren, werden dabei der White-Box-Kategorie zugeordnet[7].

2.2.3. Granularität: Prompt-Level vs. Antwort-Level

- **Prompt-Level:** Unsicherheit wird über mehrere Antworten aggregiert[8].
- **Antwort-Level:** Unsicherheit wird aus einer einzelnen Antwort abgeleitet[8].

Die ausschliesslich promptbasierte Unsicherheitsbewertung wird kritisiert, da sämtlichen Antworten auf denselben Prompt ein identischer Unsicherheitswert zugewiesen wird, ungeachtet möglicher semantischer Unterschiede. Eine Antwort-level basierte Unsicherheitsquantifizierung ermöglicht demgegenüber eine differenziertere Bewertung der Vertrauenswürdigkeit[8].

2. Grundlagen

Beispiel *Prompt:* “What is the capital of France?”

#	Antwort	Confidence c
1	Paris	0.95
2	Lyon	0.42
3	Paris	0.93

Aus den drei Antworten ergibt sich ein aggregierter Prompt-Level-Wert von $\bar{c} = 0.77$, während die Antwort-Level-Werte c die individuellen Unterschiede sichtbar machen.

2.2.4. Einordnung der in dieser Arbeit verwendeten Methoden

Die in Abschnitt 2.2 beschriebene Taxonomie ermöglicht eine eindeutige Klassifikation aller in dieser Arbeit untersuchten Unsicherheitsverfahren.¹

Tabelle 2.2.: Überblick über die in dieser Arbeit untersuchten Unsicherheitsmetriken (Abschnitt 2.4) nach der Taxonomie aus Abschnitt 2.2.

Methode	Zielgrösse	Zugriffsebene	Granularität
$p(\text{True})$	Epistemisch	Grey-Box	Response-Level
Self-Probing	Epistemisch	Black-Box	Response-Level
Probas-Mean	Epistemisch	Grey-Box	Response-Level
Probas-Min	Epistemisch	Grey-Box	Response-Level
Token-SAR	Epistemisch	Grey-Box	Response-Level

2.3. Reasoning-Strategien

LLMs können durch gezielt gestaltete Eingabeaufforderungen (*Prompts*) dazu veranlasst werden, unterschiedliche Problemlösungsstrategien explizit offenzulegen. Drei Verfahren – CoT, CoD und ToT – werden im Folgenden vorgestellt.

2.3.1. Chain-of-Thought (CoT)

Grundidee Beim CoT-Prompting wird das Modell ermutigt, eine explizite Folge logisch verknüpfter Zwischenschritte zu erzeugen, um die Gesamtaufgabe in Teilprobleme zu zerlegen.

Historischer Kontext Signifikante Leistungssteigerungen auf arithmetischen, Common-Sense- und symbolischen Benchmarks wurden berichtet [9]. Es wurde gezeigt, dass bereits ein einzelnes Zero-Shot-Instruktions-Token „Let’s think step by step.“ deutliche Genauigkeitsgewinne bewirken kann [10].

¹Die Zuordnung richtet sich nach dem jeweils überwiegenden Unsicherheitstyp. In der Praxis überlagern sich epistemische und aleatorische Anteile jedoch häufig.

2. Grundlagen

Instruktions-Template σ

Let's think step by step.
<QUESTION>

Ergebnisse Auf dem GSM8K-Mathematik-Benchmark stieg die Genauigkeit eines 540B-PaLM-Modells durch CoT von 15 % auf 57 %. Bei `text-davinci-002` erhöhte Zero-Shot-CoT die Trefferquote von 17.7 % auf 78.7 % [10].

2.3.2. Chain-of-Draft (CoD)

Grundidee CoD veranlasst das Modell, sehr knappe, semantisch dichte Zwischennotizen (*Drafts*) zu generieren, um eine CoT-ähnliche Reasoning-Qualität bei geringerem Tokenverbrauch und reduzierter Antwortlatenz zu erreichen.

Originalarbeit Der Ansatz wurde in „Chain of Draft: Thinking Faster by Writing Less“ vorgestellt [3]. Erste Experimente berichteten nahezu unveränderte Genauigkeit bei ungefähr 7.6 % des Tokenaufwands von CoT [3].

Instruktions-Template σ

Think step by step, but only keep a minimum draft for each thinking step, with 5 words at most.
<QUESTION>

Ergebnisse Die wichtigsten Resultate für das Few-Shot-Setting auf dem GSM8K-Benchmark sind in Tabelle 2.3 dargestellt. Sowohl GPT-4O als auch CLAUDE-3 reduzieren den Tokenverbrauch um knapp 80 % und verkürzen die Antwortlatenz um rund 50–75 %, wobei der Genauigkeitsverlust mit etwa 4 Prozentpunkten moderat bleibt.

Tabelle 2.3.: Benchmark-Ergebnisse auf GSM8K (Few-Shot): Vergleich CoT → CoD [3].

Modell	Tokenbudget	Latenz	Genauigkeit
GPT-4o	205 → 44 (−78.5 %)	4.2 s → 1.0 s (−76.2 %)	95.4 % → 91.1 % (−4.3 pp)
Claude-3	190 → 40 (−79.0 %)	3.1 s → 1.6 s (−48.4 %)	95.8 % → 91.4 % (−4.4 pp)

Bei kleineren Modellen im Bereich von 1.5–3 Mrd. Parametern lässt sich ebenfalls eine Tokenersparnis von 50–80 % erzielen; dieser Gewinn geht jedoch mit deutlich stärkeren Genauigkeitseinbußen von 8–27 Prozentpunkten einher [3].

2.3.3. Tree-of-Thought (ToT)

Grundidee ToT verallgemeinert CoT zu einer expliziten Baumsuche. In jedem Zwischen-schritt werden mehrere *nächste Schritte* generiert, heuristisch bewertet und gemäss dieser Bewertung selektiv vertieft. Damit werden Exploration und Exploitation ähnlich einer Monte-Carlo-Baumsuche kombiniert [4].

2. Grundlagen

Suchstrategie (BFS vs. Depth-First Search (DFS)) Für Aufgaben mit geringer Baumtiefe und klarer Zwischenbewertung – exemplarisch das *Game of 24* – erweist sich eine (BFS) als vorteilhaft. Probleme mit tieferen oder erst spät bewertbaren Suchräumen (z. B. Mini-Crosswords) profitieren von einer Tiefensuche (DFS) mit Backtracking. Im Rahmen dieser Arbeit wird ausschliesslich der BFS-Modus anhand des *Game of 24* betrachtet.

Spielregeln (Game of 24) Gegeben sind vier ganze Zahlen. Mittels der Grundrechenarten $+$, $-$, \times und \div sowie beliebiger Klammerung soll exakt der Zielwert 24 gebildet werden; jede Zahl darf höchstens einmal verwendet werden, Divisionen durch 0 sind ausgeschlossen.

Kombinatorik (Game of 24). Obere Schranke der Operationspfade (3 Züge, ungeordnete Paarwahl; beide Richtungen für $-$, \div):

$$\binom{4}{2}6 \cdot \binom{3}{2}6 \cdot \binom{2}{2}6 = 3888.$$

Diese Zahl dient als obere Grenze und überzählt: (i) ungültige Divisionen ($/0$), (ii) Zustandskollisionen (sofortige oder spätere Pfadkonvergenzen zu identischen Multimengen), (iii) Kollaps der zwei Richtungen bei identischen Operanden ($x - x$, x/x). Der effektive Suchraum ist somit $<= 3888$.

Ablauf (Game of 24, BFS, Beam-Grösse $b = 5$)

1. **Generate:** Für den aktuellen Zustand werden *mehrere nächste Schritte* vorgeschlagen.
2. **Evaluate:** Jeder Schritt wird dreifach hinsichtlich Plausibilität als „sure“, „maybe“ oder „impossible“ eingestuft. Die Bewertungen werden mittels der Gewichte 20, 1, 0.1 in einen Score überführt und aufsummiert.
3. **Select:** Die fünf (b) Teillösungen mit dem höchsten Score werden beibehalten und in der nächsten Ebene expandiert. (Der Score wird pro Schritt neu berechnet und nicht über den Gesamtpfad der 3 Schritte aufsummiert.)
4. **Terminate:** Nach genau drei Ebenen werden die vier Pfade mit dem höchsten Score als Endantwort ausgegeben.
5. **Validate:** Danach werden die Endanworten auf Korrektheit validiert.

Ergebnisse BFS

Stärken

- Deutliche Genauigkeitssteigerung bei breiten, flachen Suchräumen.

2. Grundlagen

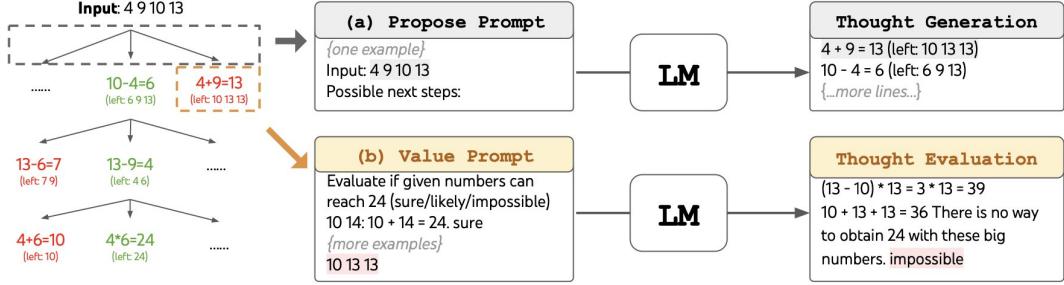


Abbildung 2.1.: Beispielhafte ToT-Suche (BFS, $b = 5$) für eine Game-of-24-Aufgabe [1].

Schwächen

- Laufzeit- und Kostenanstieg proportional zu Suchbreite, -tiefe *und* der Anzahl *Evaluate*-Aufrufe (drei pro Teillösung).
- Notwendiges Heuristik-Tuning (Branching-Faktor, Beam-Grösse, Bewertungsprompt).

Tabelle 2.4.: Erfolgsraten auf dem Game-of-24-Benchmark (GPT-4).

Methode	Beam-Grösse b	Erfolgsrate [%]	Quelle
CoT (Baseline)	—	4	[4]
ToT + BFS	1	45	[4]
ToT + BFS	5	69–74 ²	[11, 4]

Anwendungen ToT mit BFS eignet sich insbesondere flache und breite Suchräume wie Game of 24. Weitere Experimente (Mini-Crosswords, generative Story-Planung) zeigten ebenfalls Leistungsvorteile, werden jedoch nicht vertieft diskutiert.

2.4. Unsicherheitsmetriken

Die Zuverlässigkeit der von LLMs generierten Antworten wird gegenwärtig intensiv untersucht. Zur quantitativen Bewertung dieser Zuverlässigkeit dienen Unsicherheitsmetriken. Diese Metriken liefern skalare Größen, welche in nachgelagerten Schritten – etwa zur Fehlererkennung, Antwortselektion oder Modellkalibrierung – eingesetzt werden können.

Üblicherweise werden zwei grundlegende Klassen unterschieden:

1. Self-Evaluation:

Durch explizite Selbsteinschätzung wird eine Confidence c für die Richtigkeit der generierten Antwort ausgegeben.

²Das ToT-Paper meldete 74 %. Eine Reproduktion der Trajektorien (Log-Datei in ihrem Repo) erzielte 69 % [11].

2. Grundlagen

2. Aggregationsmetriken:

Implizite Wahrscheinlichkeitsinformationen aus dem Wahrscheinlichkeitsvektor \mathbf{p} werden zu einem Unsicherheitswert verdichtet.

2.4.1. Self-Evaluation

Self-Evaluation-Methoden ermöglichen es dem LLM, seine eigene Antwort auf Unsicherheit zu überprüfen, oft durch gezieltes Prompting. Im Folgenden werden zwei gängige Varianten beschrieben.

$p(\text{true})$

Definition Sei $\pi_{\text{eval}} = \sigma_{\text{eval}}[\theta]$ der Evaluations-prompt, der durch Einsetzen von q und r in das Template σ_{eval} mittels des Mappings θ entsteht. Bezeichne τ_1 als erstes Token der Modellantwort auf π_{eval} und p_1 als dessen Softmax-Wahrscheinlichkeit ($p_1 \in \mathbf{p}$). Dann gilt:

$$c = p_{\text{true}} = \begin{cases} p_1, & \text{falls } \tau_1 = \text{"True"} \\ 1 - p_1, & \text{falls } \tau_1 = \text{"False"} \end{cases}$$

Erläuterung Die Grösse p_{true} quantifiziert die Unsicherheit, indem das LLM seine zuvor generierte Antwort r als Schülerlösung präsentierte und anschliessend zwischen den Klassen „True“ und „False“ wählt. Die Softmax-Wahrscheinlichkeit des Tokens „True“ (bzw. das Komplement bei Auswahl von „False“) wird als Confidence-Score c verwendet.[12]

Instruktions-Template σ_{eval}

The problem is: <QUESTION>
A student submitted: <RESPONSE>
Is the student's answer:
(A) True
(B) False
The student's answer is:

Self-Probing

Definition Sei $\pi_{\text{eval}} = \sigma_{\text{eval}}[\theta]$ der Evaluations-Prompt, der durch Einsetzen von q und r in das Template σ_{eval} entsteht. Liefert das Modell daraufhin einen numerischen Wert $r_{\text{eval}} \in [0, 1]$, so wird

$$c = r_{\text{eval}}$$

als Confidence-Score verwendet.

2. Grundlagen

Erläuterung Self-Probing quantifiziert das subjektive Vertrauen des Modells, indem es die eigene Antwort r bewerten lässt. Das Verfahren erfordert weder Sampling noch Zugriff auf Logits. Empirische Untersuchungen [13, 14] zeigen eine signifikante Korrelation zwischen c und der tatsächlichen Antwortkorrektheit.

Instruktion-Templates σ_{eval} Nach [13] dient der folgende Prompt zur Bewertung einer Modellantwort:

Question: <QUESTION>

Possible Answer: <RESPONSE>

Q: How likely is the above answer to be correct?

Please first show your reasoning concisely and then answer in the format:

“Confidence: <numerical probability>”

2.4.2. Aggregationsmetriken

Aggregationsmetriken verdichten die Token-Wahrscheinlichkeiten $\mathbf{p} = (p_1, \dots, p_T)$ einer Antwort zu einem Skalar c , um die Unsicherheit quantitativ abzuschätzen. [15]. Dabei wird die Verteilung der Wahrscheinlichkeiten über die Tokenfolge τ berücksichtigt. Da lediglich vorhandene Token-Wahrscheinlichkeiten verarbeitet werden, entstehen nur geringe Zusatzkosten

Probas-Mean

Definition Die Konfidenz c wird als arithmetisches Mittel der maximalen Softmax-Wahrscheinlichkeiten $\mathbf{p} = (p_1, \dots, p_T)$ der erzeugten Token τ_1, \dots, τ_T definiert [15]:

$$c = \frac{1}{T} \sum_{t=1}^T p_t.$$

Erläuterung Durch Mittelung über alle Token wird die durchschnittliche Modellsicherheit approximiert. Lokalisierte Unsicherheiten werden geglättet, sodass insbesondere bei langen Antworten tendenziell zu hohe Konfidenzwerte resultieren [16].

Probas-Min

Definition Probas-Min setzt die Konfidenz c als minimale Softmax-Wahrscheinlichkeit innerhalb der Tokenfolge fest [15]:

$$c = \min_{t=1, \dots, T} p_t.$$

Erläuterung Die Berücksichtigung des Tokens mit der geringsten Wahrscheinlichkeit ermöglicht eine rasche Identifikation punktueller Unsicherheiten. Bereits ein einzelner Token mit niedrigem Wahrscheinlichkeitswert reduziert c unmittelbar und weist potenziell auf Halluzinationen, Unschärfe oder Faktizitätsfehler hin. In [15] wurde gezeigt, dass

2. Grundlagen

mithilfe dieser Eigenschaft Halluzinationen während der Textgenerierung zuverlässig erkannt werden können.

Token-SAR

Definition Token-Level Shifting Attention to Relevance (Token-SAR) [17] bezeichnet ein Mass für den Confidence-Score c einer Antwort r in Abhängigkeit von der Tokenfolge $\tau = (\tau_1, \dots, \tau_T)$ sowie den korrespondierenden Tokenwahrscheinlichkeiten $\mathbf{p} = (p_1, \dots, p_T)$. Die Berechnung erfolgt in drei aufeinanderfolgenden Schritten:

$$v_t = 1 - \text{sim}(r, r \setminus \tau_t)$$

$$\tilde{v}_t = \frac{v_t}{\sum_{t=1}^T v_t}$$

$$c = \sum_{t=1}^T \tilde{v}_t p_t$$

Erläuterung

- *Semantische Token-Wichtigkeit*

Die erste Gleichung bestimmt die semantische Relevanz eines Tokens, indem die Ähnlichkeit der ursprünglichen Antwort mit der Antwort nach Entfernung von τ_t verglichen wird (Tabelle 2.1). Ein grosser semantischer Verlust resultiert in einem hohen v_t .

- *Gewichtsnormalisierung*

Die zweite Gleichung normiert die Relevanzwerte v_t derart, dass deren Summe eins beträgt. Die so entstehenden Gewichte \tilde{v}_t spiegeln die relativen Wichtigkeiten wider.

- *Gewichtete Aggregation*

Die dritte Gleichung liefert den Confidence-Score c als nach semantischer Relevanz gewichtetes Mittel der Tokenwahrscheinlichkeiten.

Gewichtete Varianten

Motivation Tokens weisen häufig unterschiedliche Bedeutungen auf. Ein nicht-negativer Gewichtungsfaktor w_t macht es möglich, Positions-, Relevanz- oder Domänenwissen in die Aggregation der Token-Wahrscheinlichkeiten einzubeziehen.

2. Grundlagen

Gewichtetes Probas-Mean Die Konfidenz c wird als gewichtetes arithmetisches Mittel definiert:

$$c = \frac{\sum_{t=1}^T w_t p_t}{\sum_{t=1}^T w_t}.$$

gemäss Unterunterabschnitt 2.4.2.

Gewichtetes Token-SAR Die Gewichte w_t fliessen in die dritte Aggregationsstufe von Token-SAR (Unterunterabschnitt 2.4.2) ein und werden anschliessend normalisiert:

$$c = \frac{\sum_{t=1}^T w_t \tilde{v}_t p_t}{\sum_{t=1}^T w_t \tilde{v}_t}$$

Für $w_t = 1$ entsteht das ungewichtete Probas-Mean respektive Token-SAR

2.5. Bewertungsmetriken für Unsicherheitswerte

Die Qualität der modellintern prognostizierten Unsicherheitswerte wird in zwei Dimensionen beurteilt:

- **Kalibration** – prüft die Übereinstimmung zwischen prognostizierter *Confidence* c und beobachteter *Accuracy* a .
- **Diskrimination** – misst die Trennschärfe zwischen korrekten und fehlerhaften Antworten.

2.5.1. Kalibrationsmetriken

ECE

Der ECE fasst den mittleren absoluten Kalibrationsfehler über alle Konfidenzbereiche zusammen. Der Wertebereich der Confidences wird in K disjunkte Bins $\mathcal{B}_1, \dots, \mathcal{B}_K$ partitioniert. Für jedes Bin \mathcal{B}_k werden die mittlere Confidence \bar{c}_k und die empirische Accuracy a_k bestimmt:

$$\bar{c}_k = \frac{1}{n_k} \sum_{i \in \mathcal{B}_k} c_i, \quad a_k = \frac{1}{n_k} \sum_{i \in \mathcal{B}_k} \mathbf{1}\{y_i = \hat{y}_i\},$$

wobei n_k die Anzahl Beispiele im Bin bezeichnet. Der ECE lautet

2. Grundlagen

$$\text{ECE} = \sum_{k=1}^K \frac{n_k}{N} |a_k - \bar{c}_k|,$$

mit $N = \sum_{k=1}^K n_k$. Ein Wert von 0 weist auf perfekte Kalibration hin, während grössere Werte systematische over- oder underconfidence anzeigen. Die Wahl von K beeinflusst die Granularität und wird üblicherweise auf zehn bis fünfzehn Bins gesetzt.

Reliability Curve

Die Reliability Curve visualisiert die Kalibration, indem für jedes Bin (aus der ECE berechnung) das Paar (c_k, a_k) in einem zweidimensionalen Diagramm dargestellt wird. Eine ideal kalibrierte Verteilung folgt der Diagonalen $a = c$. Abweichungen oberhalb dieser Linie kennzeichnen underconfidence, Abweichungen darunter overconfidence. Somit ergänzt die Kurve die eindimensionale ECE um lokale Informationen über das gesamte Confidence-Spektrum hinweg.

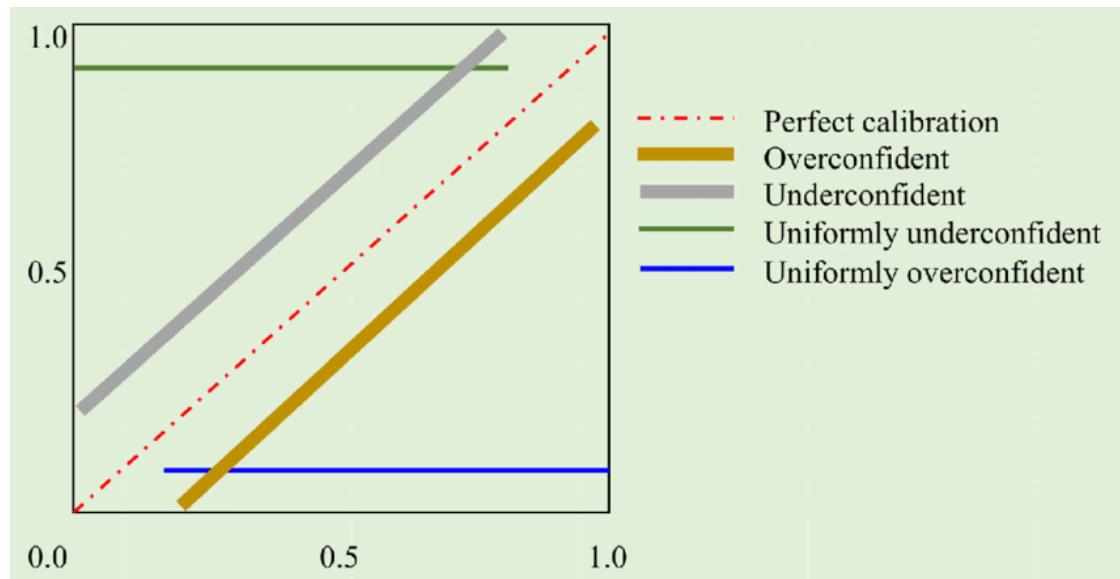


Abbildung 2.2.: Beispiel einer Reliability Curve [2].

2.5.2. Diskriminationsmetrik

AUROC

Die AUROC bewertet die Fähigkeit eines Modells, fehlerhafte Antworten (negative Klasse) von korrekten Antworten (positive Klasse) über alle möglichen Schwellen der Confidence c zu unterscheiden. Dazu wird die True Positive Rate (TPR) gegen die False Positive Rate (FPR) aufgetragen und die Fläche unter der resultierenden Kurve berechnet. Ein Wert von 0.5 entspricht zufälligem Raten, während ein Wert von 1 perfekte Diskrimination

2. Grundlagen

signalisiert. Hohe AUROC-Werte sind insbesondere in sicherheitskritischen Anwendungen wünschenswert, da dort Fehlentscheidungen minimiert werden müssen.

3. Methodik

3.1. Experimenteller Rahmen

CoT/CoD-Experiment

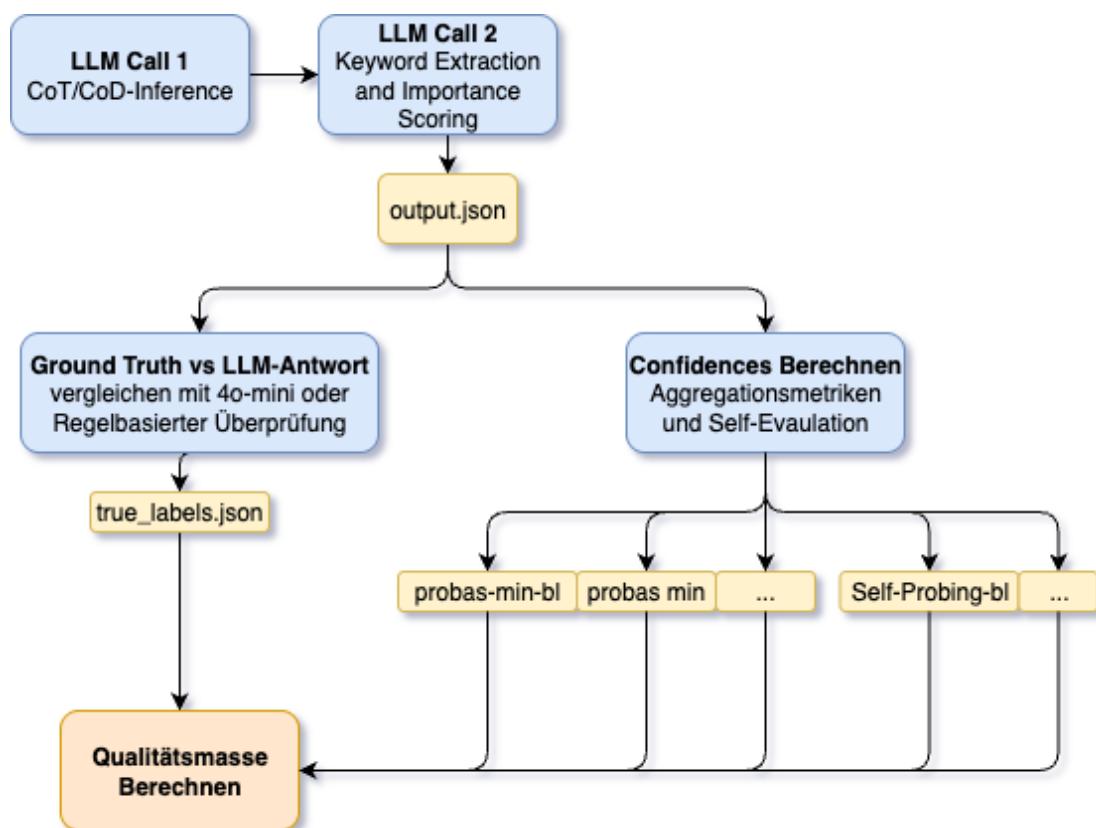


Abbildung 3.1.: Experimentelle Pipeline für CoT und CoD.

Ablauf

1. **LLM-Aufruf 1 — Inference-Prompt mit eingesetzter Frage** (vgl. Abschnitt 3.3.1): Ein Frageprompt wird an das ausgewählte LLM übermittelt. Das Modell generiert die vollständige Reasoning-Kette, die finale Antwort sowie die zuge-

3. Methodik

hörigen Token-Wahrscheinlichkeiten. Sämtliche Ergebnisse werden in `output.json` gespeichert.

2. **LLM-Aufruf 2 — Keyword-Prompt** (vgl. Abschnitt 3.3.2): Die in Schritt 1 erzeugte Antwort dient als Eingabe, um Schlüsselwörter zu extrahieren und diesen jeweils einen Importance-Score zuzuordnen. Die Resultate werden ebenfalls in `output.json` abgelegt.
3. **Berechnung der Unsicherheitsmetriken** (vgl. Abschnitt 3.3.3): Alle Metriken gemäss Abschnitt 3.3.3 werden bestimmt und als Dateien im Ordner `Confidences`persistiert.
4. **Ground-Truth-Bewertung** (vgl. Abschnitt 3.3.4): Die Label-Erzeugung ist datensatzspezifisch. (i) Für numerisch/arithmetische Datensätze (*GSM8K*, *SVAMP*, *ASDiv*) erfolgt eine regelbasierte Prüfung per String/Integer-Match (Heuristik); (ii) für freie, nicht-numerische QA (*2WikiMHQA*, *HotpotQA*) erfolgt eine semantische Äquivalenzbewertung mit `gpt-4o-mini` (LLM-Judge). Alle Labels werden in `true_labels.json` persistiert.
5. **Bestimmung der Qualitätskennzahlen** (vgl. Abschnitt 3.5.2): Die in Schritt 3 berechneten Confidence-Scores werden mit den Ground-Truth-Labels kombiniert, um ECE, Accuracy sowie AUROC zu ermitteln.

Datensatzbasis Die Experimente werden auf den in Tabelle 3.1 aufgeführten Benchmarks *HotpotQA*, *2Wiki MHQA*, *GSM8K*, *SVAMP* und *ASDiv* durchgeführt.

ToT-Experiment

Der in Unterabschnitt 2.3.3 vorgestellte Ansatz aus [4] wird reproduziert und als Baseline herangezogen. Hierfür wird ein Modell eingesetzt, das den Zugriff auf Tokenwahrscheinlichkeiten ermöglicht. Der im Original verwendete Branching-Evaluator wird durch eine Unsicherheitsmetrik ersetzt, die auf Tokenwahrscheinlichkeiten basiert (*Probas-Mean* beziehungsweise *Probas-Min*). Ziel ist eine Steigerung der Genauigkeit durch das Selektieren vertrauenswürdiger Branches und eine Reduktion der Inferenzkosten, da der Evaluationsprompt für jede vorgeschlagene Teillösung entfällt.

3.2. Materialien

3.2.1. Datengrundlagen

Übersicht der Datensätze

Die Benchmarks HOTPOTQA, 2WikiMHQA, GSM8K, SVAMP und ASDIV wurden dem *CoT-UQ*-Repository [24] entnommen und unverändert übernommen. Der Datensatz *Game-of-24* stammt aus dem offiziellen *Tree-of-Thought*-Repository [23]. Für HotpotQA und 2Wiki MHQA wurden ausschliesslich die jeweiligen *Test-Splits* ohne Kontext herangezogen,

3. Methodik

Tabelle 3.1.: Überblick über die verwendeten Datensätze.

Datensatz	$ D $	$\bar{\Omega}$ Token	Domäne	Quelle
HotpotQA	8 447	23.2	Logisches Reasoning	[18]
2WikiMHQA	1 548	14.5	Logisches Reasoning	[19]
GSM8K	1 319	58.9	Mathematisches Reasoning	[20]
SVAMP	1 000	39.4	Mathematisches Reasoning	[21]
ASDiv	2 249	38.2	Mathematisches Reasoning	[22]
Game-of-24	100	—	Arithmetisch-symbolisches Reasoning	[23]

um die Vergleichbarkeit zu gewährleisten. Beim Game-of-24-Korpus beschränkte sich die Evaluation auf die schwierigen Instanzen (IDs 901–1000). Beispielhafte Aufgaben sämtlicher Benchmarks sind im Anhang A.1 (Tabelle A.1) aufgeführt.

3.2.2. LLM-Backends

Llama-3.1-8B auf dem Slurm-Cluster

- **Modell:** `meta-llama/Llama-3.1-8B-Instruct` [25]
- **Hardware:** NVIDIA RTX A4500 (20 GB VRAM); Ausführung mittels Slurm
- **Basis-Image (Docker):**

```
FROM nvidia/cuda:11.8.0-cudnn8-runtime-ubuntu22.04
```
- **Numerische Präzision:** `torch.bfloat16`
- **Generierungsparameter:** `temperature = 1.0, max_new_tokens = 128`

Die vollständige Dockerumgebung einschliesslich sämtlicher Abhängigkeiten und Ausführungsskripte sind in Kapitel 3.6 dokumentiert.

API-basierte Modelle

GPT-4o mini [26]. Das Modell wird ausschliesslich als automatischer *Judge* zur Antwortverifikation eingesetzt; zusätzliche Konfigurationsparameter sind nicht erforderlich.

DeepSeek-v3-Chat [27]. Das Modell wird im *Tree-of-Thought*-Setting (vgl. Kapitel 3.4) genutzt. Die Temperatur variiert je nach Experiment zwischen 0.7 und 2.0; weiterführende Details sind in Kapitel 3.4 dokumentiert.

3.3. Implementierung CoT/CoD Experiment

3.3.1. Inference Prompt

Dieser Abschnitt dokumentiert den im Schritt LLM-Aufruf 1 eingesetzten Instruktionsprompt zur Generierung von Modellantworten (vgl. Abbildung 3.1). Die konzeptionellen Unterschiede zwischen der ausführlichen CoT und der kompakten CoD hinsichtlich Antwortlatenz, Tokenverbrauch und Genauigkeit wurden bereits im Kapitel 2.3 dargelegt und werden an dieser Stelle nicht erneut behandelt. Im Folgenden werden die zwei verwendeten Prompt-Varianten sowie beispielhafte Modellantworten aufgeführt.

Instruktionsprompt A: Ausführliche Chain-of-Thought (CoT).

Please reason the following question step by step.
Label each reasoning step as "Step i:", where "i" is the step number.
You need to ensure that each step builds on the previous one and contributes meaningfully toward reaching the final answer.
Once you finish all steps, put your final answer on a separate line after the reasoning steps, starting with "Final Answer:" (do not label it as a step).

Question: <QUESTION>
Response: Let's think step by step.

Instruktionsprompt B: Kompakte Chain-of-Draft (CoD).

Think step by step, but only keep a minimum draft for each thinking step, with 5 words at most.
Label each step as "Step i:", where "i" is the step number.
Once you finish all steps, put your final answer on a separate line after the reasoning steps, starting with "Final Answer:" (do not label it as a step).

Question: <QUESTION>
Response: Let's think step by step.

Beispiel (Vergleich von Bandmitgliedern). Die nachfolgenden Modellantworten veranschaulichen die Struktur beider Varianten.

CoT-Ausgabe (ausführlich):

Step 1: We Are the Ocean has 5 members.
Step 2: The Dream Academy has 3 members.
Step 3: 5 is greater than 3.
Step 4: Therefore, We Are the Ocean has more members.
Final Answer: We Are the Ocean

CoD-Ausgabe (kompakt):

3. Methodik

Step 1: We Are the Ocean: 5
Step 2: The Dream Academy: 3
Step 3: 5 > 3
Final Answer: We Are the Ocean

Hinweis. Die Extraktion der finalen Antwort erfolgt anhand der klar definierten Delimiter-Zeile `Final Answer`:. Die explizite Kennzeichnung der Teilschritte mittels `Step i`: ermöglicht ein robustes Parsing der Reasoning-Kette. Dies erleichtert die Zuordnung von Evaluationsmetriken zu den jeweiligen Argumentationsschritten und reduziert das Risiko fehlerhafter Interpretationen im nachgelagerten Analyseprozess.

3.3.2. Keyword Extraction & Importance Scoring

Zielsetzung. Im Rahmen von LLM-Aufruf 2 (vgl. Abbildung 3.1) werden aus der in Schritt 1 erzeugten Antwort in Form einer CoT- beziehungsweise CoD-Struktur für jeden einzelnen Reasoning-Schritt (*Step i*) die jeweils relevanten Schlüsselwörter extrahiert. Jedem extrahierten Schlüsselwort wird anschliessend ein ordinaler Relevanzwert auf einer Skala von 1 (geringste Relevanz) bis 10 (höchste Relevanz) zugeordnet. Die resultierenden Paare aus Schlüsselwort und Relevanzwert werden in der Datei `output.json` gespeichert.

Prompt-Template.

You will be provided with a question and a multi-step response containing reasoning steps.
For each long reasoning step labeled "Step i:", extract the keywords, only the relevant tokens for that specific reasoning step.
You also need to evaluate the importance of each keyword to the final answer. Please evaluate the importance score following the keyword by (/<importance score>/) on a scale of 1 to 10, where 1 is the least critical and 10 is the most critical.
If you find more than one keyword in a specific step, separate them with ";".
If a specific step does not contribute meaningfully to deriving the final answer (e.g., repeating information already provided in the question, introducing irrelevant assumptions or speculations), return "Step i: NO ANSWER" for that step.

Question: <QUESTION>

Multi-Step Response: <REASONING CHAIN>

Keywords for Each Reasoning Step:

Beispiel. Für die Frage «*Which band has more members, “We Are the Ocean” or “The Dream Academy”?*» und die zugehörige mehrstufige Begründung

Step 1: The question is asking which band has more members.
Step 2: “We Are the Ocean” has 5 members.
Step 3: “The Dream Academy” has 3 members.
Step 4: 5 is greater than 3.

3. Methodik

Step 5: Therefore, “We Are the Ocean” has more members.
Final Answer: We Are the Ocean

liefert das Modell den folgenden Schlüsselwort-Output:

Step 1: NO ANSWER
Step 2: We Are the Ocean(/5/); 5(/10/)
Step 3: The Dream Academy(/5/); 3(/10/)
Step 4: greater(/7/)

Anmerkungen. Die Bewertungsskala von 1 bis 10 ermöglicht eine quantitative Einschätzung der semantischen Relevanz eines Tokens im Hinblick auf die Ableitung der finalen Antwort. Ein hoher Relevanzwert kennzeichnet somit einen besonders entscheidungsrelevanten Beitrag innerhalb der Argumentationskette. Reasoning-Schritte, die keinen zusätzlichen Informationsgewinn liefern (beispielsweise Redundanzen oder irrelevante Annahmen), werden explizit mit NO ANSWER markiert. Dies erleichtert die nachgelagerte Analyse, etwa zur gezielten Filterung weniger relevanter Tokens.

3.3.3. Confidence-Scores

Aggregationsmetriken

Aufbauend auf den Aggregationsmetriken in Kapitel 2.4.2 werden in diesem Abschnitt diejenigen Varianten spezifiziert, die im Rahmen der Experimente berechnet werden. Jede Variante ist durch (i) die berücksichtigte Tokenmenge und (ii) die zugrunde liegende Aggregationsformel eindeutig definiert.

Tokenmengen. Es seien $\mathcal{J} = \{1, \dots, T\}$ die Indizes aller vom Modell erzeugten Token der vollständigen Sequenz (Reasoning *und* finale Antwort; vgl. Abschnitt 2.1). Daraus werden zwei disjunkte Mengen definiert:

- **Baseline-Menge** $\mathcal{I}_{\text{bl}} \subseteq \mathcal{J}$: ausschliesslich jene Token der finalen Antwort nach dem Limiter „Final Answer.“.
- **Keyword-Menge** $\mathcal{I}_{\text{kw}} \subseteq \mathcal{J}$: alle Token der Reasoning-Kette, die als Keywords erkannt wurden. Jedem Token $t \in \mathcal{I}_{\text{kw}}$ ist ein diskretes Wichtigtgewicht $w_t \in \{1, \dots, 10\}$ zugeordnet (vgl. Unterunterabschnitt 2.4.2).

Formale Definition der Varianten. Sei $f \in \{\text{Probas-Min}, \text{Probas-Mean}, \text{Token-SAR}\}$ eine der in Unterabschnitt 2.4.2 definierten Aggregationsfunktionen. Die Gleichungen *Probas-Min* (Unterunterabschnitt 2.4.2), *Probas-Mean* (Unterunterabschnitt 2.4.2) sowie *Token-SAR* (Unterunterabschnitt 2.4.2). Dann werden pro Funktion zwei Varianten berechnet:

3. Methodik

$$c = f(\{p_t\}_{t \in \mathcal{I}_{\text{bl}}}), \quad (\text{Baseline})$$

$$c = \begin{cases} f(\{p_t\}_{t \in \mathcal{I}_{\text{kw}}}), & \text{falls } f = \text{Probas-Min}, \\ f_w(\{(p_t, w_t)\}_{t \in \mathcal{I}_{\text{kw}}}), & \text{falls } f \in \{\text{Probas-Mean, Token-SAR}\}, \end{cases} \quad (\text{Keyword})$$

Hierbei bezeichnet f_w die in Unterunterabschnitt 2.4.2 vorgestellte gewichtete Variante ($w_t \neq 1$). Für *Probas-Min* wird das Gewicht hingegen ignoriert, da das Minimum unter positiver Skalierung unverändert bleibt.

Namenskonvention. Die resultierenden sechs UnsicherheitsScores tragen jeweils ein sprechendes Präfix bl für baseline:

bl-probas-min, bl-probas-mean, bl-token-sar, probas-min, probas-mean,
token-sar.

Dieses Schema ermöglicht eine eindeutige Zuordnung der Ergebnisse zu Metrik und Tokenmenge.

Implementierungsdetails. Alle Scores werden nach der Inferenz einmalig pro Antwort berechnet und im Verzeichnis **confidences** persistiert. Der zusätzliche Rechenaufwand ist gering, da ausschliesslich bereits vorliegende Softmax-Wahrscheinlichkeiten \mathbf{p} weiterverarbeitet werden.

Self-Evaluation

Zwei prompt-basierte Verfahren dienen zur Antwortselbstbewertung Unterabschnitt 2.4.1:

- **Self-Probing** – das LLM liefert explizit einen Confidence-Wert $c \in [0, 1]$ für die Korrektheit seiner Antwort.
- **P-True** – dem LLM wird eine *True/False*-Frage gestellt; aus der Tokenwahrscheinlichkeit der Antwort wird $c \in [0, 1]$ bestimmt.

Notation. Die vom Modell erzeugte Token-Sequenz sei $\mathcal{J} = \{1, \dots, T\}$. Jedem Token $t \in \mathcal{J}$ ist ein diskretes Wichtigkeitsgewicht $w_t \in \{1, \dots, 10\}$ zugeordnet (vgl. Absatz 3.3.3).

Darauf aufbauend werden aus Teilmengen von \mathcal{J} fünf *Kontextmengen* abgeleitet, die den **tatsächlichen Textkontext** für die prompt-basierten Verfahren bilden:

- \mathcal{C}_{bl} : Text der finalen Antwort (Baseline),
- \mathcal{C}_{all} : gesamte Reasoning-Kette,
- \mathcal{C}_{key} : Text des Reasoning-Schritts mit dem höchsten durchschnittlichen Gewicht \bar{w}_ℓ ,
- $\mathcal{C}_{\text{allkw}}$: alle erkannten Keywords,
- $\mathcal{C}_{\text{keykw}}$: ausgewählte KeyKeywords mit $w_t \geq 4$ oder die drei Keywords mit höchstem w_t .

3. Methodik

Promptvarianten. Für die beiden Self-Evaluation Metriken *Ptrue* und *Self-probing* wird mit fünf Prompt Konfigurationen gearbeitet, die den oben eingeführten Kontexte benutzen:

1. **baseline**

$$\pi = \sigma[\theta(q, \mathcal{C}_{\text{bl}})]$$

2. **allsteps**

$$\pi = \sigma[\theta(q, \mathcal{C}_{\text{bl}}, \mathcal{C}_{\text{all}})]$$

3. **keystep**

$$\pi = \sigma[\theta(q, \mathcal{C}_{\text{bl}}, \mathcal{C}_{\text{key}})]$$

4. **allkeywords**

$$\pi = \sigma[\theta(q, \mathcal{C}_{\text{bl}}, \mathcal{C}_{\text{allkw}})]$$

5. **keykeywords**

$$\pi = \sigma[\theta(q, \mathcal{C}_{\text{bl}}, \mathcal{C}_{\text{keykw}})]$$

Instruktionsvorlagen σ . Für jedes der beiden Verfahren e Instruktionsvorlage σ definiert, in die das jeweilige Substitutionsmapping $\theta(q, \mathcal{C}_{\text{bl}}, \mathcal{C}_x)$ eingesetzt wird. Nachfolgend je ein Beispiel für die Variante *allsteps*; alle weiteren Templates (Baseline, Keystep, All-/KeyKeywords) sind analog aufgebaut und in Anhang A.2 gelistet.

Self-Probing (σ , Beispiel *allsteps*)

Question: <QUESTION>

Possible Answer: <KONTEXT_bl>

A step-by-step reasoning to the possible answer:

<KONTEXT_all>

Q: Considering these reasoning steps as additional information,
how likely is the above answer to be correct?

Please first show your reasoning concisely and then answer with:

“‘Confidence: [the probability of answer <KONTEXT_bl> to be correct,
please include only the numerical number]%'”‘

Confidence:

P-True (σ , Beispiel *allsteps*)

The problem is: <QUESTION>

A student submitted: <KONTEXT_bl>

The student explained the answer, which included a step-by-step reasoning: <KONTEXT_all>

Considering these reasoning steps as additional information,

3. Methodik

is the student's answer:

- (A) True
- (B) False

The student's answer is:

Namenskonvention. self-probing-allsteps, self-probing-keystep, self-probing--allkeywords, self-probing-keykeywords sowie analog ptrue-allsteps ... ptrue--keykeywords.

Implementierungsdetails. Die Kontexte C_x werden nach der Antworterzeugung (LLM Call 1 und 2; vgl. Abbildung 3.1) generiert. Anschliessend wird das LLM pro Variante genau einmal mittels Prompt aufgerufen. Die resultierenden Self-Evaluation-Scores werden gemeinsam mit den anderen Unsicherheits-Scores im Verzeichnis `confidences` gespeichert. Der zusätzliche Rechenaufwand ist hoch, da für jede Variante ein weiterer LLM-Aufruf erforderlich ist.

3.3.4. Ground-Truth-Bewertung

Zielsetzung Für jedes beantwortete Beispiel wird ein binäres Label (korrekt/inkorrekt) erzeugt, das als Zielvariable für AUROC, ECE und Accuracy dient (vgl. Abbildung 3.1). Die Vorgehensweise unterscheidet sich nach Aufgabentyp: (i) numerisch/arithmetisch vs. (ii) frei formulierte Antworten.

(A) Numerische/arithmetische Datensätze (GSM8K, SVAMP, ASDiv). Hier wird eine regelbasierte Heuristik verwendet, die tolerant gegenüber den meisten Formatierungsvarianten ist. Das Label ist *true*, wenn die korrekte Zahl als Teilstring in der LLM-Antwort vorkommt—entweder exakt als Stringrepräsentation oder als ganzzahliger String.

(B) Freie QA (2WikiMHPQA, HotpotQA). Für nicht-numerische Aufgaben wird die inhaltliche Äquivalenz mittels eines Bewertungsmodells (gpt-4o-mini) geprüft. Zum Einsatz kommt folgender Evaluations-Prompt; die Modellantwort (“yes/no”) wird in ein binäres Label überführt:

```
You are an automated grading assistant helping a teacher grade student answers.
```

```
The problem is: <QUESTION>
```

```
The correct answer for this problem is: <GROUND_TRUTH>
A student submitted the answer: <ANSWER>
```

```
The student's answer must be correct and specific but not
overcomplete (for example, if they provide two different answers,
they did not get the question right). However, small differences in
formatting should not be penalized (e.g., "New York City" vs. "NYC").
```

3. Methodik

Did the student provide an equivalent answer to the ground truth?
Please answer yes or no without any explanation:

3.4. Tree-of-Thought (ToT) Experiment

Ziel und Überblick

Dieses Experiment verfolgt das Ziel, die publizierten Ergebnisse der ToT-BFS-Suche für das *Game of 24* im Rahmen eines Modelltransfers zu reproduzieren (Ausgangsmodell: GPT-4; Zielmodell: DeepSeek-V3-Chat). Darüber hinaus wird die bisher eingesetzte heuristische Bewertungsfunktion (**Evaluate-Heuristik**) durch eine auf Unsicherheitsmetriken basierende Evaluierung ersetzt.

Untersucht werden dabei (i) die Transferierbarkeit der ursprünglichen Prompts und notwendige Anpassungen, (ii) der Einfluss zweier Varianten der Unsicherheitsquantifizierung (A vs. B, vgl. Abschnitt 3.4.4) sowie (iii) die Wirkung unterschiedlicher Temperatureinstellungen auf Erfolgsrate und Lösungsdiversität.

Als zentrale Leistungskennzahlen (vgl. Kapitel 4) dienen *Accuracy* und Kostenprofil.

3.4.1. Modellumstellung (GPT-4 → DeepSeek-V3-Chat)

Die Umstellung erfolgte, da (i) die Inferenzkosten signifikant reduziert werden können und (ii) DeepSeek-V3-Chat im Gegensatz zum derzeit verfügbaren GPT-4-Zugang die für die Unsicherheitsquantifizierung (UQ) erforderlichen Token-Wahrscheinlichkeiten liefert.

Kosten (pro 1 M Tokens; ein Run = 100 Aufgaben).

- **GPT-4:** \$30 Input + \$60 Output → \$74 pro Run [4]
- **DeepSeek-V3-Chat:** \$0,27 Input + \$0,55 Output → \$0,72 pro Run

Ersparnis: Reduktion der Kosten um ca. 99 % allein durch den Modelltausch.

3.4.2. Baseline-Generate-Prompt

Der Wechsel des Grundmodells machte eine Prompt-Erweiterung erforderlich, da DeepSeek-V3-Chat ohne präzise Instruktionen längere und vom gewünschten Format abweichende Ausgaben erzeugte. Durch explizite Regeln wird das AusgabefORMAT konsistent gehalten. Dies entspricht dem *Generate*-Schritt in Absatz 2.3.3.

Ursprünglicher Prompt (GPT-4).

```
Input: 2 2 4 6
Possible next steps:
4 * 6 = 24  (left: 2 2 24)
2 * 6 = 12  (left: 2 4 12)
```

3. Methodik

```
2 * 4 = 8  (left: 2 8 6)
4 + 6 = 10 (left: 2 2 10)
2 + 6 = 8  (left: 2 4 8)
2 + 4 = 6  (left: 2 6 6)
6 / 2 = 3  (left: 2 4 3)
2 / 2 = 1  (left: 1 4 6)
6 - 2 = 4  (left: 2 4 4)
4 - 2 = 2  (left: 2 2 6)
2 - 2 = 0  (left: 0 4 6)

Input:
Possible next steps:
```

Angepasster Prompt (DeepSeek-V3-Chat).

```
**Rules:**
* Use each number exactly once.
* Combine two numbers per step using +, -, *, or /.
* No extra text or comments, just the "Possible next steps:" output as shown in the example.
```

```
Input: 2 8 8 14
Possible next steps:
2 + 8 = 10 (left: 8 10 14)
8 / 2 = 4  (left: 4 8 14)
14 + 2 = 16 (left: 8 8 16)
2 * 8 = 16 (left: 8 14 16)
8 - 2 = 6  (left: 6 8 14)
14 - 8 = 6 (left: 2 6 8)
14 / 2 = 7 (left: 7 8 8)
14 - 2 = 12 (left: 8 8 12)
```

```
Input: <INPUT>
Possible next steps:
```

Baseline-Evaluate-Heuristik

Zur Bewertung der in Absatz 2.3.3 beschriebenen Teillösungen wird in der Baseline ein Few-Shot-Prompt eingesetzt, der jede Zahlenkombination als „sure“, „likely“ oder „impossible“ klassifiziert. Ein gekürzter Auszug lautet:

```
Evaluate if given numbers can reach 24 (sure/likely/impossible)

10 14
10 + 14 = 24
sure

11 12
11 + 12 = 23
12 - 11 = 1
```

3. Methodik

```
11 * 12 = 132
11 / 12 = 0.91
sure

5 7 8
5 + 7 + 8 = 12 + 8 = 20
(8 - 5) * 7 = 3 * 7 = 21
I cannot obtain 24 now, but numbers are within a reasonable range
likely

5 6 6
5 + 6 + 6 = 17
(6 - 5) * 6 = 1 * 6 = 6
I cannot obtain 24 now, but numbers are within a reasonable range
likely

10 10 11
10 + 10 + 11 = 31
(11 - 10) * 10 = 10
10 10 11 are all too big
impossible

1 3 3
1 * 3 * 3 = 9
(1 + 3) * 3 = 12
1 3 3 are all too small
impossible
```

<INPUT>

Die Modellantworten werden mit den Gewichten 20 (sure), 1 (likely) beziehungsweise 0.1 (impossible) skaliert. Jede potenzielle Fortsetzung wird dreifach bewertet; die resultierenden Werte werden aufsummiert und bilden einen Score, der im *Select*-Schritt zur Auswahl der $b = 5$ bestbewerteten Teillösungen herangezogen wird.

3.4.3. Antwortvalidierung

Nach dem *Terminate*-Schritt (Schritt 4 in Absatz 2.3.3), der die vier bestbewerteten Pfade liefert, erfolgt die abschliessende Evaluierung (*Validate*, Schritt 5). Diese Prüfung verläuft zweistufig: Zunächst wird aus den Zwischenschritten eine kompakte Gleichung generiert; anschliessend wird diese Gleichung validiert.

Erzeugung der Antwortgleichung.

Your task is to generate the Answer field that shows the full trajectory of the solution. No explanation or comments.
Strictly adhere to the format shown in the example.

3. Methodik

```
Input: 4 4 6 8
Steps:
4 + 8 = 12 (left: 4 6 12)
6 - 4 = 2 (left: 2 12)
2 * 12 = 24 (left: 24)
Answer: (6 - 4) * (4 + 8) = 24
```

```
Input: <INPUT>
```

Formale Validierung.

Use numbers and basic arithmetic operations (+ - * /) to obtain 24.
Given an input and an answer, give a judgement (sure/impossible) if the answer is correct, i.e. it uses each input exactly once and no other numbers, and reach 24.

```
Input: 4 4 6 8
Answer: (4 + 8) * (6 - 4) = 24
```

```
Judge:
```

```
sure
```

```
...
```

```
Input: <INPUT>
Answer: <ANSWER>
Judge:
```

Die Modellantwort wird binär als „sure“ oder „impossible“ bewertet. Nur als *sure* eingestufte Lösungen werden akzeptiert und in die Ergebnisdarstellung aufgenommen.

3.4.4. Austausch der *Evaluate*-Metrik

Zielsetzung. Die bisherige *Evaluate*-Heuristik basierte auf festen Gewichtungen der Klassen „sure“, „likely“ und „impossible“ und erforderte pro vorgeschlagenem Schritt drei zusätzliche LLM-Prompts. Im überarbeiteten Verfahren entfällt der separate *Evaluate*-Prompt vollständig; die Bewertung erfolgt ausschliesslich anhand der Tokenwahrscheinlichkeiten des *Generate*-Prompts. Dadurch sinken sowohl das Prompt-Volumen als auch der Rechenaufwand.

Generate-Prompt mit Aufgabenfokus. Der *Generate*-Prompt wurde so angepasst, dass explizit ein *vielversprechender* nächster Schritt zur Lösung der 24-Aufgabe zu erzeugen ist. Die resultierenden Tokenwahrscheinlichkeiten spiegeln daher nicht nur die formale Korrektheit, sondern auch das modellinterne Lösungspotenzial wider.

Variante A — Einzelschritt

```
## TASK: You are an expert Game of 24 Solver.
Given remaining numbers, output exactly one valid operation in the format:
a [operation] b = result (left: remaining numbers)
Use only one of the operations (+, -, *, /) between exactly two numbers.
```

3. Methodik

Replace the used numbers with the result and list the new remaining numbers.
Always output a valid step even if not possible to reach 24
(only 2 numbers exist that can't be combined to 24).
If no more valid steps exist output "None".
No extra text or comments; do not explain.

Example:

Input: 2 8 8 14

Possible next step:

14 - 8 = 6 (left: 2 6 8)

Input: <INPUT>

Possible next step:

Variante B — Mehrfachschrift

TASK: Game of 24 Solver

Rules: Use each number exactly once; combine exactly two numbers per step using only one of the operations +, -, *, /.

No extra text or comments, just the "Possible next steps:" output as shown in the example.

Only include the next steps that are most likely to lead to a solution for Game of 24.

Example:

Input: 2 8 8 14

Possible next step:

2 + 8 = 10 (left: 8 10 14)

8 / 2 = 4 (left: 4 8 14)

14 + 2 = 16 (left: 8 8 16)

2 * 8 = 16 (left: 8 14 16)

8 - 2 = 6 (left: 6 8 14)

14 - 8 = 6 (left: 2 6 8)

14 / 2 = 7 (left: 7 8 8)

14 - 2 = 12 (left: 8 8 12)

Input: <INPUT>

Possible next step:

Score-Berechnung. Zur Bestimmung des Confidence-Scores wird das Verfahren *Probas-Mean* und *Probas-Min* verwendet.

- *Variante A*: Die Tokenwahrscheinlichkeiten des Outputs werden direkt zur Berechnung der Confidence benutzt.
- *Variante B*: Für jede ausgegebene Zeile wird separat eine Confidence anhand der zugehörigen Tokenwahrscheinlichkeiten bestimmt.

3. Methodik

Hinweis: Variante A Es wurden mehrere LLM-Anfragen ausgeführt, um dennoch mehrere mögliche Folgeschritte zu erhalten. Zur Erhöhung der Diversität in Variante A wurden Sampling-Temperaturen im Bereich $T \in [0,7, 2,0]$ evaluiert.

Kostenreduktion. Der Wegfall der *Evaluate*-Prompts reduziert das Gesamtaufkommen an API-Tokens. Da die Confidence unmittelbar aus dem *Generate*-Prompt abgeleitet wird, sinken die Inferenzkosten signifikant.

3.5. Auswertungsprotokoll

In diesem Abschnitt wird das methodische Vorgehen zur Bewertung der prognostizierten Unsicherheit beschrieben. Die Analyse orientiert sich an den in Abschnitt 2.5 definierten Messgrößen für Kalibration und Diskrimination und wird für fünf voneinander unabhängige Durchläufe (Runs 0–4) ausgeführt, um die Robustheit der Resultate abzuschätzen. Rohdaten der Modellantworten sowie sämtliche Skripte zur Berechnung und Visualisierung sind in einem separaten Repository (`Results_Analysis`) abgelegt. Dort befinden sich sowohl die JSON-Dateien mit den Modelloutputs als auch die entsprechenden Auswertungs- und Visualisierungsskripte.

3.5.1. Geltungsbereich und Datengrundlage

Die Analyse erstreckt sich auf die Experimente *Chain-of-Thought* (CoT) und *Chain-of-Draft* (CoD). Für *Tree-of-Thought* (ToT) werden lediglich die *Accuracy* sowie die auf Basis der Input- und Output-Tokens berechneten Kosten erhoben.

Für jeden Datensatz liegt eine Liste von Beispielen vor, die folgende Informationen enthält:

1. Ground-Truth-Antwort,
2. vom LLM generierte Antwort,
3. binäres Label zur Korrektheit (`True/False`),
4. Vektor von Konfidenzwerten pro Metrik und Datensatz.

Die Kennzahlen werden run-spezifisch berechnet und im Anschluss über alle Runs gemittelt, um die Robustheit der Resultate abzuschätzen.

3.5.2. Bewertungsmetriken

Die theoretischen Definitionen der Metriken finden sich in Abschnitt 2.5. Im Folgenden werden die verwendeten Kennzahlen, deren Parameter sowie die für die spätere Aggregation gespeicherten Größen festgehalten.

3. Methodik

Kalibration

Expected Calibration Error (ECE). Der Konfidenzbereich $[0, 1]$ wird in $K = 10$ äquidistante Bins unterteilt. Für jedes Bin werden gespeichert: durchschnittliche Confidence (\bar{c}_k), durchschnittliche Accuracy (a_k) sowie die Bin-Grösse (n_k). Diese Werte dienen zugleich als Stützpunkte für die Reliability Curve.

Reliability Curve. Es werden ausschliesslich die im Rahmen des ECE berechneten Bin-Paare (\bar{c}_k, a_k) visualisiert; zusätzliche Hyperparameter sind nicht erforderlich.

Diskrimination

Area Under the ROC Curve (AUROC). Für die ROC-Kurve werden 100 Stützpunkte interpoliert. Das 95 %-Konfidenzintervall des AUROC wird mittels t-Verteilung bestimmt. Pro Run wird der Skalarwert gespeichert.

Zusatzmetrik

Accuracy wird (für CoT, CoD und ToT) als Kennzahl ausgewiesen.

Schnittstelle zur Run-Aggregation In die Aggregation gehen pro Run die Skalarwerte von ECE, AUROC und Accuracy ein. Für den ECE werden zusätzlich die bin-weisen Arrays (Confidence, Accuracy, Bin-Grössen) übernommen, um aggregierte Reliability Curves zu erstellen. Ein Pooling aller Beispiele über Runs hinweg wird nicht vorgenommen.

Parameterübersicht (Defaults)

- Anzahl Bins (ECE): **10**
- ROC-Interpolation: **100** Punkte
- Konfidenzintervall: **0.95**
- Weitere Standardparameter (z. B. Pfade, Konfigurationen) sind in den YAML-Dateien im Repository **Results_Analysis** hinterlegt.

3.5.3. Workflow und Visualisierung

Der Ablauf ist zweistufig:

1. Run-weise Berechnung der Metriken und Speicherung der Resultate,
2. Aggregation der Skalarwerte über die Runs.

3. Methodik

3.5.4. Experimentsspezifische Unterschiede

- **CoT / CoD:** Vollständige Auswertung gemäss Abschnitt 3.5 (Kalibration, Diskrimination, Zusatzmetriken). Accuracy wird zusätzlich ausgewiesen.
- **ToT:** Reduktion auf Accuracy und Kosten. Die übrigen Unsicherheitsmetriken werden nicht berechnet.

3.6. Reproduzierbarkeit

Der komplette Code, Konfigurationen, Abhängigkeiten, detaillierte Anleitung zur Ausführung (README.md) und Rohdaten (SLURM-Outputs) sind im GitHub-Repository verfügbar [28]. Für alle Experimente wurden keine Seeds erzwungen.

Repos und Commit:

- `CoT-UQ` (CoT/CoD-Pipeline)
- `tree-of-thought-llm` (ToT-Experimente)
- `Results_Analysis` (Auswertung, Plots, Tabellen)

Zitierte Version: Commit `2584771` vom 10. August 2025.

Für `CoT-UQ` sind die zentralen Umgebungsdateien im Anhang wiedergegeben (Requirements, Listing A.1; Dockerfile, Listing A.3).

4. Experimente und Ergebnisse

4.1. Quantitative Resultate

4.1.1. Accuracy

CoT vs. CoD Accuracy

Die Accuracy ist bei den mathematischen Datensätzen (GSM8K, SVAMP, ASDiv) mit Werten zwischen etwa 40 % und 65 % deutlich höher als bei den logischen Multi-Hop-Aufgaben (2WikiMHQA, HotpotQA). Besonders niedrig fällt die Accuracy bei 2WikiMHQA mit nur 6 % für CoT bzw. 3 % für CoD aus. Im Durchschnitt sinkt die Accuracy bei Verwendung von CoD um 2,84 Prozentpunkte, mit Ausnahme des ASDiv-Datensatzes, wo CoD leicht überlegen ist. Detaillierte Statistiken (Mittelwert, Standardabweichung, Min/Max und 95 %-Konfidenzintervalle) sind im Anhang Abschnitt A.5 in Tabelle A.2 (CoT) und Tabelle A.3 (CoD) aufgeführt.

Tabelle 4.1.: Durchschnittliche Accuracy (5 Läufe) von CoT und CoD; Δ entspricht der Differenz (CoT–CoD).

Datensatz	CoT (%)	CoD (%)	Δ
2WikiMHQA	6.10	3.23	2.87
HotpotQA	26.08	21.02	5.06
GSM8K	41.90	37.91	3.99
SVAMP	60.73	56.73	4.00
ASDiv	64.59	66.29	-1.70
\emptyset			2.84

Accuracy der ToT-Baselines

Tabelle 4.2 vergleicht die Erfolgsraten der ToT-Baseline-Varianten mit heuristischer Evaluierung (vgl. Unterunterabschnitt 3.4.2) zwischen GPT-4 und DeepSeek-V3-Chat. Die CoT-Erfolgsrate dient als Referenz ohne ToT(BFS)-Suche und ohne heuristische Evaluierung. DeepSeek erzielt im Schnitt leicht bessere Ergebnisse.

4. Experimente und Ergebnisse

Tabelle 4.2.: Erfolgsraten auf dem *Game-of-24*-Benchmark – Vergleich GPT-4 und DeepSeek (ToT-Baselines).

Variante / Modell	Beam-Größe b	Erfolgsrate [%]	Quelle
CoT (GPT-4)	–	4	[4]
ToT+BFS (GPT-4)	1	45	[4]
ToT+BFS (GPT-4)	5	69–74	[4, 11]
ToT+BFS (DeepSeek)	1	54	Eigene Messung
ToT+BFS (DeepSeek)	5	72	Eigene Messung

Accuracy der unsicherheitsbasierten ToT-Varianten

Tabelle 4.3 fasst die Accuracy der ToT-Varianten mit tokenwahrscheinlichkeitsbasierter Evaluierung (vgl. Unterabschnitt 3.4.4) zusammen. Hierbei werden die beiden Antwort-Generierungsstrategien verglichen: Multiple-Solution (Variante B) und Single-Solution (Variante A, vgl. Unterabschnitt 3.4.4). Die Multiple-Solution-Variante erzielt mit Probas-Mean eine sehr niedrige Accuracy von nur 3 %. Im Gegensatz dazu performt die Single-Solution-Variante deutlich besser, mit einer maximalen Accuracy von 24 % bei einer Temperatur von 1,8 für Probas-Mean und 18 % bei 1,5 für Probas-Min. Insgesamt schneidet Probas-Mean in der Single-Solution-Variante besser ab.

Tabelle 4.3.: Accuracy der unsicherheitsbasierten ToT-Varianten bei $b = 5$

(a) Multiple-Solution, Probas-Mean		(b) Single-Solution, Probas-Mean		(c) Single-Solution, Probas-Min	
Temp.	Accuracy	Temp.	Accuracy	Temp.	Accuracy
0.7	3 %	0.7	14 %	1.3	16 %
		1.3	16 %	1.5	18 %
		1.5	18 %	1.8	16 %
		1.8	24 %	2.0	17 %
		2.0			

4.1.2. Diskriminationsfähigkeit

Die Diskriminationsfähigkeit wird anhand der AUROC-Werte bewertet, die die Trennschärfe zwischen korrekten und fehlerhaften Antworten messen. Die folgenden Abbildungen präsentieren AUROC-Leaderboards, die die Leistung verschiedener Unsicherheitsmetriken vergleichen (siehe Unterabschnitt 3.3.3). Die Metriken sind farblich unterteilt: Blau für Baseline-Aggregationsmetriken (basierend auf Tokenwahrscheinlichkeiten der finalen Antwort), Gelb für Keyword-Varianten (mit Extraction und Importance Scoring aus der Reasoning-Chain), Grün für P-True-Varianten und Rot für Self-Probing-Varianten. Auf der x-Achse sind die Untervarianten dargestellt, z. B. Probas-Mean, Probas-Min und

4. Experimente und Ergebnisse

Token-SAR für Aggregationsmetriken sowie Baseline, Allsteps, Keystep, Allkeywords und Keykeywords für Self-Evaluation-Ansätze. Die Leaderboards umfassen die Streuung über fünf Läufe (Violinplots) sowie eine horizontale Linie bei 0,5 zur Markierung zufälliger Diskrimination.

CoT

Abbildung 4.1 zeigt das AUROC-Leaderboard für das CoT-Experiment, einschließlich Streuung über die Läufe und Vergleich mit dem CoT-UQ-Paper (weiße Diamanten) [16]. Fehlende Diamanten für SVAMP (Self-Probing) und ASDiv (P-True) wurden im Referenzpaper nicht angegeben. Die Varianz ist bei 2WikiMHQA besonders hoch, mit starken Abweichungen zu den Referenzergebnissen bei HotpotQA (P-True) und SVAMP (P-True). Keyword-Varianten (mit Extraction und Importance Scoring) performen insgesamt am besten, wobei Token-SAR bei mathematischen Datensätzen überlegen ist, gefolgt von Probas-Min und Probas-Mean. Bei Baseline-Aggregationsmetriken fällt Token-SAR hinter Probas-Min und Probas-Mean zurück. Self-Probing und P-True schneiden bei logischen Datensätzen besser ab als bei mathematischen, wo sie oft nahe oder unter 0,5 liegen; bei logischen Datensätzen ist P-True überlegen, bei mathematischen Self-Probing.

4. Experimente und Ergebnisse

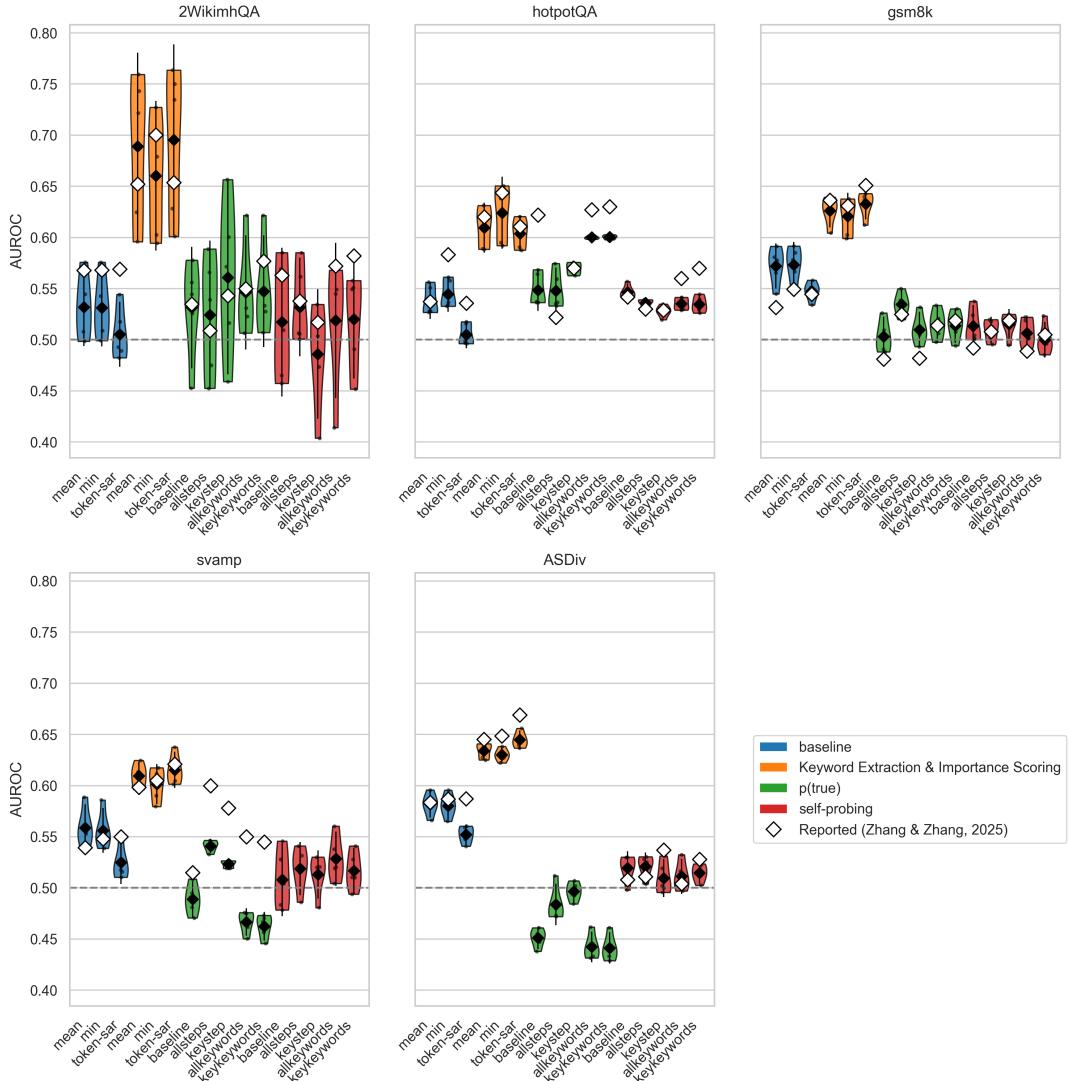


Abbildung 4.1.: AUROC-Leaderboard für das CoT-Experiment und Vergleich mit CoT-UQ-Paper-Ergebnissen. Höhere Werte sind besser.

CoD

Abbildung 4.2 präsentiert das AUROC-Leaderboard für das CoD-Experiment. Die Streuung ist bei 2WikiMHQA noch höher als im CoT-Fall. Baseline-Metriken (Probas-Mean und Probas-Min) schneiden bei HotpotQA und GSM8K besser ab als Keyword-Varianten. Bei logischen Datensätzen ist die Keyword-Variante im Vergleich zum CoT-Experiment deutlich schlechter. Bei den mathematischen Datensätzen bleibt die Keyword-Variante jedoch am besten, mit Token-SAR als Leader (Ausser GSM8K). Self-Probing performt bei mathematischen Datensätzen besser als P-True, oft nahe 0,5 oder darunter. Die

4. Experimente und Ergebnisse

Self-Evaluation-Methoden P-True und Self-Probing sind insgesamt bei den logischen Datensätzen besser (im Vergleich zu den Mathematischen Datensätzen), analog zum CoT-Experiment.

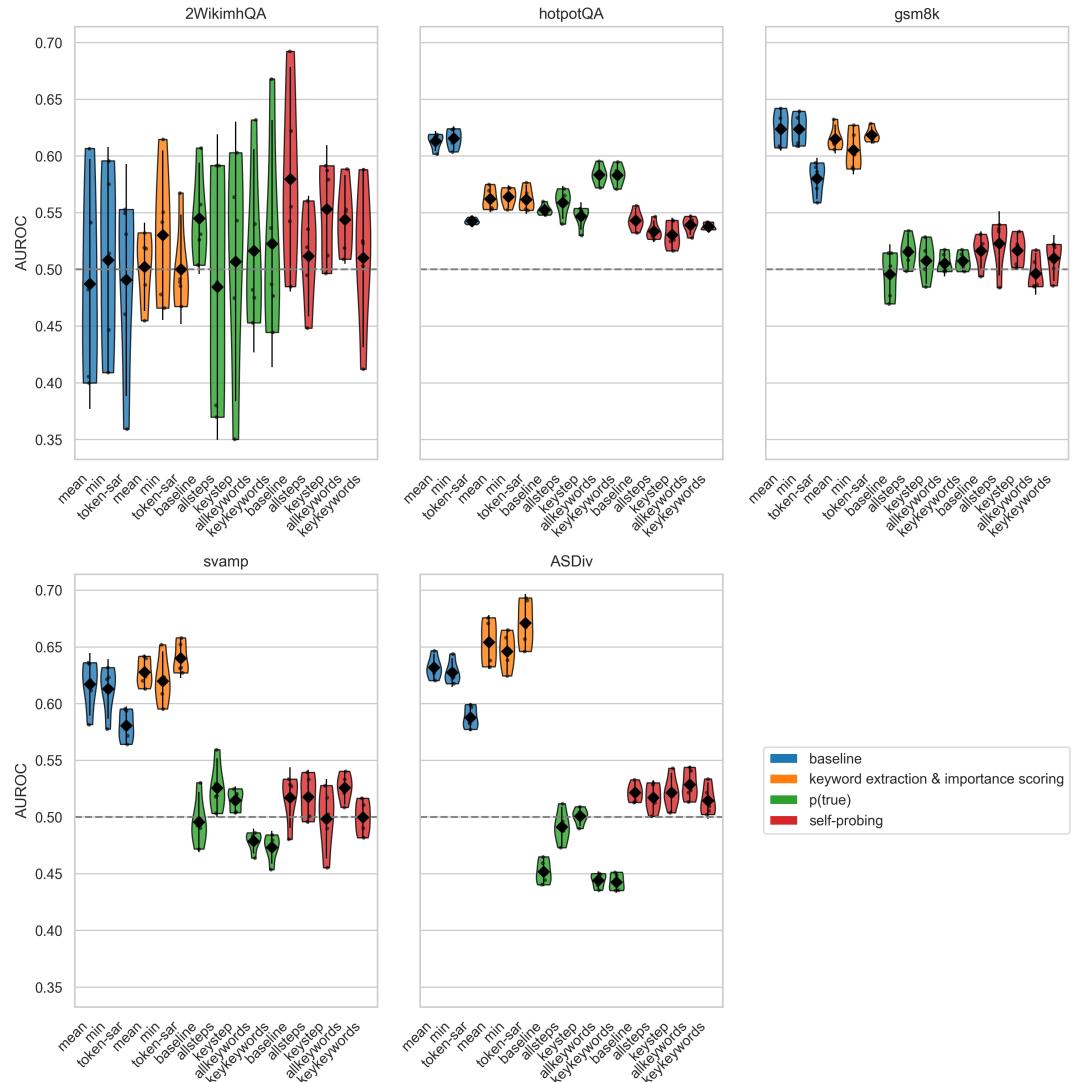


Abbildung 4.2.: AUROC Leaderboard für das CoD Experiment. Höhere Werte sind besser.

Vergleich CoD vs. CoT

Abbildung 4.3 vergleicht die drei Baseline-Varianten der Aggregationsmetriken zwischen CoT und CoD. Abgesehen vom 2WikiMHQA-Datensatz schneiden die Baseline-Metriken für CoD deutlich besser ab.

4. Experimente und Ergebnisse

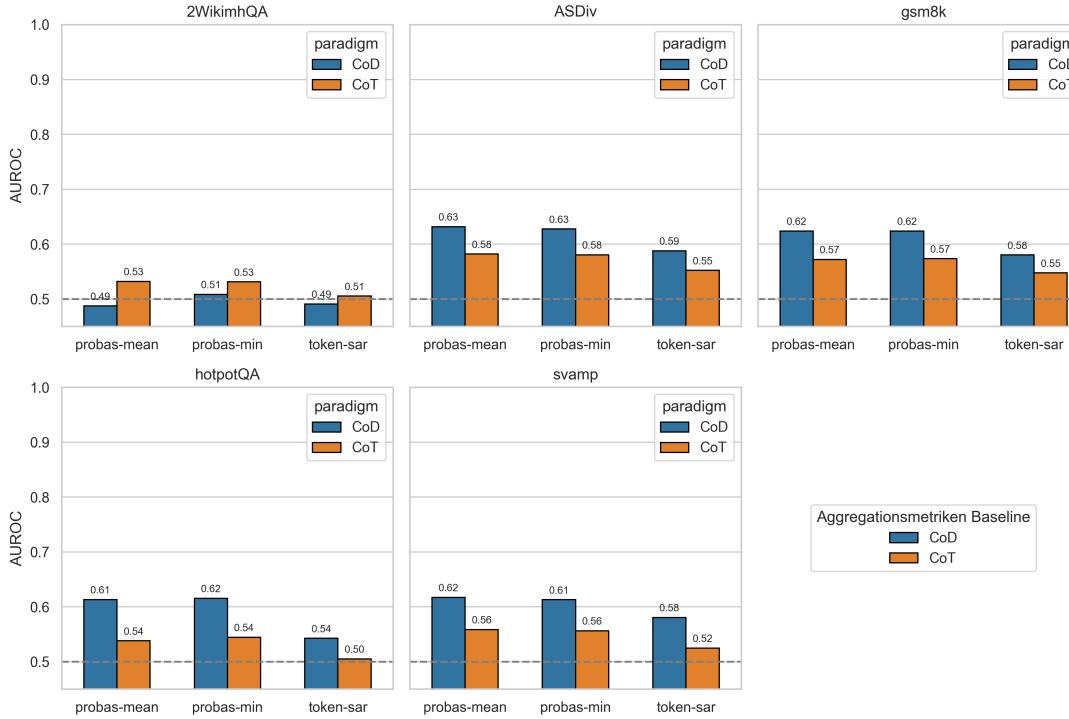


Abbildung 4.3.: Vergleich Baseline Aggregationsmetriken CoD vs. CoT. Höhere Werte sind besser.

4.1.3. Kalibrationsergebnisse

Die Kalibrationsergebnisse werden anhand des Expected Calibration Error (ECE) und Reliability Curves bewertet. Die ECE-Leaderboards (farblich codiert analog zu den AUROC-Plots: Blau für Baseline-Aggregationsmetriken, Gelb für Keyword-Varianten, Grün für P-True und Rot für Self-Probing) ordnen die Datensätze nach aufsteigender Accuracy. Die Reliability Curves visualisieren die Übereinstimmung zwischen prognostizierter Confidence und beobachteter Accuracy. Bei den Reliability Curves dient der Datensatz ASDiv als repräsentatives Beispiel, da ähnliche Trends über alle Datensätze und Strategien (CoT/CoD) hinweg beobachtet werden. Ergänzend berichten wir den Brier Score, eine skalare Fehlermetrik, die auf Fehlkalibration (Über-/Unterkonfidenz) reagiert (Tabelle A.4, Tabelle A.5; niedriger ist besser). Dort schneiden Keyword-Varianten auch konsistent besser ab als die dazugehörigen Baselines.

ECE-Leaderboards für CoT und CoD

Abbildung 4.4 und 4.5 zeigen die ECE-Leaderboards für CoT und CoD. In beiden Experimenten sind die Baseline-Varianten schlechter kalibriert als die Keyword-Varianten (mit Extraction und Importance Scoring). Probas-Min erzielt die beste Kalibration bei den Keyword-Varianten und meist auch bei den Baselines (Ausnahmen: SVAMP und

4. Experimente und Ergebnisse

GSM8K, wo Probas-Mean überlegen ist). P-True korreliert stark mit der Accuracy: Höhere Accuracy führt zu niedrigeren ECE-Werten (mehr dazu bei den Reliability Curves). Es gibt keine signifikanten Unterschiede in der Kalibrationsverteilung zwischen CoT und CoD.

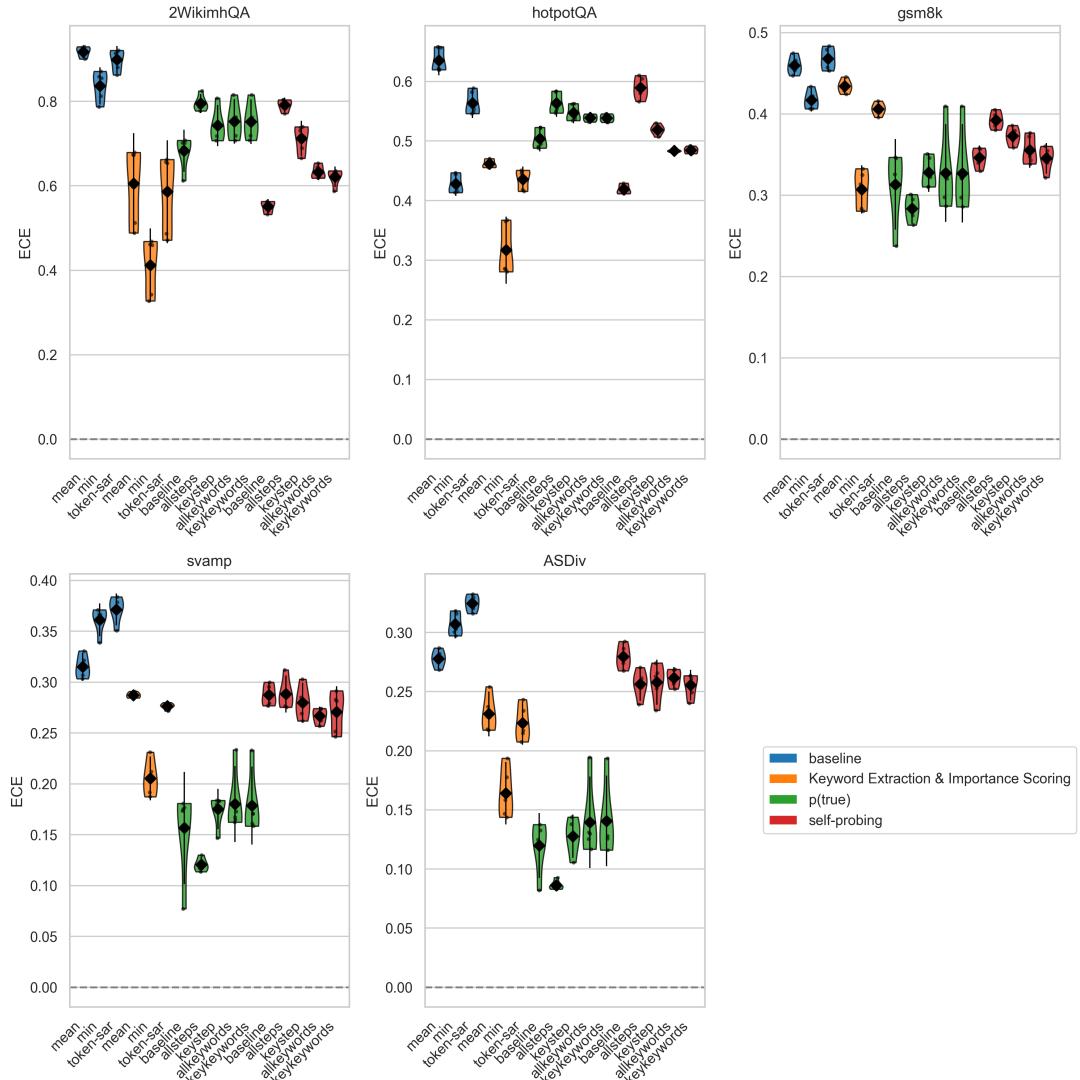


Abbildung 4.4.: ECE-Leaderboard für das CoT-Experiment. Niedrigere Werte sind besser.

4. Experimente und Ergebnisse

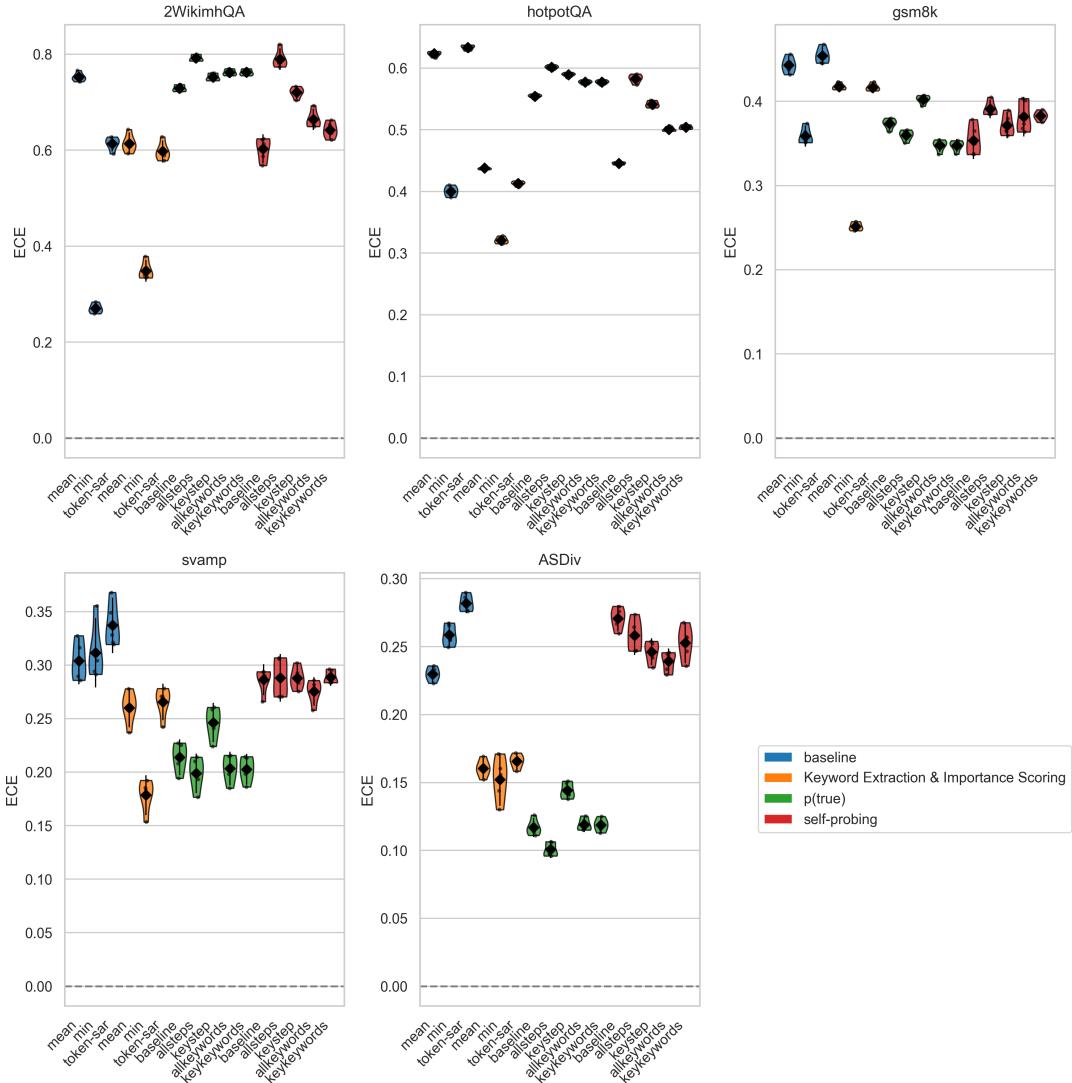


Abbildung 4.5.: ECE-Leaderboard für das CoD-Experiment. Niedrigere Werte sind besser.

Reliability Curves

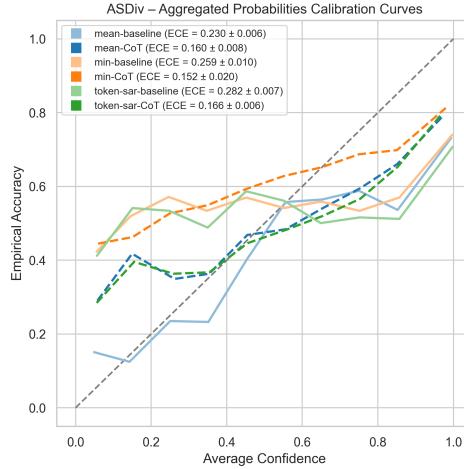
Abbildung 4.6 vergleicht Reliability Curves für das CoD-Experiment am Beispiel des ASDiv-Datensatzes, der repräsentativ für alle Datensätze ist und ähnliche Trends wie im CoT-Experiment zeigt. Vollständige Plots der anderen Datensätze sowie für CoT sind im Anhang Abschnitt A.4 zu finden. Die gestrichelte Diagonale stellt perfekte Kalibration dar. Im Folgenden werden die drei Plots separat interpretiert:

- Aggregationsmetriken:** Die gestrichelten, dunklen, gestrichelten Linien (Keyword-Varianten) nähern sich der Diagonalen besser an als die Baselines, was eine verbesserte Kalibration bei allen Aggregationsmetriken (Probas-Min, Probas-Mean,

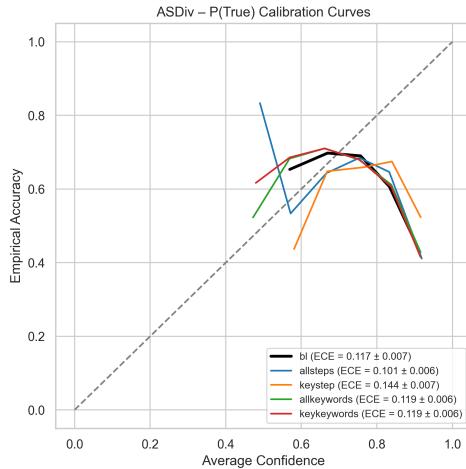
4. Experimente und Ergebnisse

Token-SAR) und Datensätzen unterstreicht.

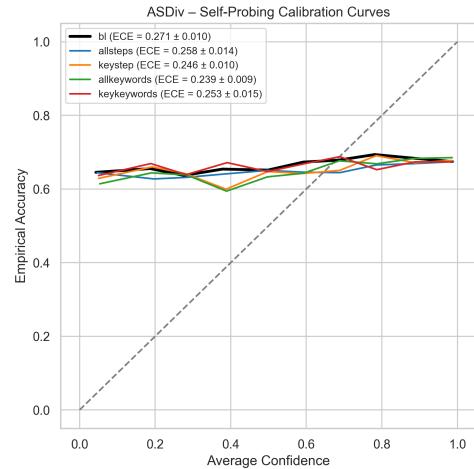
- (b) **P_{true} – Strategien:** P-True weist einen engen Confidence-Bereich (ca. 0,4–0,9) auf, was die Korrelation von ECE mit Accuracy erklärt; Unterschiede zwischen Varianten (z. B. Allsteps, Keystep) sind minimal.
- (c) **Self-Probing-Strategien:** Self-Probing ergibt eine nahezu horizontale Linie, was auf fehlende Korrelation hinweist und das Signal unzuverlässig für die Trennung korrekter und fehlerhafter Antworten macht.



(a) Aggregated Probabilities



(b) P_{true} – Strategien



(c) Self-Probing-Strategien

Abbildung 4.6.: Reliability-Vergleich für das CoD-Experiment (ASDiv als Beispiel): (a) Aggregated Probabilities, (b) P_{true}, (c) Self-Probing.

4. Experimente und Ergebnisse

AUROC vs. ECE Scatter

Abbildung 4.7 zeigt Scatter-Plots für das CoT-Experiment (Aggregationsmetriken). Die Keyword-Varianten (dunkel) verbessern sowohl AUROC als auch ECE im Vergleich zu Baselines (hell). Es bilden sich deshalb zwei klare Cluster über alle Metriken (Probas-Min, Probas-Mean, Token-SAR) und Datensätze hinweg.

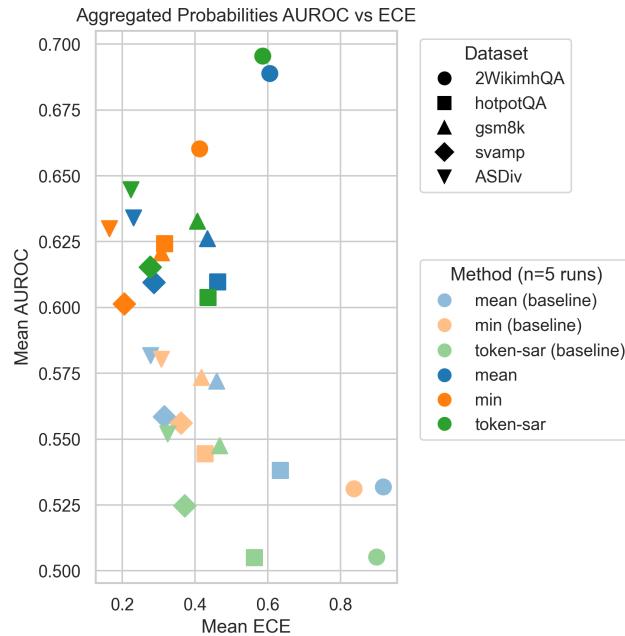


Abbildung 4.7.: AUROC vs. ECE Scatter für das CoT-Experiment (Aggregationsmetriken).

4. Experimente und Ergebnisse

4.2. Laufzeit- und Kostenanalyse

4.2.1. CoT / CoD – LLaMA-3 (Cluster)

Tabelle 4.4.: Vergleich der durchschnittlichen Laufzeit pro Run für CoT- und CoD-Experimente

Datensatz	CoT / Run [h]	CoD / Run [h]
ASDiv	19.57	18.04
2WikiMHQA	47.51	36.88
SVAMP	8.42	8.02
GSM8K	17.08	11.87
HotpotQA	86.99	123.62 ¹
<i>Summenangaben in Stunden (h) und Tagen (d)</i>		
Summe	179.57 h (7.48 d)	198.43 h (8.27 d)
Summe ($\times 5$ Runs)	897.85 h (37.41 d)	992.15 h (41.34 d)

4.2.2. GPT-4o-mini API – Kostenübersicht

Für den automatisierten Vergleich von Modellantworten mit den Ground-Truth-Lösungen kam die GPT-4o-mini-API zum Einsatz. Seit Projektbeginn sind dabei Gesamtkosten von **\$1.49** entstanden, aufgeschlüsselt wie folgt:

Tabelle 4.5.: Kumulierte GPT-4o-mini-API-Nutzung
(Tarif: \$0.15 / 10^6 Input-Tokens, \$0.60 / 10^6 Output-Tokens; Stand Juli 2025)

USD	
Input-Tokens	1.45
Output-Tokens	0.04
Summe	1.49

Somit sind die API-Kosten im Vergleich zu den Aufwendungen für die Nutzung des Rechenclusters (insbesondere Stromkosten) deutlich geringer und spielen folglich nur eine untergeordnete Rolle bei den Gesamtkosten der CoT- und CoD-Experimente.

¹Es wird vermutet, dass HotpotQA-CoD langsamer ist, weil diese Jobs nur mit *zwei* statt acht CPU-WorkerThreads liefen und dadurch ein I/O-Flaschenhals entstand. Kleinere Datensätze waren nicht betroffen, da sie deutlich weniger Batches enthalten.

4. Experimente und Ergebnisse

4.2.3. Kostenvergleich der ToT-Experimente

Die API-Kosten der durchgeföhrten Experimente sind in Tabelle 4.6 und Tabelle 4.7 zusammengefasst. Besonders auffällig ist, dass der Modellwechsel von GPT-4 zu *DeepSeek* eine Kostensenkung von etwa 98.6 % ermöglichte. Während die ToT + BFS-Experimente mit GPT-4 laut [4] 74 USD kosteten, beliefen sich die Kosten mit *DeepSeek* für dieselbe Konfiguration ($b = 5, T = 0.7$) auf lediglich 1.07 USD.

Des Weiteren zeigten die unsicherheitsbasierten Varianten (*Probas-Mean* und *Probas-Min*) eine zusätzliche Kostenreduktion gegenüber der *DeepSeek*-Baseline. Die Varianten *Probas-Mean – Multiple Solutions* (0.20 USD), *Probas-Mean – Single Solution* (0.33 USD pro Durchlauf) und *Probas-Min – Single Solution* (0.35 USD pro Durchlauf) verursachten etwa drei- bis fünfmal geringere Kosten im Vergleich zur Baseline mit 1.07 USD. Somit stellen die unsicherheitsbasierten Varianten die kosteneffizientesten Optionen dar.

Tabelle 4.6.: Kosten des ToT-Experiments (DeepSeek-API).

Run / Setting	b	T	Kosten [USD]
ToT+BFS baseline	1	0.7	0.59
ToT+BFS baseline	5	0.7	1.07
ToT+BFS Probas-Mean – Multiple Solutions	5	0.7	0.20
ToT+BFS Probas-Mean – Single Solution	5	0.7, 1.3, 1.5, 1.8, 2.0	0.33 ($\times 5$: 1.65)
ToT+BFS Probas-Min – Single Solution	5	1.3, 1.5, 1.7	0.35 ($\times 3$: 1.05)
Gesamtsumme			4.56

Tabelle 4.7.: Kosten des ToT-Experiments mit GPT-4 laut [4].

Run / Setting	b	T	Kosten [USD]	Quelle
CoT (GPT-4)	–	–	4.70	[4]
ToT+BFS (GPT-4)	5	–	74.00	[4]

4.3. Qualitative Resultate

4.3.1. Fallstudien: *Chain-of-Draft* (CoD)

Um die quantitativen Erkenntnisse aus Abschnitt 4.2 zu veranschaulichen, werden hier *zwei zufällig ausgewählte Chain-of-Draft*-Beispiele (CoD) analysiert: eines mit korrekter und eines mit fehlerhafter Antwort.

Fallstudie 1: Korrekte Antwort mit hoher Konsistenz (CoD, ASDiv, ID: nluds-0038)

Frage: "Mrs. Hilt reads 5 books a day. How many books does she read in 3 days?"

4. Experimente und Ergebnisse

Ground-Truth-Antwort: 15

LLM-Antwort: Step 1: Daily books read: 5

Step 2: Books in 3 days: $3 \times 5 = 15$

Final Answer: 15

Tabelle 4.8.: Reasoning-Kette, extrahierte Keywords und Token-Wahrscheinlichkeiten (Fallstudie 1)

Reasoning-Schritt	Keywords (Importance)	p (Token)
Step 1: Daily books read: 5	"5" (8)	1.0
Step 2: Books in 3 days: $3 \times 5 = 15$	"15" (6)	1.0
Final Answer: 15	- (Baseline)	1.0

Analyse. Das Modell erstellt eine konsistente Chain-of-Draft; alle Token-Wahrscheinlichkeiten sind maximal ($p = 1.0$). Dadurch stimmen Baseline- und Keyword-*Probas-Min* überein und zeigen eine perfekte Kalibration.

Fallstudie 2: Fehlerhafte Antwort mit Overconfidence (CoD, ASDiv, ID: nluds-0039)

Frage: "Mrs. Hilt saw 2 dogs and 2 chickens. How many animal legs did she see?"

Ground-Truth-Antwort: 12

LLM-Antwort (gekürzt): Step 1: 2 dogs + 2 chickens

Step 2: 4 legs each

Step 3: 4 + 4

Step 4: 8

Final Answer: 8

Tabelle 4.9.: Reasoning-Kette, extrahierte Keywords und Token-Wahrscheinlichkeiten (Fallstudie 2)

Reasoning-Schritt	Keywords (Importance)	p (Token)
Step 1: 2 dogs + 2 chickens	"2 dogs" (10), "2 chickens" (10)	0.924; 0.963
Step 2: 4 legs each	"4 legs each" (10)	0.706; 0.695; 0.017
Step 3: 4 + 4	"4 + 4" (10)	0.604; 0.293; 1.0; 1.0
Step 4: 8	"8" (10)	1.0
Final Answer: 8	- (Baseline)	1.0

Analyse. Das Modell halluziniert, dass Hühner vier Beine haben. Obwohl die finale Antwort falsch ist, signalisieren Baseline-Metriken volle Sicherheit (*Probas-Min* = 1.0). Die Keyword-Variante deckt jedoch Unsicherheit auf ($p_{\min} = 0.017$ für "4 legs each").

4.3.2. Stichprobenprüfung: String-Matching-Verifier (ASDiv) und LLM-Judge (HotpotQA)

Stichprobe. Für den *String-Matching-Verifier* wurde aus *ASDiv* eine zufällige, label-balancierte Stichprobe von $n = 10$ Instanzen gezogen (je fünf `true`- und fünf `false`-

4. Experimente und Ergebnisse

Labels). Für den *LLM-Judge* wurde analog aus *HotpotQA* eine zufällige, label-balancierte Stichprobe von $n = 10$ Instanzen erstellt.

Ergebnisse. In beiden Stichproben traten *keine* Abweichungen zwischen Verifier/Judge und dem manuellen Urteil auf (*ASDiv*: 0 von 10; *HotpotQA*: 0 von 10).

Zusätzliche Befunde (unsystematisch). Bei einer weiterführenden, *nicht* systematischen Durchsicht wurden folgende Punkte beobachtet:

- **String-Matching-Probleme (ASDiv).** Eine semantisch korrekte Antwort wurde fälschlicherweise als inkorrekt markiert: Zahlenformatierung 14280.0 vs. 14,280.00.
- **Dataset-Fehler (HotpotQA).** Inkorrekt Ground-Truth-Eintrag (der „Paul Sloane / 9 Emmys“-Eintrag).

4.4. Zusammenfassung der Ergebnisse

Genauigkeit *Chain-of-Thought* (CoT) übertrifft *Chain-of-Draft* (CoD) im Mittel um 2.84 Prozentpunkte; *ASDiv* bildet eine Ausnahme zugunsten von CoD. Logische Multi-Hop-Datensätze (*2WikiMHQA*, *HotpotQA*) erweisen sich insgesamt als deutlich schwieriger als mathematische (*GSM8K*, *SVAMP*, *ASDiv*).

Diskrimination (AUROC) Keyword-gestützte Aggregationsmetriken (insbesondere Token-SAR) erzielen über Datensätze hinweg höhere AUROC-Werte als Baselines, die ausschließlich die finale Antwort berücksichtigen. *Self-Probing* und *P-True* funktionieren bei logischen Aufgaben tendenziell besser als bei mathematischen; bei letzteren liegen sie nahe 0.5. Mit CoD bleiben die relativen Trends erhalten; Die CoD Baseline-Aggregationen schneiden durchgehend besser ab als bei CoT (ausser *2WikiMHQA*).

Kalibration (ECE, Reliability) Keyword-gestützte Aggregationsmetriken sind konsistent besser kalibriert als die korrespondierenden Baselines (niedrigere ECE, Kurven näher an der Diagonalen). *Probas-Min* erzielt häufig die beste Kalibration; bei *SVAMP* und *GSM8K* übertrifft teils *Probas-Mean*. *P-True* weist einen engen Konfidenzbereich (ca. 0.4–0.9) auf, wodurch ECE stark mit der Aufgaben-Genauigkeit korreliert. *Self-Probing* ist in mehreren Datensätzen kaum informativ (nahezu horizontale Reliability-Kurven).

ToT (Game of 24) Die Baseline-Reproduktion mit DeepSeek erreicht Genauigkeiten, die vergleichbar oder etwas höher als die publizierte GPT-4-Baseline sind, jedoch bei drastisch niedrigeren Kosten ($\approx 99\%$ Kostensenkung). Der Ersatz der **Evaluate**-Heuristik durch tokenwahrscheinlichkeitsbasierte Scores reduziert die Kosten weiter, führt jedoch zu deutlichen Genauigkeitseinbussen (bis auf 3–24%). Die Single-Solution Variante schneidet deutlich besser ab als Multiple-Solution; insgesamt bleibt die Heuristik klar überlegen.

4. Experimente und Ergebnisse

Laufzeit und Kosten CoD spart gegenüber CoT sowohl Tokens als auch Zeit, allerdings nicht durchgängig (z. B. I/O-Flaschenhals bei HotpotQA-CoD). Die GPT-4o-mini-Kosten für das Grading sind vernachlässigbar. Für ToT gilt: DeepSeek-basierte Runs sind $3\text{--}5\times$ günstiger als die eigene DeepSeek-Baseline, wenn UQ-Scores anstelle der Heuristik genutzt werden – allerdings bei deutlicher Genauigkeitseinbusse.

5. Diskussion

5.1. Interpretation der Hauptergebnisse

Die empirischen Resultate lassen sich in vier zentrale Befunde verdichten.

- **Semantisch gewichtete Aggregation entlang der Reasoning-Kette verbessert die Unsicherheitsabschätzung konsistent.** Über alle Datensätze hinweg erhöhen relevanz- bzw. keyword-gestützte Aggregationsmetriken die Diskrimination (AUROC) und senken den Kalibrationsfehler (ECE) gegenüber Baselines, die ausschliesslich finale Antworttokens aggregieren (vgl. Abbildung 4.1, Abbildung 4.2, Abbildung 4.4, Abbildung 4.5; AUROC–ECE-Scatter: Abbildung 4.7). *Probas-Min* und *Token-SAR* zeigen die stabilsten Zugewinne. Die Wirkmechanismen lassen sich dreifach gliedern:
 - *Keywords = Relevanzselektion (Längen-/Formatbias)*. Keywords definieren die Menge semantisch tragender Tokens und filtern Format-, Füll- und Gliederungselemente heraus. Dadurch sinkt die Empfindlichkeit gegenüber Antwortlänge und formaler Homogenität; die Aggregation bezieht sich primär auf inhaltstragende Segmente.
 - *Importance Scores = Gewichtung (Entscheidungsrelevanz)*. Importance Scores differenzieren innerhalb der relevanten Menge und verstärken den Einfluss entscheidungsrelevanter Tokens (z. B. Schlussfolgerungswörter, numerische Knergebnisse, Entitäten). Dies schärft die Aggregation: niedrige Wahrscheinlichkeiten an kritischen Stellen drücken den Gesamtscore stärker, während irrelevante Schwankungen gedämpft werden. Der Effekt ist aber nur sichtbar bei *Token-SAR* und *Probas-Mean*; *Probas-Min* wird hingegen primär durch die Keyword-Selektion bestimmt.
 - *Reasoning-Chain als Evidenzträger (Früherkennung und Kalibration)*. Die Schritt-für-Schritt-Verteilung der Token-Wahrscheinlichkeiten entlang der Begründungskette ermöglicht die Früherkennung von Begründungsfehlern und relativiert deterministisch hohe finale Token-Wahrscheinlichkeiten. Die Fallstudien in Tabelle 4.8 und Tabelle 4.9 illustrieren, dass *Probas-Min* lokale Fehlstellen früh sichtbar macht (z. B. das Token „4 legs“ bei Hühnern), die bei ausschliesslicher Betrachtung der finalen Antwort verborgen bleiben. Dies geht mit niedrigeren ECE-Werten und konsistenten AUROC-Gewinnen in CoT und CoD einher.
- **CoD konserviert die UQ-Qualität bei deutlich schlankerer Ausführung.** Trotz kompakterer Reasoning-Ketten verlaufen die AUROC- und ECE-Trends

5. Diskussion

weitgehend parallel zu CoT. Die mittlere Genauigkeit sinkt moderat ($-2,84$ Prozentpunkte), während die Unsicherheitssignale stabil bleiben. CoD ist damit attraktiv, wenn Latenz und Token-Budget kritisch sind. Der beobachtete Laufzeiteffekt zugunsten von CoD wurde auf Pipeline-Ebene gemessen (inkl. Extraktion, Self-Evaluation etc.) und ist daher nicht ausschliesslich der Inferenz zuzuordnen; er stützt jedoch die Grundannahme geringerer Kosten und Latenz in der Praxis..

- **Tokenwahrscheinlichkeitsbasierte Scoring-Ersatzverfahren für ToT-BFS sind kostengünstig, aber deutlich weniger treffsicher.** Der Ersatz der publizierten `evaluate`-Heuristik durch *Probas-Mean*/*Probas-Min* reduziert die Kosten gegenüber der DeepSeek-Baseline, führt jedoch zu substanziellem Genauigkeitseinbussen (Erfolgsraten ca. 24 % vs. 72 % bei $b=5$; vgl. Tabelle 4.3, Tabelle 4.2). Hauptursachen sind:
 - *Formatartefakte*. Tokenwahrscheinlichkeiten ganzer, formatierter Schrittzeilen werden durch Ziffern, Klammern oder wiederkehrende Muster (z. B. „left：“) dominiert und spiegeln den semantischen Beitrag eines Schritts zum Ziel (Game of 24) unzureichend.
 - *Konkurrenz in Mehrfachlisten*. Bei Multiple-Solution-Generierung beeinflussen sich Tokenwahrscheinlichkeiten der Vorschläge gegenseitig. Die Single-Solution-Variante mildert diese Interferenz sichtbar und erzielt deutlich bessere Resultate. Insgesamt bleibt die `Evaluate`-Heuristik im ToT-Setting (BFS) klar überlegen.
 - *Modell- und Prompt-Sensitivität (Few-shot-Regelbefolgung)*. In den vorliegenden Versuchen zeigte DeepSeek im Vergleich zu *GPT-4* eine geringere Robustheit gegenüber wenigen Beispielen und eine stärkere Tendenz zu verboseren bzw. formatvarianten Ausgaben, sodass zusätzliche Instruktionen zur Verbositätskontrolle und Strukturierung erforderlich waren. Dies könnte die Wirksamkeit rein tokenwahrscheinlichkeitsbasierter Scores weiter mindern.
- **Zusätzliche Beobachtungen mit praktischer Relevanz.**
 - *Final-Answer-Baselines profitieren unter CoD häufiger als unter CoT*. Baseline-Aggregationsmetriken (nur finale Antworttokens) schneiden unter CoD meist besser ab als unter CoT (Abbildung 4.3). Eine plausible Erklärung ist *Confidence-Inflation* durch lange, formal homogene CoT-Ketten: Ausgedehnte, standardisierte Schrittfolgen konditionieren den Decoder auf kanonische Final-Answer-Formulierungen und erhöhen die Wahrscheinlichkeiten finaler Tokens weitgehend unabhängig von der inhaltlichen Korrektheit. Kürzere CoD-Ketten begrenzen diesen Effekt und reduzieren Überkonfidenz, insbesondere bei Final-Answer-only-Signalen.
 - *Kalibrationskurven und Aufgabenschwierigkeit*. Für Datensätze mit geringerer Genauigkeit verschieben sich die Reliability Curves stärker in Richtung Überkonfidenz (Abschnitt A.4). Dies legt nahe, dass Kalibrierung (und damit die Wahl geeigneter Selektionsschwellen) aufgabenspezifisch ist und ggf. pro Domäne bzw. Schwierigkeitsgrad feinjustiert werden sollte.

5.2. Vergleich mit Vorarbeiten

- **CoT-UQ [16].** Die zentrale Beobachtung, dass Reasoning-Kontext die Unsicherheitsabschätzung auf Antwortebene verbessert, wird repliziert und um robuste Evidenz aus fünf unabhängigen Läufen (inkl. Varianzbetrachtung) erweitert. Die in [16] berichteten AUROC-Verbesserungen werden bestätigt und durch konsistente Kalibrationsgewinne ergänzt (niedrigere ECE, günstigere Reliability Curves). Keyword-basierte Aggregationsmetriken — einschließlich Importance-gewichteter Varianten — liefern über Datensätze und Paradigmen hinweg konsistente AUROC- und ECE-Verbesserungen gegenüber Final-Answer-Baselins. Abweichend von [16] zeigen *P-True* und *Self-Probing* keine durchgängig überlegene Performance über alle Settings: *P-True* weist einen engen Konfidenzbereich (geringe Dynamik) auf und limitiert damit die Diskrimination, während *Self-Probing* eine schwache Kopplung zwischen prognostizierter Confidence und beobachteter Accuracy zeigt, was sich in den Reliability Curves zeigt.
- **Min-Prob als Halluzinationsindikator [15].** *Probas-Min* erweist sich erneut als starkes, lokalsensitives Warnsignal: Fallstudie 2 (vgl. Absatz 4.3.1) demonstriert, dass sehr niedrige Token-Wahrscheinlichkeiten an semantisch kritischen Stellen Fehlantworten früh anzeigen können.
- **CoD [3].** Die beobachteten Trade-offs (geringere Kosten/Latency bei sehr moderatem Genauigkeitsverlust von 2,84 Prozentpunkten) sind konsistent mit den in der Literatur beschriebenen Tendenzen. Die vorliegenden Laufzeitmessungen umfassen Pipeline-Overhead (z. B. Extraktion, Self-Evaluation, Postprocessing) und sind daher konservativ; die Originalarbeit quantifiziert zusätzlich Token- und Latenzgewinne auf Modell- und Prompt-Ebene.
- **ToT [4].** Die starke Wirkung der Evaluate-Heuristik wird bestätigt; dies gilt modellübergreifend (Austausch *GPT-4* → *DeepSeek*). Ein naiver Ersatz durch zeilenbasierte Token-Log-Wahrscheinlichkeiten ist im BFS-Setting für *Game of 24* unzureichend.

5.3. Limitationen und Validitätsbedrohungen

- **Externe Validität (Modell, Domäne):** Ein Grossteil der Experimente nutzt *Llama-3.1-8B-Instruct*; die Übertragbarkeit auf grössere Modelle (z.,B. 70–120B, GPT-ähnlich; DeepSeek-V3-Chat) oder andere Domänen (Fachtexte, Mehrsprachigkeit) ist nicht garantiert. DeepSeek-V3-Chat wurde ausschliesslich im ToT-Setup eingesetzt.
- **Konstruktvalidität der Labels:** Für Zahlenaufgaben wurde String/Integer-Matching eingesetzt, für Multi-Hop-QA ein LLM-Judge (GPT-4o-mini). Beide Verfahren können Fehlklassifikationen verursachen. Eine Stichprobenprüfung in Unterabschnitt 4.3.2 erhöhen die Nachvollziehbarkeit.

5. Diskussion

- **Run-to-Run-Varianz:** Sampling, nichtdeterministische GPU-Operatoren und API-Drift verursachen Streuung. Fünf Runs reduzieren, aber eliminieren diese nicht. Konfidenzintervalle werden berichtet; Pooling über Runs wurde vermieden.
- **Laufzeitmessung CoT vs. CoD:** Die Wall-Clock-Zeiten umfassen die gesamte Pipeline (inkl. Keyword-Extraktion, Self-Evaluation, CPU/I/O). Ein isolierter Inferenzvergleich war nicht Ziel der Studie; die Effizienzaussage stützt sich primär auf konzeptionelle Überlegungen und Literatur (vgl. [3]).

5.4. Praktische Implikationen

- **Selektive Vorhersage und Risikosteuerung:** Keyword-gewichtete Aggregationsmetriken anhand der Reasoning-Kette liefern robuste Signale für accept/abstain/escalate-Entscheidungen. Schwellen sollten auf einem Validationssplit anhand von Risk-Coverage-Kurven gewählt und die Kalibration (ECE) geprüft bzw. bei Bedarf nachkalibriert werden.
- **Effizienz durch CoD:** CoD ist eine sinnvolle Default-Strategie, wenn Kosten/Latenz knapp sind und ein geringer Genauigkeitsverlust toleriert wird. Die UQ-Qualität bleibt weitgehend erhalten; Keyword-basierte Aggregationsmetriken funktionieren unter CoD ähnlich gut wie unter CoT.
- **Aufgabenspezifische Kalibrierung:** Da Datensätze mit geringerer Genauigkeit zu Überkonfidenz tendieren, sollten Kalibrations- und Schwellenmodelle pro Domäne bzw. Schwierigkeitsgrad getrennt trainiert werden.

6. Fazit und Ausblick

6.1. Zusammenfassung der wichtigsten Erkenntnisse

Keyword-gewichtete Aggregationsmetriken, die semantisch relevante Tokens der Reasoning-Kette stärker berücksichtigen, übertreffen Final-Answer-Baselines konsistent sowohl in der Diskrimination (AUROC) als auch in der Kalibration (ECE). CoD erhält die Qualität der Unsicherheitsabschätzung weitgehend aufrecht, reduziert jedoch Token-Budget und Latenz deutlich. Im ToT-Setting erweist sich die publizierte *Evaluate*-Heuristik als nur schwer ersetzbar: Naive, tokenwahrscheinlichkeitsbasierte Scores senken zwar die Kosten, führen aber zu spürbaren Genauigkeitseinbussen.

6.2. Beantwortung der Forschungsfragen

F1 (Aggregation). Ja. Die Einbeziehung von gewichteten Reasoning-Ketten Informationen (Keywords und Importance-Scores) verbessert die Aggregationsmetriken (*Probas-Mean*, *Probas-Min*, *Token-SAR*) gegenüber Final-Answer-only-Baselines, gemessen an höherer AUROC, niedrigerem ECE und günstigeren Reliability-Kurven. *Probas-Min* und *Token-SAR* zeigen die stabilsten Zugewinne.

F2 (Self-Evaluation). Eher nein bzw. nur begrenzt. Zusätzliche Reasoning-Kontexte (Allsteps/Keystep/Allkeywords/Keykeywords) verbessern *P-True* und *Self-Probing* gegenüber ihren Final-Answer-only-Baselines nicht konsistent. *P-True* weist einen engen Confidence-Bereich auf, *Self-Probing* zeigt eine schwache Kopplung zwischen Confidence und Accuracy. Beide Metriken zeigen zudem sehr tiefe AUROC-Werte.

F3 (Effizienz CoD vs. CoT). Weitgehend ja. CoD erhält die UQ-Qualität (AUROC/ECE) von CoT bei reduzierter Antwortlänge und typischerweise geringerer Latenz. Der beobachtete Accuracy-Trade-off beträgt im Mittel $-2,84$ Prozentpunkte. Laufzeitgewinne können durch Pipeline-Overhead (z. B. I/O) teilweise überdeckt werden; der Inferenzvorteil bleibt jedoch konzeptionell bestehen.

F4 (ToT: Kosten–Genauigkeit). Der Trade-off fällt ungünstig aus. Der Ersatz der *Evaluate*-Heuristik durch tokenwahrscheinlichkeitsbasierte Scores reduziert die Kosten deutlich ($3\text{--}5\times$ vs. DeepSeek-ToT-Baseline; $\approx 98,6\%$ vs. GPT-4), senkt aber die Erfolgsrate stark. Bestes Generierungsregime: Single-Solution mit *Probas-Mean* bei $T \approx 1,8$ (24 %) bzw. *Probas-Min* bei $T \approx 1,5$ (18 %); Multiple-Solution war mit $\approx 3\%$ klar unterlegen. Die Heuristik bleibt im BFS-Setting (*Game of 24*) klar überlegen.

6.3. Ausblick auf zukünftige Arbeiten

- **Generalisierung von ToT mit UQ:** Entwicklung einer Aufgaben- und Domänen-übergreifenden ToT-Variante mit unsicherheitsbewusster Bewertung (jenseits *Game of 24*) und systematischer Vergleich zu CoT/CoD.
- **Längen-Effekte und Confidence-Inflation:** Quantitative Analyse des Einflusses der Reasoning-Kettenlänge auf Tokenwahrscheinlichkeiten und Kalibration der Finalen Antwort.
- **Bias in Keyword/Importance Scoring:** Untersuchung potenzieller Modell-Biases bei Keyword-Extraktion und Importance-Scoring (“Self-Model Bias”); Vergleich mit externen, modellunabhängigen Scorern.
- **Modelltransfer:** Replikation auf stärkeren LLMs und unterschiedlichen Architekturen; Analyse der Übertragbarkeit der UQ-Gewinne (Diskrimination und Kalibration).
- **Fairness und Domänenübertragbarkeit:** Evaluation in weiteren Domänen (z. B. Medizin, Recht) inklusive Prüfung auf Verzerrungen in UQ-Signalen sowie domänenspezifische Nachkalibrierung.

Literaturverzeichnis

- [1] S. Yao, “Tree of thoughts – beispielbild auf x,” <https://x.com/ShunyuYao12/status/1659357554709807106/photo/1>, 2023, online-Bild, zuletzt abgerufen am 2025-07-09.
- [2] S. Rajaraman, P. Ganesan, and S. K. Antani. (2021) A sample sketch of the reliability diagram showing perfectly calibrated, overconfident, underconfident, uniformly overconfident, and uniformly underconfident predictions. Figure 2 in “Does deep learning model calibration improve performance in class-imbalanced medical image classification?”. [Online]. Available: <https://www.researchgate.net/publication/355060397/figure/fig2/AS:1076315306373127@1633624974709/A-sample-sketch-of-the-reliability-diagram-showing-perfectly-calibrated-overconfident.png>
- [3] S. Xu, W. Xie, L. Zhao, and P. He, “Chain of draft: Thinking faster by writing less,” 2025. [Online]. Available: <https://arxiv.org/abs/2502.18600>
- [4] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, “Tree of thoughts: Deliberate problem solving with large language models,” 2023. [Online]. Available: <https://arxiv.org/abs/2305.10601>
- [5] O. Shorinwa, Z. Mei, J. Lidard, A. Z. Ren, and A. Majumdar, “A survey on uncertainty quantification of large language models: Taxonomy, open research challenges, and future directions,” 2025. [Online]. Available: <https://arxiv.org/abs/2412.05563>
- [6] L. Liu, Y. Pan, X. Li, and G. Chen, “Uncertainty estimation and quantification for llms: A simple supervised approach,” 2024. [Online]. Available: <https://arxiv.org/abs/2404.15993>
- [7] Y. Yang, H. Yoo, and H. Lee, “Maqa: Evaluating uncertainty quantification in llms regarding data uncertainty,” 2025. [Online]. Available: <https://arxiv.org/abs/2408.06816>
- [8] X. Qiu and R. Miikkulainen, “Semantic density: Uncertainty quantification for large language models through confidence measurement in semantic space,” 2024. [Online]. Available: <https://arxiv.org/abs/2405.13845>
- [9] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, “Chain-of-thought prompting elicits reasoning in large language models,” 2023. [Online]. Available: <https://arxiv.org/abs/2201.11903>
- [10] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, “Large language models are zero-shot reasoners,” 2023. [Online]. Available: <https://arxiv.org/abs/2205.11916>

- [11] Princeton NLP Group, “Reproduced game-of-24 run: gpt-4_0.7_propose1_value3_greedy5_start900_end1000.json,” https://raw.githubusercontent.com/princeton-nlp/tree-of-thought-llm/master/logs/game24/gpt-4_0.7_propose1_value3_greedy5_start900_end1000.json, 2023, reproduktionslauf, gemeldete Erfolgsrate: 69 %, steht auch im readme.
- [12] S. Kadavath, T. Conerly, A. Askell, T. Henighan, D. Drain, E. Perez, N. Schiefer, Z. Hatfield-Dodds, N. DasSarma, E. Tran-Johnson, S. Johnston, S. El-Showk, A. Jones, N. Elhage, T. Hume, A. Chen, Y. Bai, S. Bowman, S. Fort, D. Ganguli, D. Hernandez, J. Jacobson, J. Kernion, S. Kravec, L. Lovitt, K. Ndousse, C. Olsson, S. Ringer, D. Amodei, T. Brown, J. Clark, N. Joseph, B. Mann, S. McCandlish, C. Olah, and J. Kaplan, “Language models (mostly) know what they know,” 2022. [Online]. Available: <https://arxiv.org/abs/2207.05221>
- [13] M. Xiong, Z. Hu, X. Lu, Y. Li, J. Fu, J. He, and B. Hooi, “Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms,” 2024. [Online]. Available: <https://arxiv.org/abs/2306.13063>
- [14] K. Tian, E. Mitchell, A. Zhou, A. Sharma, R. Rafailov, H. Yao, C. Finn, and C. D. Manning, “Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback,” 2023. [Online]. Available: <https://arxiv.org/abs/2305.14975>
- [15] N. Varshney, W. Yao, H. Zhang, J. Chen, and D. Yu, “A stitch in time saves nine: Detecting and mitigating hallucinations of llms by validating low-confidence generation,” 2023. [Online]. Available: <https://arxiv.org/abs/2307.03987>
- [16] B. Zhang and R. Zhang, “Cot-uq: Improving response-wise uncertainty quantification in llms with chain-of-thought,” 2025. [Online]. Available: <https://arxiv.org/abs/2502.17214>
- [17] J. Duan, H. Cheng, S. Wang, A. Zavalny, C. Wang, R. Xu, B. Kailkhura, and K. Xu, “Shifting attention to relevance: Towards the predictive uncertainty quantification of free-form large language models,” 2024. [Online]. Available: <https://arxiv.org/abs/2307.01379>
- [18] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning, “Hotpotqa: A dataset for diverse, explainable multi-hop question answering,” 2018. [Online]. Available: <https://arxiv.org/abs/1809.09600>
- [19] X. Ho, A.-K. D. Nguyen, S. Sugawara, and A. Aizawa, “Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps,” 2020. [Online]. Available: <https://arxiv.org/abs/2011.01060>
- [20] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, “Training verifiers to solve math word problems,” 2021. [Online]. Available: <https://arxiv.org/abs/2110.14168>

Literaturverzeichnis

- [21] A. Patel, S. Bhattacharya, and N. Goyal, “Are nlp models really able to solve simple math word problems?” 2021. [Online]. Available: <https://arxiv.org/abs/2103.07191>
- [22] S.-Y. Miao, C.-C. Liang, and K.-Y. Su, “A diverse corpus for evaluating and developing english math word problem solvers,” 2021. [Online]. Available: <https://arxiv.org/abs/2106.15772>
- [23] S. Yao, Y. Liu, and Y. Bisk. (2023) Tree-of-Thought: Official github repository. GitHub repository – includes Game-of-24 subset. [Online]. Available: <https://github.com/princeton-nlp/tree-of-thought-llm>
- [24] B. Zhang and R. Zhang. (2025) CoT-UQ: Codebase for “improving response-wise uncertainty quantification in llms with chain-of-thought”. GitHub repository. [Online]. Available: <https://github.com/ZBox1005/CoT-UQ>
- [25] (2025) Meta-llama-3.1-8b-instruct – hugging face model card. Meta AI. Model card. [Online]. Available: <https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct>
- [26] (2025) Gpt-4o mini model documentation. OpenAI. OpenAI API docs. [Online]. Available: <https://platform.openai.com/docs/models/gpt-4o-mini>
- [27] (2025) Deepseek chat v3 api documentation. DeepSeek. API reference. [Online]. Available: <https://api-docs.deepseek.com/>
- [28] M. Hirt, “Unsicherheitsabschätzung in large language models mittels reasoning-strategien — code & daten,” <https://github.com/maurohirt/Unsicherheitsabschaetzung-in-Large-Language-Models-mittels-Reasoning-Strategien>, 2025, commit 2584771 (258477120a28eeb1e831436c12caf6956220bff9), accessed: Aug. 10, 2025.

A. Appendix

A.1. Beispielaufgaben aus jedem Datensatz

Tabelle A.1.: Beispielaufgaben aus jedem Datensatz

Dataset	Question	Gold answer
2WikiMHQA	Who is the mother of the director of the film <i>Polish-Russian War</i> ?	Małgorzata Braунek
HotpotQA	Which band has more members, <i>We Are the Ocean</i> or <i>The Dream Academy</i> ?	We Are the Ocean
SVAMP	Each pack of DVDs costs \$76 with a \$25 discount. How much does one pack cost after the discount?	51
ASDiv	Seven red apples and two green apples are in the basket. How many apples are in the basket?	9
GSM8K	Janet's ducks lay 16 eggs per day. She eats 3, bakes 4, and sells the rest at \$2 each. How much does she earn per day?	18
Game-of-24	Numbers: 2 3 5 12	$12/(3 - (5/2)) = 24$

A.2. Komplette Instruktionstemplates für Self-Evaluation Methoden

A.2.1. Self-Probing

Baseline

Question: <QUESTION>

Possible Answer: <KONTEXT_b1>

Q: How likely is the above answer to be correct?

Please first show your reasoning concisely and then answer with:

““Confidence: [the probability of answer <KONTEXT_b1> to be correct, please include only the numerical number]%““

Confidence:

A. Appendix

Allsteps

Question: <QUESTION>

Possible Answer: <KONTEXT_b1>

A step-by-step reasoning to the possible answer:

<KONTEXT_all>

Q: Considering these reasoning steps as additional information, how likely is the above answer to be correct?

Please first show your reasoning concisely and then answer with:

“‘Confidence: [the probability of answer <KONTEXT_b1> to be correct, please include only the numerical number]%'“‘

Confidence:

Keystep

Question: <QUESTION>

Possible Answer: <KONTEXT_b1>

The most critical step in reasoning to the possible answer: <KONTEXT_key>

Q: Considering this critical reasoning step as additional information, how likely is the above answer to be correct?

Please first show your reasoning concisely and then answer with:

“‘Confidence: [the probability of answer <KONTEXT_b1> to be correct, please include only the numerical number]%'“‘

Confidence:

Allkeywords

Question: <QUESTION>

Possible Answer: <KONTEXT_b1>

Keywords during reasoning to the possible answer: <KONTEXT_allkw>

Q: Considering these keywords as additional information, how likely is the above answer to be correct?

Please first show your reasoning concisely and then answer with:

“‘Confidence: [the probability of answer <KONTEXT_b1> to be correct, please include only the numerical number]%'“‘

Confidence:

Keykeywords

Question: <QUESTION>

Possible Answer: <KONTEXT_b1>

Keywords during reasoning to the possible answer: <KONTEXT_keykw>

A. Appendix

Q: Considering these keywords as additional information,
how likely is the above answer to be correct?

Please first show your reasoning concisely and then answer with:

“Confidence: [the probability of answer <KONTEXT_b1> to be correct,
please include only the numerical number]%”

Confidence:

A.2.2. P-True

Baseline

The problem is: <QUESTION>

A student submitted: <KONTEXT_b1>

Is the student's answer:

- (A) True
- (B) False

The student's answer is:

Allsteps

The problem is: <QUESTION>

A student submitted: <KONTEXT_b1>

The student explained the answer, which included a step-by-step
reasoning: <KONTEXT_all>

Considering these reasoning steps as additional information,
is the student's answer:

- (A) True
- (B) False

The student's answer is:

Keystep

The problem is: <QUESTION>

A student submitted: <KONTEXT_b1>

The student explained the answer, where the most critical step is:
<KONTEXT_key>

Considering this critical reasoning step as additional information,
is the student's answer:

- (A) True
- (B) False

The student's answer is:

A. Appendix

Allkeywords

The problem is: <QUESTION>
A student submitted: <KONTEXT_b1>
The student explained the answer, which included the following keywords
<KONTEXT_allkw>

Considering these keywords as additional information,
is the student's answer:

- (A) True
- (B) False

The student's answer is:

Keykeywords

The problem is: <QUESTION>
A student submitted: <KONTEXT_b1>
The student explained the answer, which included the following keywords
<KONTEXT_keykw>

Considering these keywords as additional information,
is the student's answer:

- (A) True
- (B) False

The student's answer is:

A.3. Reproduzierbare Umgebung und Ausführung

A.3.1. Python-Abhängigkeiten

```
# --- Core Numerics ---
numpy==1.26.4
scipy==1.13.1
pandas==2.2.2
tqdm==4.66.4

# --- PyTorch Stack ---
torch==2.2.2+cu118

# --- Hugging Face Ecosystem ---
transformers==4.45.0
accelerate==0.30.1
datasets==2.20.0
safetensors==0.4.3
```

A. Appendix

```
sentencepiece==0.1.99
peft==0.12.0
trl==0.9.4
sentence-transformers==3.0.1

# --- Metrics / Evaluation / NLP Utilities ---
torchmetrics==1.4.0.post0
rouge-chinese==1.0.3
jieba==0.42.1
nltk==3.8.1

# --- Serving / API / Web ---
fastapi==0.95.1
uvicorn==0.23.2
sse-starlette==1.6.5
gradio==3.50.2
openai==1.6.0

# --- Utilities ---
pyyaml==6.0.1
einops==0.8.0
```

Listing A.1: Python Abhängigkeiten (requirements.txt)

A.3.2. Singularity Build Skript

```
#!/bin/bash
#SBATCH --job-name=build_cotuq_sif
#SBATCH --partition=performance
#SBATCH --gres=gpu:0
#SBATCH --cpus-per-task=4
#SBATCH --mem=32G
#SBATCH --time=00:30:00
#SBATCH --output=build_logs/build_sif_%j.out

module load singularity

# Cache-Verzeichnis (verhindert grosse Daten im HOME)
export SINGULARITY_CACHEDIR=/tmp/$USER-singularity-cache
mkdir -p "$SINGULARITY_CACHEDIR"

# Zielpfad fuer das resultierende SIF
SIF=$HOME/containers/cot-uq_latest.sif
mkdir -p "$(dirname "$SIF")"

# Direkter Pull vom Docker-Registry (nur falls noch nicht vorhanden)
if [ ! -f "$SIF" ]; then
    echo "Pulling $SIF"
```

A. Appendix

```
singularity pull "$SIF" docker://docker.io/maurohirtfhnw/cot-uq:  
latest  
else  
    echo "SIF already exists: $SIF"  
fi
```

```
echo "SIF ready at $SIF"
```

Listing A.2: Singularity Build / Pull Skript

A.3.3. Dockerfile

```
FROM nvidia/cuda:11.8.0-cudnn8-runtime-ubuntu22.04

ARG DEBIAN_FRONTEND=noninteractive
ENV DEBIAN_FRONTEND=noninteractive TZ=Europe/Zurich

# System-Pakete & Python 3.9
RUN apt-get update && \  
    apt-get install -y software-properties-common curl && \  
    add-apt-repository ppa:deadsnakes/ppa && \  
    apt-get update && \  
    apt-get install -y --no-install-recommends \  
        git build-essential tzdata \  
        python3.9 python3.9-distutils python3.9-venv && \  
    python3.9 -m ensurepip && \  
    python3.9 -m pip install --upgrade pip && \  
    ln -sf /usr/bin/python3.9 /usr/bin/python && \  
    ln -sf /usr/local/bin/pip /usr/bin/pip && \  
    rm -rf /var/lib/apt/lists/*

WORKDIR /app

# Requirements (eine Datei im Repository)
COPY requirements.txt .

# Installation aller Abhaengigkeiten (einschliesslich PyTorch
Wheels via extra-index-url in requirements)
RUN python -m pip install --no-cache-dir -r requirements.txt

# Projektcode
COPY CoT-UQ/ CoT-UQ/
ENV PYTHONPATH=/app/CoT-UQ:$PYTHONPATH
WORKDIR /app/CoT-UQ

CMD ["/bin/bash"]
```

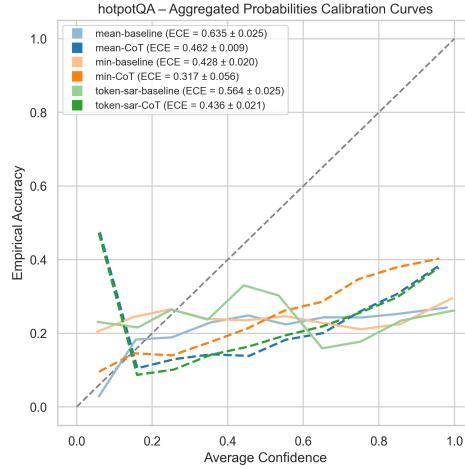
Listing A.3: Dockerfile (verwendet requirements.txt)

A. Appendix

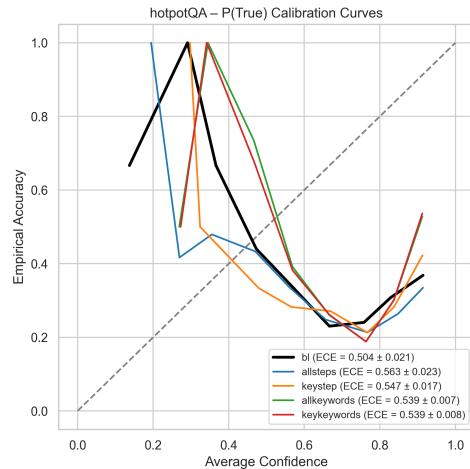
A.4. Reliability Curves

A.4.1. Chain-of-Thought (CoT)

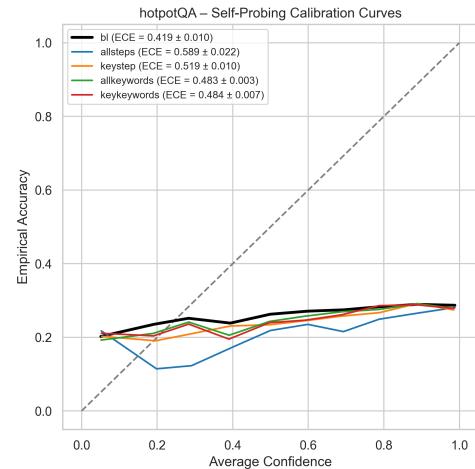
HotpotQA



(a) Aggregated Probabilities



(b) P_{true} -Strategien

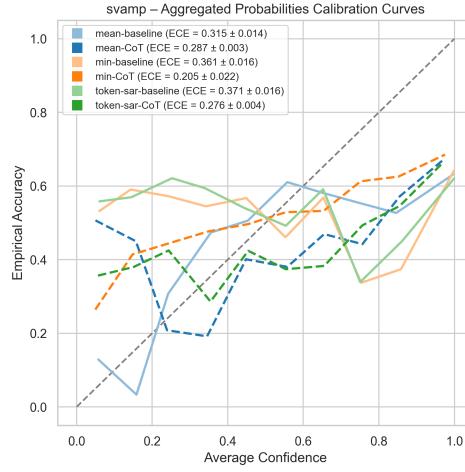


(c) Self-Probing-Strategien

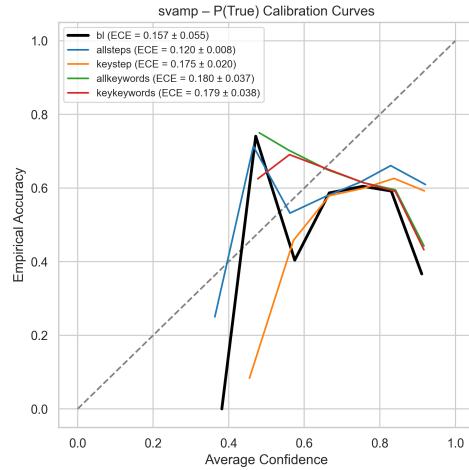
Abbildung A.1.: Reliability-Vergleich der CoT-Varianten für HotpotQA.

A. Appendix

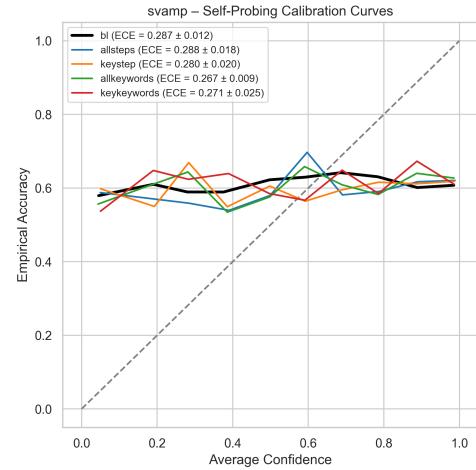
SVAMP



(a) Aggregated Probabilities



(b) P_{true} -Strategien

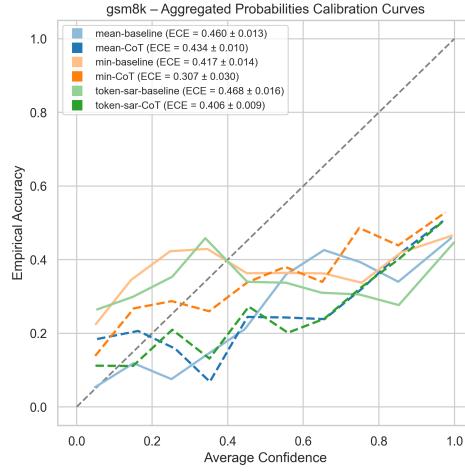


(c) Self-Probing-Strategien

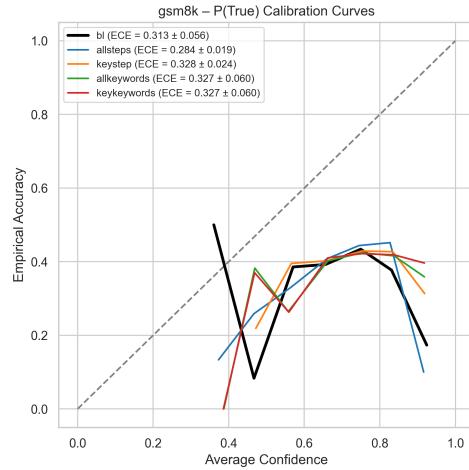
Abbildung A.2.: Reliability-Vergleich der CoT-Varianten für SVAMP.

A. Appendix

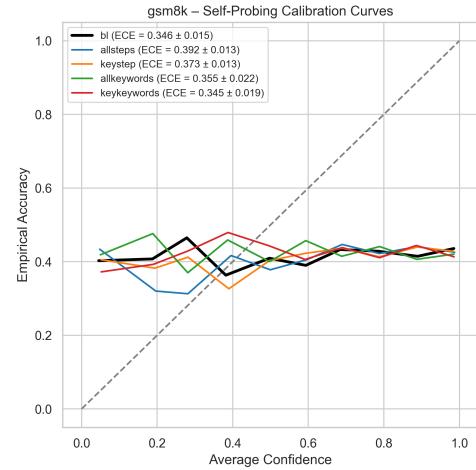
GSM8K



(a) Aggregated Probabilities



(b) P_{true} -Strategien

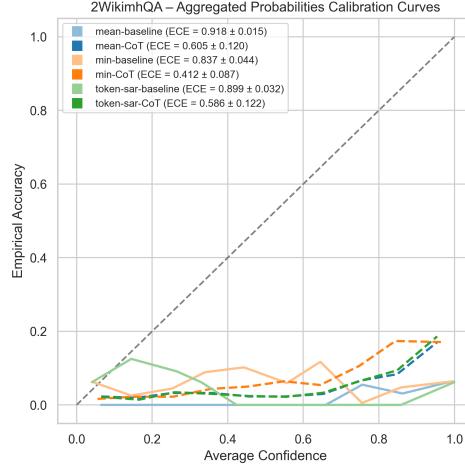


(c) Self-Probing-Strategien

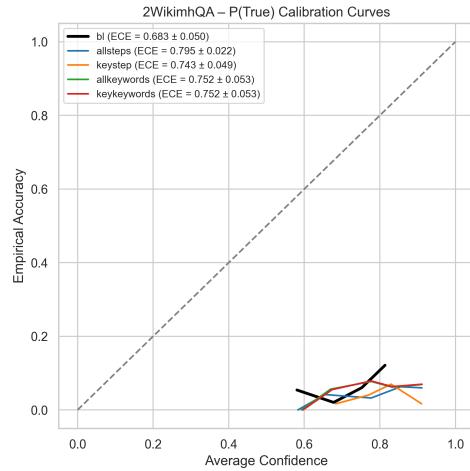
Abbildung A.3.: Reliability-Vergleich der CoT-Varianten für **GSM8K**.

A. Appendix

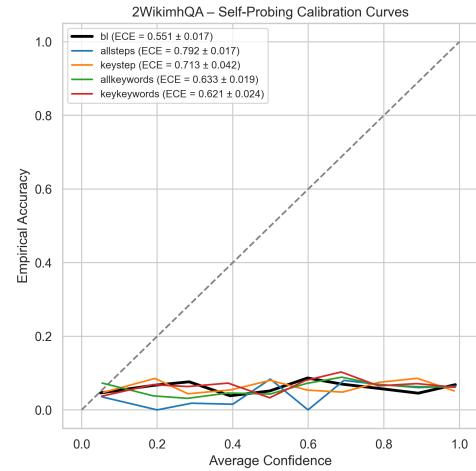
2WikiMHQA



(a) Aggregated Probabilities



(b) P_{true} -Strategien

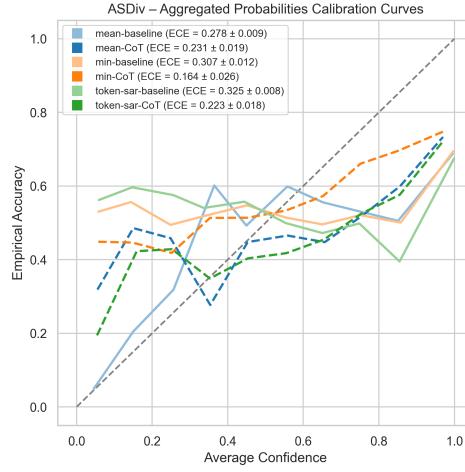


(c) Self-Probing-Strategien

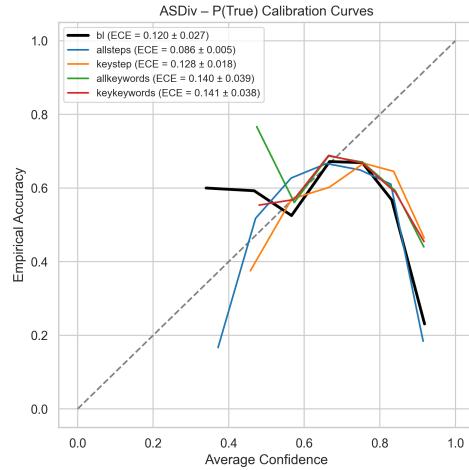
Abbildung A.4.: Reliability-Vergleich der CoT-Varianten für **2WikiMHQA**.

A. Appendix

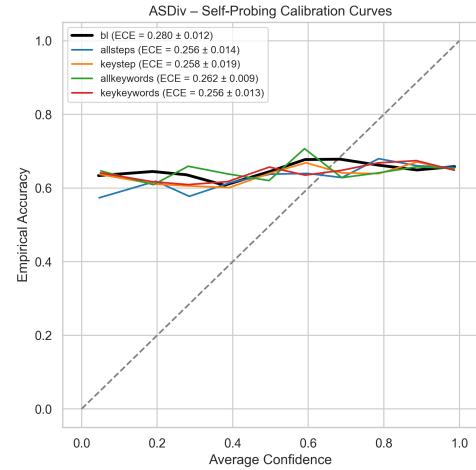
ASDiv



(a) Aggregated Probabilities



(b) P_{true} -Strategien



(c) Self-Probing-Strategien

Abbildung A.5.: Reliability-Vergleich der **CoT**-Varianten für **ASDiv**.

A. Appendix

A.4.2. Chain-of-Draft (CoD)

HotpotQA

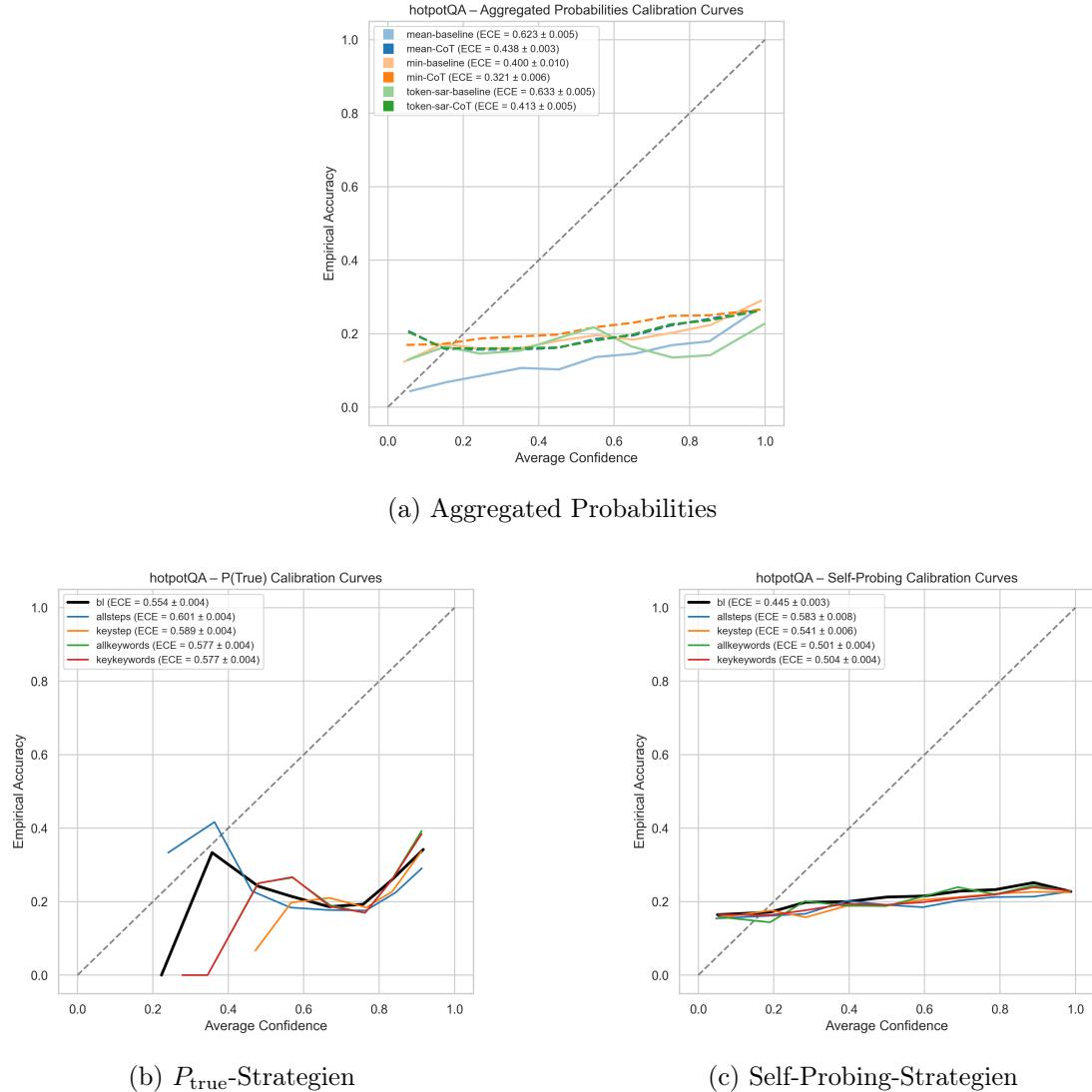
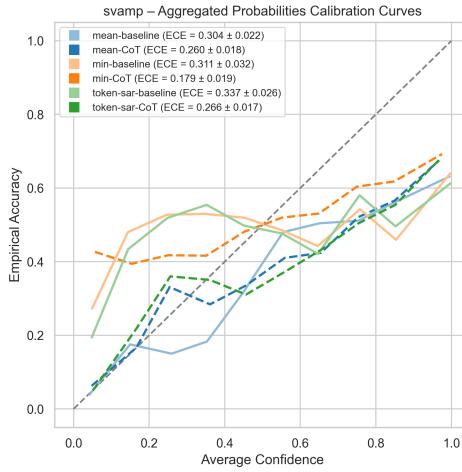


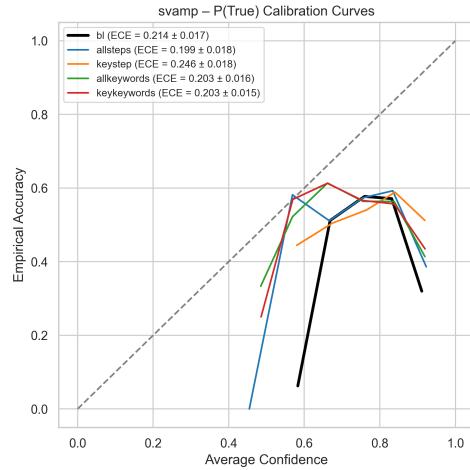
Abbildung A.6.: Reliability-Vergleich der CoD-Varianten für HotpotQA.

A. Appendix

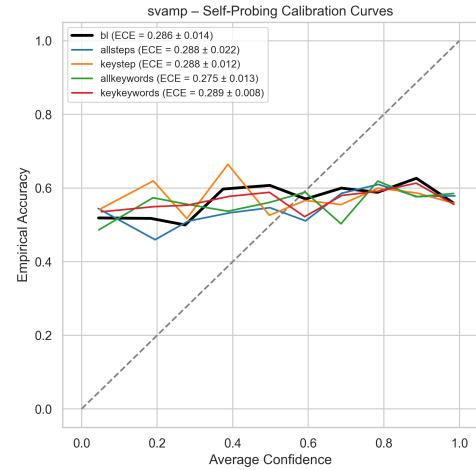
SVAMP



(a) Aggregated Probabilities



(b) P_{true} -Strategien

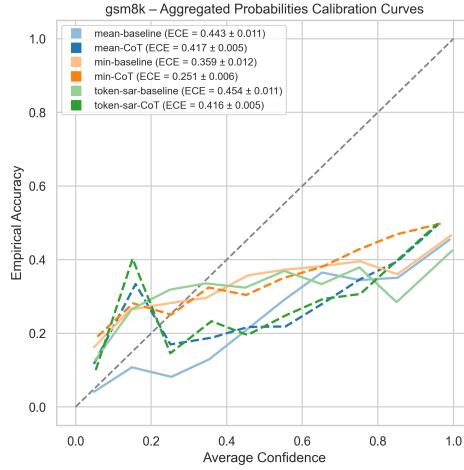


(c) Self-Probing-Strategien

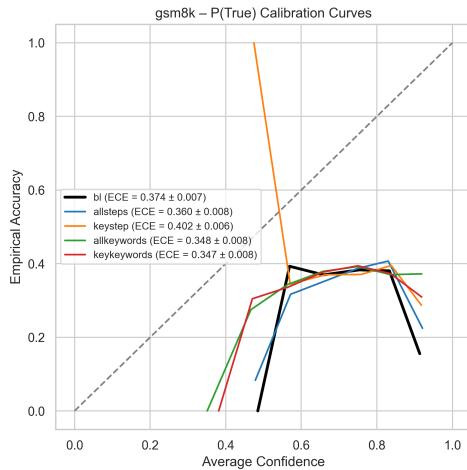
Abbildung A.7.: Reliability-Vergleich der CoD-Varianten für SVAMP.

A. Appendix

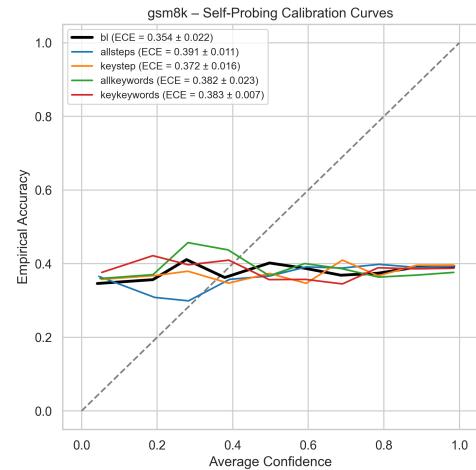
GSM8K



(a) Aggregated Probabilities



(b) P_{true} -Strategien



(c) Self-Probing-Strategien

Abbildung A.8.: Reliability-Vergleich der CoD-Varianten für **GSM8K**.

A.5. Erweiterte Accuracy-Auswertungen

Die folgenden Tabellen berichten erweiterte Kennzahlen der Accuracy pro Datensatz über $n = 5$ Läufe: Mittelwert, Standardabweichung, Minimum, Maximum sowie 95 %-Konfidenzintervalle (KI) basierend auf dem Student- t -Intervall mit $df = 4$. Alle Werte sind in Prozent angegeben.

A. Appendix

Tabelle A.2.: Erweiterte Genauigkeitsstatistiken (CoT). Fünf Läufe pro Datensatz; Angaben in Prozent. 95 %-KI nach Student-*t* (df = 4).

Datensatz	Mittelwert	StdAbw.	Min	Max	95 %-KI	Läufe
2WikiMHQA	6.10	0.52	5.63	6.95	[5.45, 6.75]	5
HotpotQA	26.08	1.57	24.24	27.52	[24.13, 28.03]	5
GSM8K	41.90	1.52	40.62	44.20	[40.01, 43.79]	5
SVAMP	60.73	0.35	60.22	61.10	[60.30, 61.16]	5
ASDiv	64.59	0.87	63.50	65.42	[63.51, 65.67]	5

Tabelle A.3.: Erweiterte Genauigkeitsstatistiken (CoD). Fünf Läufe pro Datensatz; Angaben in Prozent. 95 %-KI nach Student-*t* (df = 4).

Datensatz	Mittelwert	StdAbw.	Min	Max	95 %-KI	Läufe
2WikiMHQA	3.23	0.61	2.50	3.88	[2.47, 3.99]	5
HotpotQA	21.02	0.32	20.64	21.52	[20.62, 21.42]	5
GSM8K	37.91	0.55	37.28	38.77	[37.23, 38.59]	5
SVAMP	56.73	1.37	55.36	58.70	[55.03, 58.43]	5
ASDiv	66.29	0.60	65.66	66.98	[65.55, 67.03]	5

A.6. Brier Score

A.6.1. Brier Score (CoD)

Der Brier Score wird als Mittelwert über fünf Läufe berichtet; niedriger ist besser.

Tabelle A.4.: Brier Score (CoD): Baseline vs. Keyword-Variante (Mittel über 5 Läufe). Niedriger ist besser.

Datensatz	Metrik	Baseline	Keyword	Δ
ASDiv	Probas-Mean	0.2656	0.2363	0.029
ASDiv	Probas-Min	0.2824	0.2419	0.041
ASDiv	Token-SAR	0.2983	0.2348	0.064
GSM8K	Probas-Mean	0.4384	0.4048	0.034
GSM8K	Probas-Min	0.3907	0.3123	0.078
GSM8K	Token-SAR	0.4723	0.4069	0.065
HotpotQA	Probas-Mean	0.5700	0.4104	0.160
HotpotQA	Probas-Min	0.4008	0.3338	0.067
HotpotQA	Token-SAR	0.6351	0.4126	0.223
SVAMP	Probas-Mean	0.3301	0.3026	0.028
SVAMP	Probas-Min	0.3356	0.2758	0.060

Fortsetzung auf der nächsten Seite

A. Appendix

Datensatz	Metrik	Baseline	Keyword	Δ
SVAMP	Token-SAR	0.3571	0.3030	0.054
2WikiMHQA	Probas-Mean	0.6291	0.4871	0.142
2WikiMHQA	Probas-Min	0.2059	0.2615	-0.056
2WikiMHQA	Token-SAR	0.5556	0.4810	0.075

A.6.2. Brier Score (CoT)

Der Brier Score wird als Mittelwert über fünf Läufe berichtet; niedriger ist besser.

Tabelle A.5.: Brier Score (CoT): Baseline vs. Keyword-Variante (Mittel über 5 Läufe). Niedriger ist besser.

Datensatz	Metrik	Baseline	Keyword	Δ
ASDiv	Probas-Mean	0.3040	0.2708	0.033
ASDiv	Probas-Min	0.3217	0.2511	0.071
ASDiv	Token-SAR	0.3345	0.2715	0.063
GSM8K	Probas-Mean	0.4648	0.4235	0.041
GSM8K	Probas-Min	0.4369	0.3449	0.092
GSM8K	Token-SAR	0.4846	0.4216	0.063
HotpotQA	Probas-Mean	0.6102	0.4215	0.189
HotpotQA	Probas-Min	0.4508	0.3179	0.133
HotpotQA	Token-SAR	0.5907	0.4178	0.173
SVAMP	Probas-Mean	0.3427	0.3125	0.030
SVAMP	Probas-Min	0.3682	0.2798	0.088
SVAMP	Token-SAR	0.3779	0.3134	0.065
2WikiMHQA	Probas-Mean	0.9027	0.4675	0.435
2WikiMHQA	Probas-Min	0.8147	0.2817	0.533
2WikiMHQA	Token-SAR	0.8944	0.4483	0.446

Ehrlichkeitserklärung

Ich versichere hiermit, dass ich den vorliegenden Leistungsnachweis selbstständig verfasst und alle nicht von mir stammenden Textstellen, Bilder und sonstigen Quellen gemäss den wissenschaftlichen Zitierregeln korrekt ausgewiesen habe. Ebenso bestätige ich, dass sämtliche eingesetzten KI-Assistenzsysteme – wie *ChatGPT* oder *DeepSeek* – ausschliesslich unterstützend bei der sprachlichen Überarbeitung und Formulierung einzelner Textpassagen eingesetzt wurden. Alle verwendeten Materialien wurden entweder von mir selbst erstellt oder die entsprechenden Rechte ordnungsgemäss erworben.

Ich bin mir bewusst, dass meine Arbeit auf Plagiate sowie auf eine nicht autorisierte Drittautorschaft (menschlichen oder technischen Ursprungs) überprüft werden kann und dass ein Verstoss gegen diese Erklärung disziplinarische Konsequenzen nach sich ziehen kann.

Windisch, August 2025



Unterschrift