User Guide

zu

Kombinatorische Methoden für Konsistenz-Tests von Large Language Models – KomMKonLLM

ein

netidee Projekt*

Förderjahr 2024 / Projekt Call #19 / ProjectID: 7409

finanziert durch die

Internet Privatstiftung Austria†

ausgeführt von

Bernhard Garn Ludwig Kampel Manuel Leithner
MATRIS Research Group, SBA Research

Kurzfassung

Motivation. Die jüngsten Fortschritte in der *künstliche Intelligenz* (KI) und die zunehmende Popularität von *großen Sprachmodellen*[1] wie beispielsweise GPT-4[2] und ChatGPT[3] haben das Interesse an potenten und zuverlässigen KI-Systemen verstärkt. Ein großes

*https://www.netidee.at/kommkonllm
†www.internetstiftung.at
[1]Übersetzung ins Deutsche für den englischen Begriff *large language models*, abgekürzt *LLMs*.
[2]https://openai.com/index/gpt-4/
[3]https://chatgpt.com/

Problem bei der Nutzung solcher Modelle in realen Anwendungen sind jedoch Zweifel bezüglich ihrer Konsistenz, insbesondere in Bezug auf die semantische Kohärenz der gegebenen Antworten. Empirische Untersuchungen haben gezeigt, dass LLMs bei bestimmten Arten von Konsistenz, wie z. B. der semantischen Konsistenz, widersprüchliche Antworten liefern können.

Methodik. Der Beitrag von KomMKonLLM zur Konsistenztestung von LLMs besteht darin, frei verfügbare Software[4] für die Anwendung von kombinatorischen Methoden für das semantische Konsistenztesten zu entwickeln. In dem von KomMKonLLM implementierten Ansatz werden Eingaben systematisch nach einer kombinatorischen Strategie variiert und dabei die Semantik des ursprünglichen Satzes bewahrt, sodass kompakte Mengen (i.e., Mengen mit einer möglichst geringen Kardinalität) von Konsistenztests ertellt werden können, mit denen LLMs dennoch auf mannigfaltige Weise auf die Fähigkeit zur konsistenten Beantwortung getestet werden können. Dies erfolgt unter der Erwartung, dass ein LLM auf alle Varianten einer Frage oder Aussage konsistent reagieren soll, solange die zugrundeliegende Semantik der ursprünglichen Frage erhalten bleibt. Ein zentraler Aspekt des in KomMKonLLM implementierten Ansatzes ist die Nutzung von Synonymen und semantisch ähnlichen Begriffen, um die Wörter in der Eingabe zu variieren, ohne die Bedeutung des Satzes grundlegend zu verändern. Dies ermöglicht eine breite Abdeckung verschiedener Formulierungen, was für die Konsistenztestung von entscheidender Bedeutung ist.

Implementierung. KomMKonLLM bietet eine server-seitige Software, welche die Erzeugung und Durchführung von kombinatorischen Konsistenztests ermöglicht. Die Software besteht aus mehreren Komponenten, welche individuelle Aufgaben übernehmen und miteinander über HTTP kommunizieren. Für das Management der Server-Komponenten wird Docker-Compose verwendet. Die Kontrolllogik ist in Python implementiert und die Ergebnisse werden in einer relationalen Datenbank gespeichert. Ein niederschwelliger Zugang zum System wird durch ein JupyterLab Notebook bereitgestellt, welches erweiterbaren Code zur Analyse beinhaltet.

Fazit. Kombinatorische Testmethoden bieten einen vielversprechenden Ansatz für die Konsistenztestung von LLMs, indem sie die Eingabemengen effektiv diversifizieren und derartig eingesetzt werden können, dass dabei die Semantik der ursprünglichen Eingabe bewahrt bleibt. Durch die Ausführung der so erzeugten Konsistenztests können Inkonsistenzen im Verhalten von LLMs identifiziert werden, was zur Verbesserung ihrer Zuverlässigkeit und Vertrauenswürdigkeit beiträgt. Durch die Anbindung an verschiedene kombinatorische Testfallgenerierungssoftware zusammen mit effizienter Wiederverwendung erzeugter Testfälle bietet KomMKonLLM Flexibilität und Performance in der Konsistenztesterzeugung. Weiters wird die Ausführung der erzeugten Testfälle gegen verschienede LLMs über eine gemeinsame Schnittstelle vereinfacht. Die aufbereitete Darstellung und Analyse der Ergebnisse ist durch die interaktive Umgebung eines JupyterLab Notebooks leicht zugänglich.

[4] Die Software Ergebnisse von KomMKonLLM sind unter der `MIT` Lizenz unter `https://github.com/KomMKonLLM/KomMKonLLM` verfügbar.

Inhaltsverzeichnis

1 Kombinatorische Methoden für Konsistenz-Tests von LLMs

Im Bereich der Testmethoden für KI und insbesondere für LLMs gewinnt das Konsistenztesten zunehmend an Bedeutung. Die Konsistenz eines LLMs ist ein entscheidender Faktor für das Vertrauen der Nutzer in die Korrektheit und Verlässlichkeit seiner Ausgaben. Im Folgenden wird ein kombinatorischer Ansatz zur Testung der Konsistenz von LLMs vorgestellt. Der Fokus liegt auf der Entwicklung eines Eingabe-Parameter-Modells (im Englischen als *input parameter model* bezeichnet, abgekürzt *IPM*) für Sätze, die als Eingabe in LLMs verwendet werden, sowie auf der Anwendung kombinatorischer Methoden zur Generierung von Testdatensätzen, die verschiedene Varianten dieser Sätze abdecken, ohne die ursprüngliche Semantik zu verfälschen.

KomMKonLLM Ansatz zur Konsistenztestung von LLMs Der Ansatz für die Konsistenztestung von LLMs basiert auf der Methodik des kombinatorischen Testens (CT, abgekürzt vom Englischen *combinatorial testing*), welcher die Eingabefragen systematisch - bis zu einem gewissen vorgegebenen Grad - nach einer kombinatorischen Strategie diversifiziert. Der KomMKonLLM-Prozess umfasst folgende Schritte:

1. Frage-Modellierung mittels eines IPMs: Zu gegebener Frage wird ein IPM erstellt, um die Eingabe für die Anwendung kombinatorischer Methoden zu strukturieren. Diese Frage kann eine einfache Testfrage aus einem bestehenden Datensatz sein, die so modelliert wird, dass sie eine breite Variation der möglichen semantisch-äquivalenten Umformulierungen erlaubt.

2. Erstellung eines kombinatorischen Testdatensatzes: Basierend auf der Struktur eines covering arrays (CAs), werden alternative Eingabefragen generiert, welche verschiedene Kombinationen von Wörtern, Phrasen und Synonymen beinhalten, während die semantische Bedeutung der ursprünglichen Frage beibehalten wird. Der so erstellte Testdatensatz stellt sicher, dass verschiedene Anfragen bis zu einer gewissen vorgegebenen Diversitätsstärke erzeugt werden.

3. Anwendung auf LLMs und Antwortvergleich: Die generierten Eingaben werden an ein LLM übergeben und die erhaltenen Antworten werden miteinander verglichen, um Inkonsistenzen zu identifizieren. Dies dient der Überprüfung, ob das Modell in der Lage ist, auf verschiedene Formulierungen der gleichen Frage konsistent zu antworten.

Im Folgenden wird zuerst die Motivation für KomMKonLLM dargelegt (Kapitel 1.1), gefolgt von einer Einbettung des KomMKonLLM-Ansatzes innerhalb verwandter wissenschaftlicher Arbeiten (Kapitel 1.2), sowie einer Beschreibung des KomMKonLLM-Testansatzes im Detail (Kapitel 1.3).

1.1 Hintergrund zu aktuellen Entwicklungen im Bereich KI und LLMs sowie Motivation

Mehrere Forscher sind sich einig, dass wir uns derzeit in der sogenannten *dritten Welle* der KI [9] befinden. Jüngste Entwicklungen scheinen einen Durchbruch in den Fähigkeiten von LLMs hervorgebracht zu haben, die nun in der Lage sind, menschenähnliche Gespräche zu führen. Solche KI-Systeme erfahren derzeit große Aufmerksamkeit in der Gesellschaft, werden intensiv in der Forschung diskutiert, häufig in den Medien behandelt und sind Gegenstand politischer Debatten [5,8]. Der jüngste Anstieg der Aufmerksamkeit für LLMs ist nicht zuletzt auf die Popularität von ChatGPT und dessen Plus-Variante zurückzuführen, die GPT-4 einer breiten Öffentlichkeit zugänglich macht [16]. Leistungssteigerungen und die verbesserte Genauigkeit von KI-Systemen gehen oft mit einer erhöhten Komplexität der verwendeten Modelle einher, was wiederum höhere Lernaufwände erfordert. Die dadurch entstehenden finanziellen und ökologischen Kosten - gemessen am Ressourcenaufwand des Trainings der Modelle und den verursachten CO_2-Emissionen - werden von Strubell et al. [17] abgeschätzt. Für das Training der Basisversion von BERT wird eine CO_2-Emission von 650 kg und ein Kostenaufwand zwischen $\$3,751$ und $\$12,571$ angegeben [17]. Solche Zahlen unterstreichen die hohen Kosten und den großen Datenbedarf für das Training und die Testung moderner KI-Modelle. Wie auch von Khashabi et al. [12] erwähnt, benötigen LLMs große Trainingsmengen, deren Erstellung sehr aufwändig ist. Um beispielsweise den BoolQ-Datensatz wie in [2] beschrieben zu erstellen, wurden fast 16.000 Beispielfragen gesammelt und von unabhängigen (menschlichen) Annotatoren verarbeitet. Dabei wurde jede Beispielfrage mit einem Wikipedia-Abschnitt gepaart, der genügend Informationen zur Beantwortung enthielt, um schließlich die richtige Antwort annotierten zu können. Neben dem reinen Training von KI-Systemen wird auch deren Testung zunehmend wichtiger. Wie beispielsweise Wotawa [18] betont, besteht mit der wachsenden Bedeutung von KI-Systemen die Notwendigkeit, geeignete Test- und Qualitätssicherungsmaßnahmen zu entwickeln. Im Gegensatz zu früheren Generationen von KI-Systemen bieten moderne LLMs erweiterte Fähigkeiten, menschenähnliche Gespräche zu führen. Studien von Jung, Dong und Lee [11] haben gezeigt, dass KI-Systeme, die menschenähnlicher agieren, bei den Interaktionspartnern ein höheres Vertrauen hervorrufen. Experimente, wie solche die in [3] durchgeführt wurden, zeigen, dass menschenähnliches Verhalten entscheidend für den Aufbau von resilientem Vertrauen sein kann, d. h. einer höheren Widerstandsfähigkeit gegenüber Vertrauensverlusten bei Automationsfehlern. Die Autoren von [3] folgern daher, dass der Grad der Menschenähnlichkeit einer KI von kritischer Bedeutung ist, die bei der Entwicklung neuer KI-Systeme und bei der Bewertung des Vertrauens der Menschen in KI-Systeme sorgfältig berücksichtigt werden sollte. Die Fähigkeit moderner KI-Systeme, menschenähnliche Gespräche zu führen, ist zwar beeindruckend, allerdings darf ein Faktor für das Vertrauen in solche Systeme nicht unterschätzt werden: *die Konsistenz*. Es gibt Studien, die argumentieren, dass Konsistenz einer der wichtigsten Faktoren beim Aufbau von Vertrauen in zwischenmenschlichen Beziehungen ist. In diesem Zusammenhang führte Dunn [4] zwei Experimente durch, in denen drei Aspekte von Vertrauen untersucht wurden. Er kam zu dem Schluss, dass Konsistenz das wichtigste Element für die Förderung kognitiv basierten Vertrauens in einer

dyadischen Beziehung ist.

Die Notwendigkeit auf Inkonsistenzen zu testen ist besonders dann dringlich, wenn LLMs in Bereichen eingesetzt werden, in denen Verlässlichkeit und Vertrauen eine zentrale Rolle spielen. Eine mögliche Lösung für dieses Problem ist der Einsatz kombinatorischer Testmethoden, die es ermöglichen, eine breite Palette möglicher Eingaben abzudecken und die Konsistenz der Antworten unter verschiedenen Bedingungen zu überprüfen, während die Anzahl der Tests überschaubar gehalten werden kann [7].

1.2 Verwandte wissenschaftliche Arbeiten zum (Konsistenz-) Testen von LLMs

Der Ansatz Abfragen an ein LLM zu manipulieren, während die Bedeutung erhalten bleibt, hat in letzter Zeit vermehrt Aufmerksamkeit bekommen. Die Arbeit von Gardner et al. [6] schlägt die Erstellung von *contrast sets* aus standard Testdatensätzen für *supervised learning* vor, indem die Testinstanzen manuell verändert werden, was zu einer genaueren und umfassenderen Bewertbarkeit der sprachlichen Fähigkeiten eines Modells führt. Ähnlich dazu ist die von Khashabi et al. [12]vorgeschlagene Methode zur Generierung von Trainingsdatensätzen, bei der auf einen kleinformatigen Ausgangsdatensatz manuelle Variationen der natürlichen Sprache angewendet werden. Die Autoren evaluieren ihren Ansatz unter Verwendung des BoolQ-Datensatzes [2] und stellen fest, dass dies die Robustheit von LLMs verbessert.

Jang und Lukasiewicz [10] präsentieren eine Konsistenzanalyse von ChatGPT. Sie analysieren die Fähigkeiten des LLMs, logisch konsistente Antworten in Bezug auf drei Arten von Äquivalenz zu erzeugen: semantische Äquivalenz, logische Negation und symmetrische Konsistenz. Aus ihren Experimenten schließen die Autoren, dass ChatGPT eine höhere Konsistenz bei negierten Ausdrücken und der Verwendung von Antonymen erreicht, verglichen mit anderen vortrainierten LLMs (wie BERT [17]). Allerdings stellen sie fest, dass dieses LLM in Bezug auf symmetrische und semantische Äquivalenz häufig inkonsistente Antworten liefert. Ihre Ergebnisse legen nahe, dass ChatGPT nicht in der Lage ist, logisch korrekte Antworten zu erzeugen, trotz seines scheinbar hohen Maßes an Sprachverständnis.

Abschließend möchten wir hervorheben, dass Jang und Lukasiewicz [10] bei ihrer Konsistenzanalyse von ChatGPT bzw. GPT-4 [1] feststellen, dass es insbesondere beim Testen der semantischen Konsistenz von LLMs äußerst schwierig ist, alle möglichen Varianten von Formulierungen von Eingabeanfragen abzudecken.

1.3 Kombinatorische Testmethoden für Konsistenztests von LLMs

Kombinatorische Testmethoden, insbesondere t-faches Testen, bieten eine effiziente Möglichkeit, Testdatensätze zu generieren, die eine breite Abdeckung des Eingaberaums gewährleisten. Diese Tests basieren auf der Annahme, dass es ausreichend ist, nur eine bestimmte Anzahl von Kombinationen (t-Werten) zu testen, um die wichtigsten Interaktionen zwischen den Eingabefaktoren abzudecken. Dies reduziert die Anzahl der benötigten Tests erheblich im Vergleich mit dem gesamten Eingaberaums, während dennoch eine ho-

he Abdeckung des möglichen Eingaberaums gewährleistet bleibt [13][5]. Daher werden für die Konsistenztestung von LLMs in KomMKonLLM kombinatorische Tests verwendet, welche sicherstellen, dass die erzeugten Testdatensätze Varianten von Eingabewörtern oder -phrasen bis zu einer vorgegebenen Stärke enthalten. Die Herausforderung besteht darin, dass bei der Veränderung von Wörtern oder Satzstrukturen die ursprüngliche Bedeutung nicht verloren gehen darf. Hierbei kommen Methoden zum Einsatz, die es ermöglichen, Synonyme oder semantisch ähnliche Begriffe zu verwenden, ohne die ursprüngliche Semantik der Eingabefrage zu verfälschen. Weitere Details sind in [7] zu finden.

In den folgenden drei Abschnitten präsentieren wir den in KomMKonLLM implementierten kombinatorischen Testansatz zur Konsistenzevaluierung von LLMs. Hierzu betrachten wir eine entsprechende Instanziierung des CT-Prozesses in Abschnitt 1.3.1, die Generierung von t-fachen Frage-Testdatensätzen in Abschnitt 1.3.2 und widmen uns der Übersetzung und Ausführung der erhaltenen Frage-Testdatensätze in Abschnitt 1.3.3.

1.3.1 Instanziierung des kombinatorischen Testprozesses zur Testung der Konsistenz großer Sprachmodelle

In diesem Abschnitt beschreiben wir die Realisierung eines CT-Prozesses zur Testung der Konsistenz von LLMs.

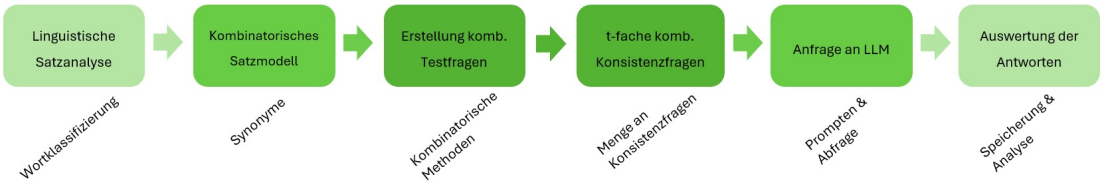

Abbildung 1: Ein Überblick über den CT-Prozess zur Testung der Konsistenz von LLMs.

Um einen Überblick zu geben (siehe dazu auch Fig. 1): Wir beginnen mit einem gegebenen Satz, z. B. aus einem Trainings- oder Evaluationsdatensatz für LLMs. Indem wir jedes *Wort*[6] des gegebenen Satzes als Parameter eines IPM interpretieren, erzeugen wir daraus mehrere abgeleitete Sätze, indem wir Wörter gemäß eines covering arrays durch Synonyme ersetzen. Die abgeleiteten Sätze können anschließend, vermöge eines geigneten Prompt-Designs, an ein LLM als Eingabe übergeben werden. Unter der Annahme, dass der gegebene Satz eine Annotation besitzt - z. B. eine Ja/Nein-Zuweisung zu einer Aussage oder eine korrekte Antwort auf eine Frage - verwenden wir diese Annotation, um auch die abgeleiteten Sätze zu annotieren. Diese Annotationen können anschließend von einem Testorakel verwendet werden, um die Antworten des LLMs auf die abgeleiteten Sätze zu bewerten; anderenfalls werden schlicht die Antworten auf alle (abgeleiteten) Fragen auf ihre Gleichheit überprüft. Annotierte Sätze können innerhalb von Benchmarks bereitgestellt werden, wie zum Beispiel im BoolQ-Benchmark [2]. Um unseren Ansatz zu demonstrieren, wählen wir einen Satz aus diesem Benchmarkset aus und verwenden ihn

[5] Siehe dazu auch `https://www.netidee.at/kommkonllm/kombinatorisches-testen-aller-kuerze`.
[6] Die Aufteilung kann auch nach tokens erfolgen.

in einem fortlaufenden Beispiel. Darüber hinaus zeigt Tabelle 1 die Identifizierung zwischen Begriffen aus dem Bereich der LLMs und jenen aus CT, wie sie durch Anwendung des KomMKonLLM Ansatz entstehen, und im Folgenden erläutert werden.

LLM	**Kombinatorisches Testen**
LLM Instanz	SUT
Satz	IPM
Wort	Parameter
Synonyme	Parameterwerte
Eingabe an die LLM Instanz senden	Ausführung des Tests
Annotation eines Satzes	Erwartetes Test Resultat
Korrekte Rückmeldung der LLM Instanz	Erfolgreicher Test
Falsche Rückmeldung der LLM Instanz	Nicht erfolgreicher Test

Tabelle 1: Übersicht über die Verknüpfung zwischen Konzepten aus dem Bereich der LLMs (links) und aus dem Bereich des CT (rechts) (siehe dazu auch [7]).

Example 1 *Wir veranschaulichen die Schritte unseres Ansatzes anhand eines (fortlaufenden) Beispiels, bei dem wir den folgenden konkreten ursprünglichen Satz betrachten, nämlich die 23. Frage aus dem Trainingsdatensatz des BoolQ-Benchmarks [2]: "can you drink alcohol in public in denmark". Dieser Satz besteht aus acht Wörtern, weshalb das entsprechende IPM acht Parameter hat. Wir setzen eine maximale Anzahl der betrachteten Synonyme v_{max} auf drei. Für die Wörter "can", "you" und "in" wählen wir diese Wörter selbst als ihre einzigen Synonyme aus. Tabelle 2 zeigt die betrachteten Synonyme für jedes Wort dieses Satzes. Die entsprechenden Parameter p_1 bis p_8 können jeweils* $1, 1, 3, 3, 1, 3, 1$ *und* 3 *Werte annehmen. Wir betonen, dass der erste Wert jedes Parameters mit dem entsprechenden Wort im ursprünglichen Satz ident ist.*

1.3.2 Generierung von t-fachen Frage-Testdatensätzen

Wir verwenden ein IPM, wie oben beschrieben, um einen kombinatorischen Testdatensatz zu erzeugen, welcher eine Abdeckung aller t-fachen Kombinationen von Synonymen für eine festgelegte Interaktionsstärke t erreicht. Für praktische Anwendungen gibt es spezielle CT-Tools, wie CAgen [15], bei denen ein konkretes IPM mit definierten Parameterwerten als Eingabe verwendet wird um einen kombinatorischen Testdatensatz mit der gewünschten Interaktionsstärke t zu generieren. Jede Zeile der zurückgegebenen

p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
can	you	drink	alcohol	in	public	in	denmark
		drinking	alcoholic drink		populace		kingdom of denmark
		booze	alcoholic beverage		world		danmark

Tabelle 2: Die ausgewählten Synonyme im Beispiel erhalten die Semantik der Frage weitgehend.

#	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
q_1	can	you	drink	alcohol	in	public	in	denmark
q_2	can	you	drink	alcoholic drink	in	populace	in	kingdom of denmark
q_3	can	you	drink	alcoholic beverage	in	world	in	danmark
q_4	can	you	drinking	alcohol	in	populace	in	danmark
q_5	can	you	drinking	alcoholic drink	in	world	in	denmark
q_6	can	you	drinking	alcoholic beverage	in	public	in	kingdom of denmark
q_7	can	you	booze	alcohol	in	world	in	kingdom of denmark
q_8	can	you	booze	alcoholic drink	in	public	in	danmark
q_9	can	you	booze	alcoholic beverage	in	populace	in	denmark

Tabelle 3: Ein kombinatorische Testmenge der Stärke $t = 2$ für die test frage *"can you drink alcohol in public in denmark"* und das entsprechende IPM mit acht Parametern. Es enthält abgeleitete Fragen, wobei die ursprüngliche Frage als erste Zeile q_1 auftritt.

(abstrakten) kombinatorischen Testsuite stellt dann einen abgeleiteten Satz dar und entspricht somit einem (abstrakten) Testfall. Die Tatsache, dass das erste Element jeder Synonymliste das ursprüngliche Wort ist, führt - zusammen mit den Eigenschaften des CAgen-Tools - dazu, dass der erste abgeleitete Satz in der kombinatorischen Testsuite dem ursprünglichen Satz gleicht – zumindest mit geringfügigen Abweichungen wie ersetzten Trennzeichen oder Groß-/Kleinschreibung, die durch Vorverarbeitung entstehen können. Insgesamt besitzen alle abgeleiteten Sätze in einem erzeugten kombinatorischen Testdatensatz die Eigenschaft, dass jede Kombination von t Synonymen für t unterschiedliche Wörter in mindestens einem Testfall vorkommt.

Example 1 *(Fortsetzung) Das in Tabelle 2 dargestellte IPM kann nun verwendet werden, um einen paarweise kombinatorischen Testdatensatz zu erzeugen, der in Tabelle 3 gezeigt wird und aus der Beispielfrage abgeleitet wurde. Die Spaltenüberschriften p_1 bis p_8 stehen für die Parameter bzw. die Wörter, und jede Zeile q_1 bis q_9 stellt einen Testfall dar, also eine kombinatorisch abgeleitete Frage. Insgesamt erreichen alle diese abgeleiteten Fragen zusammen in dem kombinatorischen Testdatensatz eine vollständige 2-fache Abdeckung, d. h. sie gewährleisten, dass jedes Wortpaar von Synonymen welche aus zwei verschiedenen Wörtern des Beispielsatzes hervorgehen, in mindestens einer der abgeleiteten Fragen gemeinsam vorkommt. Zur Veranschaulichung dieser paarweisen Abdeckungseigenschaft betrachten wir die Parameter p_4 und p_8, bei denen jede Kombination aus dem kartesischen Produkt $\{alcohol, alcoholic\ drink, alcoholic\ beverage\} \times \{denmark, kingdom\ of\ denmark, danmark\}$ in mindestens einem der Tests q_1 bis q_9 vorkommt. Betrachten wir zum Beispiel das Paar $(booze, kingdom\ of denmark)$, d. h. das Synonym booze für das ursprüngliche Wort drink und das Synonym kingdom of denmark für das ursprüngliche Wort denmark, so sehen wir, dass es in der siebten abgeleiteten Frage q_7 vorkommt. Auf gleiche Weise können die übrigen paarweisen Abdeckungsanforderungen überprüft werden.*

1.3.3 Übersetzung und Ausführung der Frage-Testdatensätzen

Die generierten kombinatorischen Testdatensätze dienen nun als Grundlage für die Testausführung gegen LLM Instanzen. Jede Zeile eines kombinatorischen Testdatensatzes wird dabei in einen Testfall übersetzt, d. h. eine Abfrage an ein LLM. Um aus den abgeleiteten Sätzen ausführbare Testfälle zu erhalten, die eine auswertbare Antwort liefern, ist ein geeignetes Prompt-Design erforderlich. In diesem Schritt werden die Prompts so angepasst, dass eine Boolesche Antwort erzwungen wird. Wir verweisen Interessierte auf Arbeiten die sich ausführlich mit diesem Thema beschäftigen [19], [14]. Der generierte ausführbare Testfall (d. h. die abgeleitete Frage kombiniert mit einem passenden Prompt) wird zusammen mit potenziellen weiteren Konfigurationswerten an eine LLM Instanz übermittelt.

Example 1 *(Fortsetzung) Wir übersetzen die erste abgeleitete Frage d. h. Testfall q_1 aus Tabelle 3, in einen ausführbaren Testfall. Konkret wird der Frage "can you drink alcohol in public in denmark" ein modell-spezifisches Suffix zum Prompt hinzugefügt, um einen ausführbaren Testfall zu erhalten; dieser lautet:*

```
can you drink alcohol in public in denmark? Return a
JSON Boolean.
```
(1)

1.3.4 Testorakel und Auswertung

Unter der Annahme, dass die Synonymersetzung die Semantik der ursprüglichen Frage erhält, entspricht die Bedeutung der abgeleiteten Fragen in einem kombinatorischen Frage-Testdatensatz der Bedeutung der ursprünglichen Frage. Basierend auf dieser Annahme kann ein Testorakel für jede abgeleiteten Fragen erstellt werden, wenn für die ursprüngliche Frage eine Annotation vorhanden ist. Im speziellen Fall, dass die ursprüngliche Frage eine boolesche Frage ist, deren Annotation durch eine Ja/Nein-Zuweisung gegeben ist, kann auf folgende Weise ein Testorakel erstellt werden:
Wird ein Testfall gegen ein LLM ausgeführt, erwarten wir, dass das LLM sich mit dem Wahrheitsgehalt der Testfrage auseinandersetzt. Daher sollte es möglich sein, die Antwort in eine der drei Klassen *ja*, *nein* oder *undefiniert* (bzw. `true`, `false` und `undefined`) einzuordnen. Diese Klassifizierung erfolgt durch eine textuelle Nachverarbeitung mittels Schlüsselwörtern in der zurückgegebenen Antwort. Das Testoracle vergleicht dann die zugewiesene Klasse der LLM-Antwort mit der gegebenen Annotation der ursprünglichen Booleschen Frage: Stimmt die Klasse der Antwort mit der Ja/Nein-Annotation der ursprünglichen Frage überein, wird die abgeleitete Frage als bestandener Testfall gewertet, andernfalls als fehlgeschlagener Testfall.

Example 1 *(Fortsetzung) Das Prompt aus* (1) *übergeben an DeepSeek liefert Folgendes:*

```
> can you drink alcohol in public in denmark? Return a
 JSON Boolean.
» True
```
(2)

#	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	DeepSeek	Llama 3.2
q_1	can	you	drink	alcohol	in	public	in	denmark	`True`	`False`
q_2	can	you	drink	alcoholic drink	in	populace	in	kingdom of denmark	`True`	`True`
q_3	can	you	drink	alcoholic beverage	in	world	in	danmark	`undefined`	`True`
q_4	can	you	drinking	alcohol	in	populace	in	danmark	`True`	`True`
q_5	can	you	drinking	alcoholic drink	in	world	in	denmark	`True`	`True`
q_6	can	you	drinking	alcoholic beverage	in	public	in	kingdom of denmark	`True`	`False`
q_7	can	you	booze	alcohol	in	world	in	kingdom of denmark	`True`	`True`
q_8	can	you	booze	alcoholic drink	in	public	in	danmark	`True`	`False`
q_9	can	you	booze	alcoholic beverage	in	populace	in	denmark	`True`	`True`

Tabelle 4: Ein kombinatorische Testmenge der Stärke $t = 2$ für die test frage *"can you drink alcohol in public in denmark"* und das entsprechende IPM mit acht Parametern. Es enthält abgeleitete Fragen, wobei die ursprüngliche Frage als erste Zeile q_1 auftritt. Die korrekte Antwort, also die Annotation, ist `True`.

Daher wird die Antwort als `true` *klassifiziert. Ein weiteres Beispiel: Die Ausführung des Testfalls aus der abgeleiteten Frage q_3 aus Tabelle 3 zusammen mit der Antwort von DeepSeek:*

```
> can you drink alcoholic beverage in world in danmark?
  Return a JSON Boolean.
» {"response":{"canDrinkAlc":true,"message":"Yes, you can
  drink alcoholic beverages in Denmark."}}
```
(3)

Nachdem die Rückgabe in (3) *kein JSON Boolean ist, wird sie als* `undefined` *kategorisiert.*

Interpretation Die in Tabelle 4 angegebenen, kategorisierten Antworten der LLMs DeepSeek und Llama 3.2 lassen die folgenden Schlüsse zu:

- DeepSeek antwortet auf die abgeleiteten Fragen zu "can you drink alcohol in public in denmark" weitestgehend konsistent, trotz Ersetzung mehrerer Wörter durch Synonyme. Das gilt insbesondere für die Ersetzung von Wörtern durch Paare von Synonymen, die für die Ersetzung betrachtet wurden, da dem in Tabelle 4 angegebenen Testdatensatz ein CA der Stärke zwei zugrunde liegt.
- Alle Antworten von Llama 3.2 können als `True`/`False` klassifiziert werden, d. h. die Rückgabevorgabe eines JSON Boolean wurde eingehalten. Allerdings variieren die erhaltenen Wahrheitswerte deutlich, sodass sich Llama 3.2 in einem gewissen Sinne *weniger konsistent* verhält als DeepSeek.

2 User Anleitung

Der im vorherigen Kapitel beschriebene Testprozess basierend auf Methoden des kombinatorischen Testens wurde mittels Python implementiert und iwird auf GitHub[7] unter der MIT Lizenz zur Verfügung gestellt. Eine englische Kurzfassung[8] der User Anleitung ist dort auch enthalten.

Abbildung 2 gibt einen Überblick über die Systemarchitektur von KomMKonLLM, wobei die dargestellten Komponenten folgende Aufgaben haben:

- Ein zentraler Runner, der den Prozess steuert;
- Eine Modell-Schnittstelle, die eine einheitliche Kommunikation mit LLMs ermöglicht;
- Ein externer Covering-Array-Generator;
- Ein Set von zu evaluierenden Fragen;
- Eine Datenbank mit Synonymen, bereitgestellt durch enthaltene Bibliotheken;
- Eine Datenbank zur Speicherung der Testergebnisse.

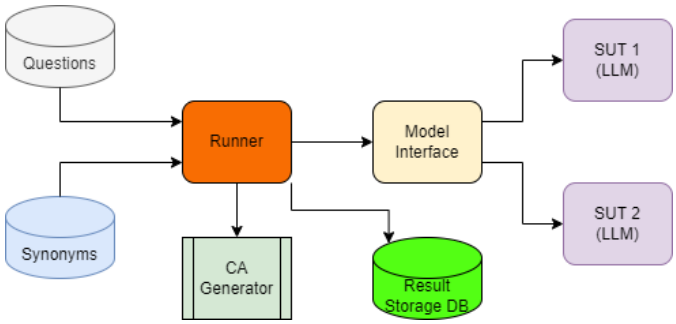

Abbildung 2: Systemarchitektur von KomMKonLLM.

2.1 Einrichtung

Um die KomMKonLLM-Software auszuführen, wird Docker und Docker Compose benötigt. Neuere Versionen von Docker enthalten Compose standardmäßig; für ältere Versionen muss das `docker-compose` Plugin installiert werden. Die Leistungsanforderungen hängen hauptsächlich vom verwendeten LLM ab. In der Regel sollten 10GB freier Speicherplatz für kleine Modelle ausreichen.

[7] `https://github.com/KomMKonLLM/KomMKonLLM`
[8] `https://github.com/KomMKonLLM/KomMKonLLM/blob/main/README.md`

2.1.1 Erstellung

Um die Umgebung zu erstellen (entweder initial oder nach Änderungen am Quellcode), muss das KomMKonLLM Repository geklont oder (neu-) heruntergeladen werden. In dem Ordner, der diese Dateien enthält, müssen folgende Befehle ausgeführt werden:

```
# Neuere Versionen von Docker
docker compose build

# Aeltere Versionen
docker-compose build
```

2.1.2 Konfiguration

Konfigurationsoptionen werden in der Datei `docker-compose.yaml` über Umgebungsvariablen definiert. Die wichtigsten sind:

- `CONTINUE_RUN`: Wenn auf `true` gesetzt, versucht das Framework, den letzten getesteten Satz aus dem vorherigen Testlauf zu identifizieren und dort fortzufahren.
- `EXECUTION_NOTE`: Eine benutzerdefinierte Notiz, die allen getesteten Sätzen hinzugefügt wird.
- `MODEL_UNDER_TEST`: Welches Modellinterface verwendet werden soll (`OLLAMA`, `T5`, oder `LLAMA`).
- `USE_POSTFIX` und `USE_PREFIX`: Ob ein bestimmter String an Abfragen angehängt/vorangestellt werden soll.
- `CA_GENERATOR`: Welcher Covering-Array-Generator verwendet wird (`CAGEN`, `ACTS`, `PICT`).

Außerdem müssen die zu testenden Sätze als `/app/train.jsonl` im `meta_runner`-Container eingebunden werden. Diese Datei kann entweder direkt bearbeitet (und dabei ist sicherstellen, dass sie nicht im Abschnitt `volumes` des `meta_runner`-Containers überschrieben wird) oder eine eigene Datei erstellt und sie als Volume einbunden werden; z. B.:

```
volumes:
    - ./my-own-questions.jsonl:/app/train.jsonl
```

Im Abschnitt 2.1.4 wird das Format der .jsnl Datei beschrieben. Je nach zu testendem LLM müssen möglicherweise zusätzliche Container hinzugefügt oder konfiguriert werden. Für den `executor`-Container muss die Umgebungsvariable `MODEL_IP` auf den Hostnamen und Port des jeweiligen LLMs verweisen, mit dem kommuniziert werden soll (in der Regel ist dies der Name eines anderen Containers plus der Port, auf dem das LLM lauscht).

Falls `MODEL_UNDER_TEST` auf `OLLAMA` gesetzt ist, benötigt der `executor`-Container zusätzlich die Umgebungsvariable `OLLAMA_MODEL`, die auf den Namen eines Modells gesetzt

werden muss, z. B. ‘llama3.2‘. Eine Liste darin verfügbarer LLMs ist im Ollama GitHub-Repository oder auf der speziellen Ollama-Bibliotheksseite zu finden.

2.1.3 CA-Generatoren

Unser Framework unterstützt drei Covering-Array-Generatoren:

- **CAgen**: Dies ist der schnellste allgemeine CA-Generator. Eine Web-Version ist hier verfügbar, jedoch ist sie nur einthreadig. Die Befehlszeilenversion ist jedoch nur auf Anfrage erhältlich.
- **ACTS**: Entwickelt von NIST, ist ACTS ein weit verwendeter CA-Generator. Eine Basisversion ist auf der NIST-Seite für kombinatorische Testwerkzeuge verfügbar, während eine erweiterte Version auf Anfrage bereitgestellt wird.
- **PICT**: Microsofts Pairwise Independent Combinatorial Tool ist auf GitHub verfügbar. Es eignet sich hauptsächlich für Testen mit Stärke 2, unterstützt jedoch höhere Stärken mit reduzierter Leistung. Es ist zu beachten, dass die ausführbare Datei selbst gemäß den Anweisungen im Repository erstellt werden muss.

Die ausführbare Datei (wahrscheinlich `cagen`, `fipo-cli`, `acts.jar` oder `pict`) muss in einem Unterordner des `meta-runner`-Ordners abgelegt werden und dann die Umgebungsvariablen `CA_GENERATOR` und `CA_GENERATOR_PATH` in der Datei `docker-compose.yaml` entsprechend angepasst werden.

2.1.4 Fragen/Antworten

Das KomMKonLLM-Framework erfordert eine Liste von Fragen mit den zugehörigen Antworten in einer Textdatei. Diese Datei muss eine JSON-Zeile pro Zeile enthalten, wobei jedes Objekt die folgenden Eigenschaften haben muss:

- `question`: Ein String, der eine Frage definiert.
- `answer`: Die korrekte Antwort. *Hierbei ist zu beachten, dass derzeit nur Ja/Nein-Fragen unterstützt werden.* Diese Eigenschaft sollte daher ein Boolean sein und kein String.
- `passage`: Zusätzliche Informationen zur Erklärung der Antwort. Hier kann ein leerer String verwendet werden.

Ein Beispieldatensatz von Fragen und Antworten (zusammen mit deren Analyse) ist unter `https://zenodo.org/records/15209547` zu finden.

2.2 Ausführung

Nach der Konfiguration wird das Framework wiefolg gestartet:

```
# Neuere Versionen von Docker
docker compose up --abort-on-container-exit

# Aeltere Versionen
docker-compose up --abort-on-container-exit
```

Über die Konsolenausgabe kann der Testprozess in Aktion betrachtet werden.

2.3 Analyse

Nach Abschluss der Testausfürung können die Ergebnisse mit einem bereitgestellten JupyterLab analysiert werden, siehe Abbildung 3. Der Zugriff auf JupyterLab erfolgt über die angegebene URL im Output der Konsole nachdem die folgenden Befehle ausgeführt wurden:

```
docker start db
docker start -a store
```

Das JupyterLab Notebook enthält bereits Beispielcode zur Analyse und kann beliebig erweitert werden. Die Kommentare im JupyterLab Notebook geben die dafür nötigen Hinweise.

3 Kommentare und Kontaktinformation

Für Fragen kontaktieren Sie uns per E-Mail (`kommkonllm at sba-research.org`) oder erstellen Sie ein Issue im Repository. Des weiteren finden Sie dieses Projekt in den sozialen Medien: KomMKonLLM@Bluesky, sowie Informationen über die MATRIS Forschungsgruppe unter `https://matris.sba-research.org/`.

Danksagung

KomMKonLLM ist ein Projekt, das von der Internetstiftung Österreich als *netidee Projekt* des Förderjahres 2024 im Rahmen des *Call #19* mit ProjektId 7409 gefördert wird.

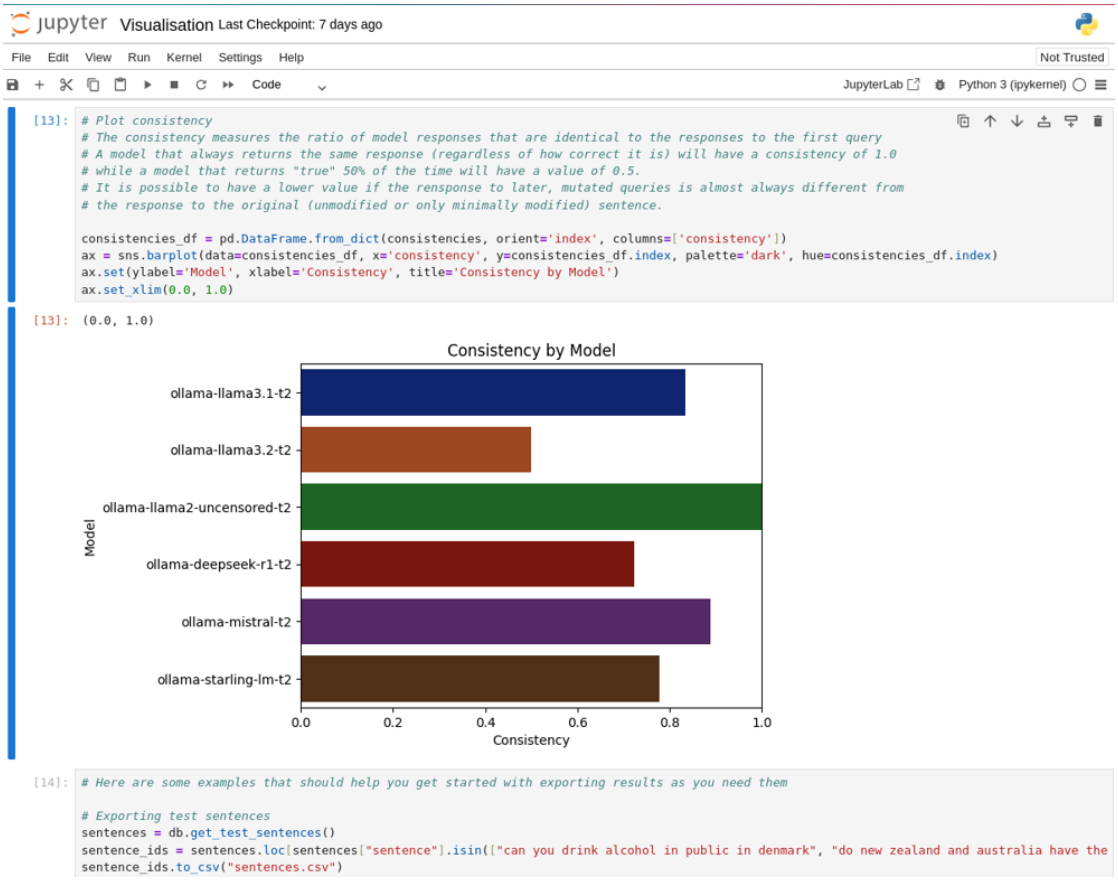

Abbildung 3: Automatisierte Analyse mittels JupyterLab.

Literatur

[1] Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y.T., Li, Y., Lundberg, S., Nori, H., Palangi, H., Ribeiro, M.T., Zhang, Y.: Sparks of artificial general intelligence: Early experiments with GPT-4. Available online at `https://doi.org/10.48550/arXiv.2303.12712` (Apr 2023)

[2] Clark, C., Lee, K., Chang, M.W., Kwiatkowski, T., Collins, M., Toutanova, K.: BoolQ: Exploring the surprising difficulty of natural yes/no questions. In: Proc. of the 2019 Conf. of the North American Chapter of the Assoc. for Comp. Linguistics: Human Language Technologies, Vol 1. pp. 2924–2936. Assoc. for Comp. Linguistics, Minneapolis, Minnesota (Jun 2019)

[3] De Visser, E.J., Monfort, S.S., McKendrick, R., Smith, M.A., McKnight, P.E., Krueger, F., Parasuraman, R.: Almost human: Anthropomorphism increases trust resilience in cognitive agents. Journal of Experimental Psychology: Applied **22**(3), 331–349 (2016). https://doi.org/https://doi.org/10.1037/xap0000092

[4] Dunn, P.: The importance of consistency in establishing cognitive-based trust: A laboratory experiment. Teaching Business Ethics **4**, 285–306 (2022)

[5] European Commission: Ethics guidelines for trustworthy AI. Available online at `https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai` (2019), accessed 2023-04-24

[6] Gardner, M., Artzi, Y., Basmova, V., Berant, J., Bogin, B., Chen, S., Dasigi, P., Dua, D., Elazar, Y., Gottumukkala, A., Gupta, N., Hajishirzi, H., Ilharco, G., Khashabi, D., Lin, K., Liu, J., Liu, N., Mulcaire, P., Ning, Q., Singh, S., Smith, N., Subramanian, S., Tsarfaty, R., Wallace, E., Zhang, A., Zhou, B.: Evaluating models' local decision boundaries via contrast sets. In: Findings of the Association for Computational Linguistics Findings of ACL. pp. 1307–1323. Findings of the Association for Computational Linguistics Findings of ACL: EMNLP 2020, Association for Computational Linguistics (ACL) (2020)

[7] Garn, B., Kampel, L., Leithner, M., Celic, B., Çulha, C., Hiess, I., Kieseberg, K., Koelbing, M., Schreiber, D.P., Wagner, M., Wech, C., Zivanovic, J., Simos, D.E.: Applying pairwise combinatorial testing to large language model testing. In: Testing Software and Systems. pp. 247–256. Springer Nature Switzerland, Cham (2023)

[8] High-Level Expert Group on Artificial Intelligence (AI HLEG): The assessment list for trustworthy artificial intelligence (ALTAI). Available online at `https://ec.europa.eu/newsroom/dae/document.cfm?doc_id=68342` (2020). https://doi.org/doi:10.2759/002360, accessed 2023-04-24

[9] Hoffmann, C.H.: Is AI intelligent? An assessment of artificial intelligence, 70 years after Turing. Technology in Society **68**, 101893 (2022)

[10] Jang, M., Lukasiewicz, T.: Consistency analysis of ChatGPT. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. pp. 15970–15985. Association for Computational Linguistics, Singapore (Dec 2023). https://doi.org/10.18653/v1/2023.emnlp-main.991, `https://aclanthology.org/2023.emnlp-main.991`

[11] Jung, E.S., Dong, S.Y., Lee, S.Y.: Neural correlates of variations in human trust in human-like machines during non-reciprocal interactions. Scientific Reports **9**(1), 9975 (2019)

[12] Khashabi, D., Khot, T., Sabharwal, A.: More bang for your buck: Natural perturbation for robust question answering. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 163–170. Association for Computational Linguistics, Online (Nov 2020)

[13] Kuhn, D., Kacker, R., Lei, Y.: Introduction to Combinatorial Testing. Chapman & Hall/CRC Innovations in Software Engineering and Software Development Series, Taylor & Francis, 1st edn. (2013)

[14] Li, Y.: A practical survey on zero-shot prompt design for in-context learning. In: Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing. pp. 641–647. INCOMA Ltd., Shoumen, Bulgaria, Varna, Bulgaria (Sep 2023), https://aclanthology.org/2023.ranlp-1.69

[15] MATRIS: CAgen. Available online at https://matris.sba-research.org/tools/cagen, Accessed on 2025-01-01

[16] Sanderson, K.: GPT-4 is here: what scientists think. Nature **615**(7954), 773 (2023)

[17] Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for deep learning in NLP. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 3645–3650. Association for Computational Linguistics, Florence, Italy (Jul 2019)

[18] Wotawa, F.: On the use of available testing methods for verification & validation of AI-based software and systems. CEUR Workshop Proceedings **2808** (2021)

[19] Zamfirescu-Pereira, J., Wong, R.Y., Hartmann, B., Yang, Q.: Why Johnny can't prompt: How non-AI experts try (and fail) to design LLM prompts. In: Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems. CHI '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3544548.3581388, https://doi.org/10.1145/3544548.3581388