Grado en Ingeniería en Tecnología de Telecomunicaciones

Trabajo de Fin de Grado

Detección de alucinaciones en modelos del lenguaje grandes en el ámbito médico

Autor
Javier Díaz de Rábago Pemán

Director
David Contreras Bárcena

Madrid
Julio 2024

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Detección de alucinaciones en modelos del lenguaje grandes en el ámbito médico

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2023/2024 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio

de otro, ni total ni parcialmente y la información que ha sido tomada

de otros documentos está debidamente referenciada.

Fdo.: Javier Díaz de Rábago Pemán   Fecha: 02/07/2024

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: David Contreras Bárcena   Fecha: 03/07/2024

Grado en Ingeniería en Tecnología de Telecomunicaciones

Trabajo de Fin de Grado

Detección de alucinaciones en modelos del lenguaje grandes en el ámbito médico

Autor
Javier Díaz de Rábago Pemán

Director
David Contreras Bárcena

Madrid
Julio 2024

Quiero agradecer a mis padres su apoyo incondicional durante todos estos años de mi trayectoria académica y personal. Gracias a ellos he llegado hasta donde estoy.

A mis compañeros, muchas gracias por vuestra amistad y apoyo durante toda la carrera. Han sido cuatro años llenos de experiencias tanto dentro como fuera de la universidad.

Gracias también a todos los amigos que he conocido en Delft, por hacer de este año uno de los mejores de mi vida. Muchas gracias especialmente por mantenerme animado durante estas últimas semanas de tanto trabajo.

Por último, quiero dar las gracias a David, mi director, por contar conmigo para este proyecto y por sus consejos e ideas a lo largo de todo el proceso.

Gracias a todos por creer en mí y por ayudarme a alcanzar mis metas. Este logro es tanto mío como vuestro.

DETECCIÓN DE ALUCINACIONES EN MODELOS DEL LENGUAJE GRANDES EN EL ÁMBITO MÉDICO

Autor: Díaz de Rábago Pemán, Javier
Director: Contreras Bárcena, David
Entidad Colaboradora: ICAI — Universidad Pontificia Comillas

Resumen

Desde el lanzamiento de ChatGPT en noviembre de 2022, los grandes modelos del lenguaje (LLMs, por sus siglas en inglés) han demostrado un gran potencial en diversas aplicaciones, desde redacción de texto y generación de ideas hasta programación. Sin embargo, estos modelos tienden a generar información falsa, conocida como alucinación, lo que dificulta su adopción en ámbitos sensibles como el de la salud. Este trabajo se centra en desarrollar y evaluar técnicas de detección de alucinaciones en las respuestas de los LLMs, particularmente en el campo médico.

Diseñamos y evaluamos varios métodos de detección, incluyendo Verificación Contextual Básica (BCV), BCV con cadenas de pensamiento (BCV-CoT), Análisis Contextual por Oraciones (SCA), Contraste Iterativo de Oraciones y Muestras (ISSC) e ISSC-CoT, en datasets tanto generales como de ámbito médico. En cada benchmark, comparamos varios modelos juez como llama2, llama3, nous-hermes2, gemma y mistral.

Los resultados muestran que BCV-CoT, particularmente con nous-hermes2 como juez, alcanza el mejor rendimiento en términos de sensibilidad y especificidad. Asignar un rol de experto al modelo juez puede mejorar aún más la precisión en la detección. Estos resultados subrayan la importancia de la ingeniería de prompts y demuestran el potencial de las herramientas basadas en LLMs para aplicaciones médicas.

Palabras clave: grandes modelos del lenguaje, alucinación, cadena de pensamiento, ingeniería de prompts

Introducción

Desde el lanzamiento de ChatGPT en noviembre de 2022, los LLMs han demostrado su potencial en diversas aplicaciones como la redacción de texto, la generación de ideas y la programación. Sin embargo, este progreso tan rápido ha generado debates sobre cuestiones legales, éticas y prácticas, incluyendo dudas sobre derechos de autor y la generación de desinformación.

Esta investigación se centra en un problema muy común con los LLMs: las alucinaciones. Las alucinaciones, o la generación de información falsa por parte de los modelos, representan un desafío especialmente en dominios sensibles como la salud. Los últimos avances técnicos para detectar estas alucinaciones utilizan un segundo LLM para evaluar la veracidad del texto generado. Así lo hacen ChainPoll [1] (que utiliza cadenas de pensamiento), Rowen [2] (que evalúa la consistencia semántica entre idiomas) y SelfCheckGPT [3] (que contrasta múltiples respuestas alternativas).

Para evaluar la efectividad de estos métodos de detección, se emplean varios benchmarks como RealHall [1], HaluEval [4] o TruthfulQA [5]. En el ámbito médico, se ha investigado muy poco en la detección de alucinaciones. Med-HALT [6] es el único benchmark relativamente completo en este campo, y sus resultados están lejos de ser prometedores, dejando mucho margen para posibles mejoras.

El objetivo principal del proyecto es desarrollar, ajustar y evaluar técnicas automáticas de detección de alucinaciones adaptadas al ámbito médico. En última instancia, la meta es facilitar la toma de decisiones clínicas, garantizar la salud del paciente, promover el uso ético de la IA y la adopción de herramientas basadas en IA en aplicaciones reales.

Metodología

Este proyecto evalúa varios métodos de detección de alucinaciones y modelos juez en conjuntos de datos tanto de conocimiento general como de dominio médico. HaluEval incluye un contexto para cada par pregunta-respuesta. También incluye respuestas alucinadas, lo que facilita la creación de una test de detección. TruthfulQA, en lugar de contexto, incluye múltiples muestras para cada respuesta fáctica y alucinada. En el ámbito médico, MedQuad [7] recopila información de fuentes oficiales, aunque requiere cierto preprocesamiento, como limpiar los datos crudos y generar respuestas alucinadas. Finalmente, EHRTest contiene pares de preguntas y respuestas generados a partir de una serie de expedientes clínicos electrónicos anonimizados.

Dentro de las limitaciones del hardware disponible, se seleccionaron los siguientes modelos abiertos para actuar como LLM juez: llama2 (7B), llama2:13b, llama3 (8B), llama3-gradient, nous-hermes2, gemma y mistral.

Incorporando ideas de diferentes técnicas, se diseñaron una serie de métodos para detección de alucinaciones. La Verificación Contextual Básica (BCV) simplemente pide al modelo juez que contraste un par pregunta-respuesta con un contexto dado. BCV-CoT añade la técnica de chain-of-thought (CoT), donde se pide al modelo que explique su proceso de razonamiento antes de emitir un veredicto final. El Análisis Contextual por Oraciones (SCA) usa una cadena de pensamiento más compleja, donde se le pide al modelo que divida la respuesta dada en oraciones y encuentre un extracto del contexto que apoye o contradiga cada oración. En el Contraste Iterativo de Oraciones y Muestras (ISSC), el modelo repetidamente recibe una oración de la respuesta y se le pide que la contraste con otra respuesta muestreada. Por último, también se implementa una alternativa con CoT de este último método, ISSC-CoT.

Este proyecto se llevará a cabo en dos fases principales. En la primera parte, se utilizarán técnicas de ingeniería de prompts y pruebas manuales para generar y refinar plantillas de prompts para los diferentes métodos de detección de alucinaciones. A continuación, se evaluarán con los diferentes datasets y modelos juez para encontrar las combinaciones más efectivas.

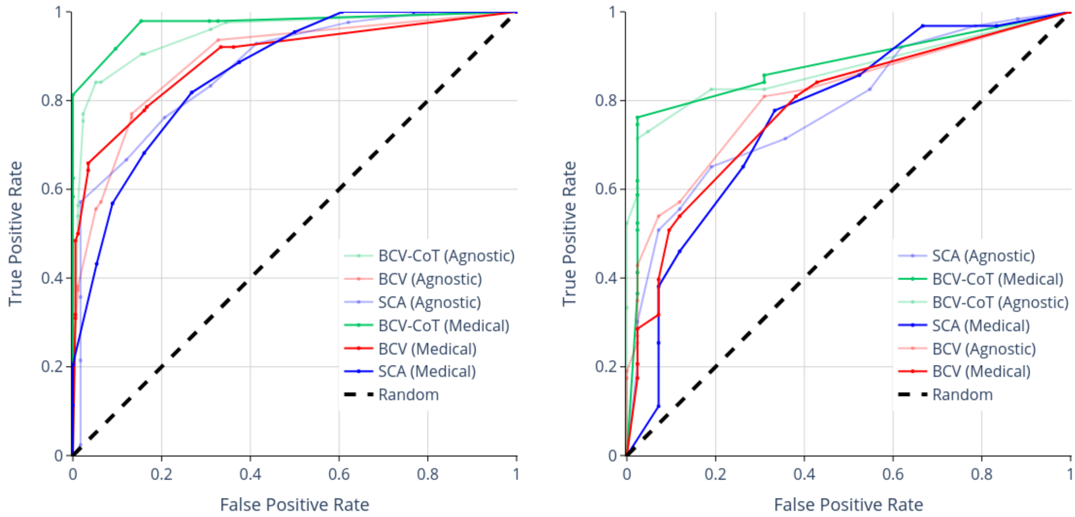
Resultados

La primera fase del proyecto destacó varios de los problemas que surgen al diseñar prompts para LLM. Por ejemplo, uno de los desafíos fue obtener respuestas consistentes con un formato fácil de analizar. Otros problemas que se tuvieron que resolver mediante una cuidadosa ingeniería de prompts incluyeron las alucinaciones de temática fantástica (muy evidentes) al crear tests y la tendencia a agregar una advertencia siempre que se genera una alucinación.

Para cada benchmark, se seleccionaron los métodos de detección más apropiados según la estructura de los datos. Por ejemplo, HaluEval se probó con BCV y BCV-CoT. Aunque nous-hermes2 y llama3 lograron buenos resultados, estas pruebas mostraron que algunos modelos tienen un sesgo en sus veredictos, prefiriendo juzgar todos los casos como fáctico, o como alucinado. TruthfulQA, que contiene muestras adicionales para cada par de preguntas y respuestas, parecía ideal para probar los métodos ISSC e ISSC-CoT. Sin embargo, los resultados fueron mucho menos prometedores que con BCV(-CoT). Dado el alto número de inferencias necesarias para probar un solo ejemplo y su bajo rendimiento, ISSC(-CoT) se descartó de las siguientes pruebas.

En los datasets médicos, MedQuad y EHRTest, las respuestas más largas permitieron evaluar SCA además de BCV(-CoT). En ambos casos, solo nous-hermes2 funcionó razonablemente bien sin CoT. En BCV-CoT, nous-hermes2, llama3 y mistral lograron una sensibilidad y especificidad relativamente altas. Añadir CoT también ayudó a reducir el sesgo en los veredictos. Por último, SCA mostró mejores resultados que BCV, pero pobres comparados con los de BCV-CoT. Las instrucciones y el formato más complejos requeridos por SCA llevaron a diferentes errores en el razonamiento de los modelos. Además, requiere prompts y respuestas notablemente más extensas.

Para los diferentes datasets médicos y métodos de detección, se ha evaluado una segunda versión de cada prompt donde se asigna al modelo juez el papel de un experto médico. La Figura 1 muestra las curvas ROC resultantes de usar nous-hermes2 como juez. En estos gráficos, BCV-CoT claramente destaca como la técnica de detección más efectiva y parece mejorar al agregar el rol de experto. Sin embargo, el efecto de este rol en las otras pruebas es más ambiguo.



(a) EHRTest (b) MedQuad

?figurename? 1: Efectos del rol de experto en nous-hermes2

Conclusiones

Este proyecto ha tenido limitaciones de hardware, que no han permitido utilizar los modelos más recientes con decenas de miles de millones de parámetros. Incluso con modelos más pequeños, para cada uno se han llevado a cabo dieciséis pruebas diferentes, ejecutadas secuencialmente durante un total de más de 110 horas de computación. Con hardware más capaz y una inferencia más optimizada, las técnicas propuestas de detección de alucinaciones podrían probarse en modelos como llama3:70b o mixtral:8x22b.

Durante el desarrollo de las plantillas, se tuvieron que superar muchos de los desafíos típicos en la ingeniería de prompts. El sesgo de veredicto y las inconsistencias de formato se trasladaron a algunos de los tests a gran escala en modelos menos capaces.

Los resultados muestran que la mejor combinación para la detección de alucinaciones es la Verificación Contextual Básica con un enfoque CoT, con nous-hermes2 como modelo juez. A futuro, posibles mejoras serían hacer pruebas en modelos más grandes y potentes, así como la adaptación de las plantillas de prompt específicamente para cada modelo juez. Se podría desarrollar un filtro de alucinaciones para facilitar la implementación de LLMs en un asistente médico virtual o en otras aplicaciones del mundo real.

Referencias

[1] Robert Friel y Atindriyo Sanyal. *Chainpoll: A high efficacy method for LLM hallucination detection*. 2023. arXiv: `2310.18344 [cs.CL]`.

[2] Hanxing Ding et al. *Retrieve Only When It Needs: Adaptive Retrieval Augmentation for Hallucination Mitigation in Large Language Models*. 2024. arXiv: `2402.10612 [cs.CL]`.

[3] Potsawee Manakul, Adian Liusie y Mark J. F. Gales. *SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models*. 2023. arXiv: `2303.08896 [cs.CL]`.

[4] Junyi Li et al. *HaluEval: A Large-Scale Hallucination Evaluation Benchmark for Large Language Models*. 2023. arXiv: `2305.11747 [cs.CL]`.

[5] Stephanie Lin, Jacob Hilton y Owain Evans. *TruthfulQA: Measuring How Models Mimic Human Falsehoods*. 2022. arXiv: `2109.07958 [cs.CL]`.

[6] Ankit Pal, Logesh Kumar Umapathi y Malaikannan Sankarasubbu. *Med-HALT: Medical Domain Hallucination Test for Large Language Models*. 2023. arXiv: `2307.15343 [cs.CL]`.

[7] Asma Ben Abacha y Dina Demner-Fushman. "A Question-Entailment Approach to Question Answering". En: *BMC Bioinform.* (). URL: `https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-019-3119-4`.

MEDICAL-DOMAIN HALLUCINATION DETECTION IN LARGE LANGUAGE MODELS

Author: Díaz de Rábago Pemán, Javier
Supervisor: Contreras Bárcena, David
Collaborating Entity: ICAI — Universidad Pontificia Comillas

Abstract

Since the release of ChatGPT in November 2022, large language models (LLMs) have shown significant potential across various applications, including writing, idea generation, and coding. These models offer a wide range of benefits, but they also sometimes generate false information, known as hallucinations, posing challenges to wide adoption, especially in sensitive domains like healthcare. This research focuses on developing and benchmarking techniques for detecting hallucinations in LLM outputs, particularly in the medical field.

To evaluate the effectiveness of different hallucination detection methods, we benchmarked several approaches, including Basic Contextual Verification (BCV), BCV with chain-of-thought (BCV-CoT), Sentence-level Contextual Analysis (SCA), Iterative Sentence-Sample Comparison (ISSC), and ISSC-CoT, across both general knowledge and medical datasets. Each benchmark was conducted using various judge models, such as llama2, llama3, nous-hermes2, gemma, and mistral.

Our findings indicate that BCV-CoT, especially when used with the nous-hermes2 model, achieves the highest sensitivity and specificity in detecting hallucinations. Additionally, incorporating a medical expert role in the prompts can further enhance performance. These results underscore the importance of tailored prompt engineering and demonstrate the potential for improving LLM-based tools for medical applications. By refining these techniques, we can significantly enhance the reliability and usefulness of LLMs in critical fields such as healthcare.

Keywords: Large Language Model, hallucination, Chain of Thought, prompt engineering

Introduction

Since the release of ChatGPT in November 2022, LLMs have demonstrated their potential in various applications such as writing, idea generation, and coding. However, this rapid progress has sparked debates about legal, ethical, and practical issues, including copyright concerns and the generation of misinformation.

This research focuses on a very common problem with LLMs — hallucinations. Hallucinations, or the generation of false information by LLMs, pose a significant challenge, especially in sensitive domains like healthcare. SOTA techniques for detecting these hallucinations make use of a second LLM to evaluate the factuality in the generated text, this is the case of ChainPoll[1] (which leverages chain-of-thought prompting), Rowen[2] (semantic consistency accross languages) and SelfCheckGPT[3] (constrasting multiple sampled responses).

To evaluate the effectiveness of these detection methods, several benchmarks are employed, i.e. RealHall[1], HaluEval[4] and TruthfulQA[5]. If looking specifically at the medical domain, very little research has been done in hallucination detection. Med-HALT[6] is the only comprehensive benchmark in this field, and its results are far from promising, leaving much room for improvement.

The primary objective of the project is to develop, refine and benchmark automatic hallucination detection techniques tailored to the medical domain. Ultimately, the goal is to facilitate clinical decision-making, ensure patient safety, promote ethical AI usage and the adoption of AI-based tools in real-world applications.

Methodology

This project benchmarks a variety of hallucination methods and judge LLMs on both general knowledge and medical domain datasets. HaluEval includes a context field for each question-answer pair. It also includes hallucinated answers, which makes creating a detection test quite simple.

TruthfulQA, instead of context, includes multiple samples for both factual and hallucinated responses. In the medical domain, MedQuad[7] compiles information from trusted sources, although it requires some processing such as cleaning the raw, scraped data and generating hallucinated answers. Finally, EHRTest contains QA pairs generated from a series of anonymized electronic health records.

Within the limitations of the available hardware, the following open-source models are selected to act as judge LLM: llama2 (7B), llama2:13b, llama3 (8B), llama3-gradient, nous-hermes2, gemma and mistral.

Incorporating ideas from different SOTA techniques and benchmarks, a series of approaches to hallucination detection were designed. Basic Contextual Verification (BCV) simply asks the judge LLM to contrast a QA pair with some given context. BCV-CoT adds chain-of-thought prompting, where the judge model is asked to explain its reasoning process before outputting a final verdict. Sentence-level Contextual Analysis (SCA) is a more complex CoT methodology where the model is asked to split the given answer into sentences and find a extract from the context that supports or contradicts that sentence. In Iterative Sentence-Sample Comparison (ISSC), the judge LLM is repeatedly fed one sentence from the answer and asked to contrast it with a different sampled response. Lastly, a chain-of-thought variation of the latter method is also implemented, ISSC-CoT.

There are two main phases to this project. In the first part, prompt engineering techniques and small-scale manual tests will be used to iteratively generate and refine prompt templates for the different hallucination detection methods. Then, they will be benchmarked on different datasets and judge models to find the best performing combinations.

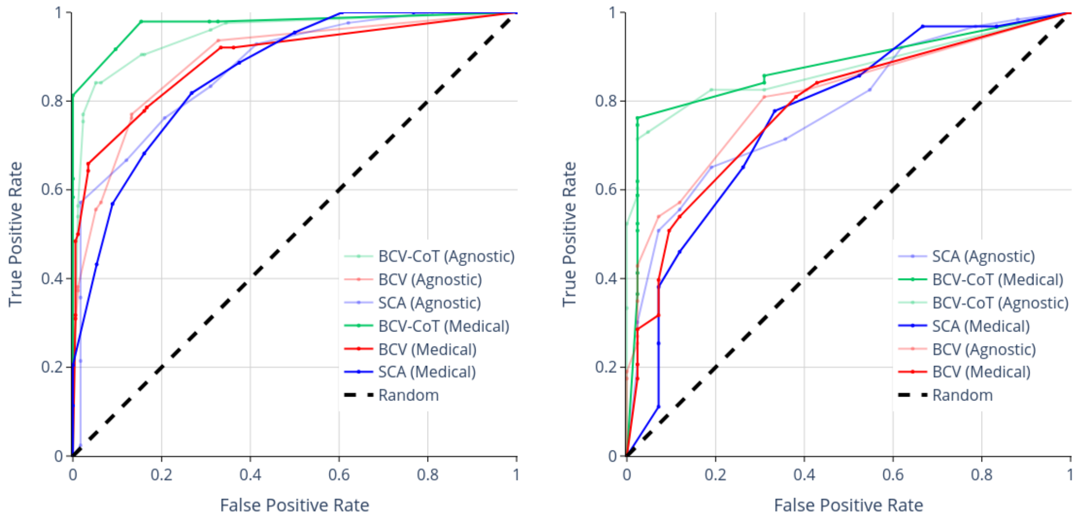
Results

The first phase of the project highlighted some of the many issues that arise when engineering LLM prompts. For instance, the challenge of consistently obtaining a response with an easy-to-parse format. Other problems that had to be solved through careful prompt engineering were blatantly obvious fantasy-themed hallucinations when creating tests, and adding disclaimers whenever a hallucination is generated.

For each test, the most appropiate detection methods were selected according to the dataset structure. For instance, HaluEval was benchmarked on BCV and BCV-CoT. While producing very good results on nous-hermes2 and llama3, these tests showed that some models have a verdict bias, by which they have a clear preference for always judging every case as factual or every case as hallucinated. TruthfulQA, which contains extra samples for each QA pair, seemed ideal to test the ISSC and ISSC-CoT methods. The results for these benchmarks are much less promising than with BCV(-CoT), and given the high number of inferences required to test a single example and its poor performance, ISSC(-CoT) was discarded from further testing.

On the medical datasets, MedQuad and EHRTest, the longer multi-sentence answers allow for SCA as well as BCV(-CoT) to be benchmarked. In both cases, only nous-hermes2 performed reasonably well without chain-of-thought prompting. On BCV-CoT, nous-hermes2, llama3 and mistral achieve relatively high sensitivity and specificity. This method also serves to reduce verdict bias. Finally, SCA does show improved results with respect to raw BCV, but it falls short compared to BCV-CoT. The more complex instructions and format required in the SCA method lead to different errors in the models' reasoning. It also requires noticeably longer prompts and judge LLM responses.

For both medical datasets and the three hallucination detection methods benchmarked on them, a second version of each prompt template was tested where the judge LLM is instructed to take the role of a medical expert. Figure 2 shows the resulting ROC curves when using nous-hermes2 as the judge model. From these plots, BCV-CoT clearly stands out as the superior detection technique, and it seems to benefit from adding such an expert role. However, the effect of this role on the other tests is ambiguous.



(a) EHRTest (b) MedQuad

Figure 2: Effect of expert role on nous-hermes2

Conclusion

This project faced considerable hardware limitations, which made running the latest models with tens of billions of parameters impossible. Even with smaller models, for each one sixteen different benchmarks were carried out, which had to be run sequentially over more than 110 execution hours. With more capable hardware and optimized inference processes, the proposed hallucination detection techniques could be tested on models like llama3:70b or mixtral:8x22b.

During the prompt template development process, many of the challenges involved in prompt engineering had to be overcome. Verdict bias and format inconsistencies carried on to some of the large-scale benchmarks on less capable models.

The results show that the best combination for hallucination detection is a chain-of-thought approach to Basic Contextual Verification, leveraging nous-hermes2 as its judge model. Possible future improvements include testing on larger, more powerful models, as well as tailoring the prompt templates for specific judge LLMs. This could well lead to a comprehensive hallucination filter for an LLM-based medical assistant or other real-life applications.

References

[1] Robert Friel and Atindriyo Sanyal. *Chainpoll: A high efficacy method for LLM hallucination detection*. 2023. arXiv: 2310.18344 [cs.CL].

[2] Hanxing Ding et al. *Retrieve Only When It Needs: Adaptive Retrieval Augmentation for Hallucination Mitigation in Large Language Models*. 2024. arXiv: 2402.10612 [cs.CL].

[3] Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. *SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models*. 2023. arXiv: 2303.08896 [cs.CL].

[4] Junyi Li et al. *HaluEval: A Large-Scale Hallucination Evaluation Benchmark for Large Language Models*. 2023. arXiv: 2305.11747 [cs.CL].

[5] Stephanie Lin, Jacob Hilton, and Owain Evans. *TruthfulQA: Measuring How Models Mimic Human Falsehoods*. 2022. arXiv: 2109.07958 [cs.CL].

[6] Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. *Med-HALT: Medical Domain Hallucination Test for Large Language Models*. 2023. arXiv: 2307.15343 [cs.CL].

[7] Asma Ben Abacha and Dina Demner-Fushman. "A Question-Entailment Approach to Question Answering". In: *BMC Bioinform.* (). URL: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-019-3119-4.

Contents

Chapter 1

Introduction

Since the release of ChatGPT in November 2022, Artificial Intelligence has gained a spotlight in the media and in everyday conversation. While Deep Learning models have been in development for years, it is the rise of Large Language Models that has finally proven how much of an impact these tools can have in the near future. ChatGPT, in particular, quickly became a widely used tool for writing documents, coming up with ideas or even coding.

OpenAI's work on ChatGPT has not stopped. One of the most impressive functionalities that has progressively been added to the tool is its multimodality. ChatGPT is not longer limited to processing text but can now extract information from inputted images, generate new images through its integration with Dall-E, and even engage in real-time voice conversations.

The vertiginous development in this field has sparked significant debate, particularly regarding legal and ethical issues. For instance, AI tools can already generate never-before-seen images, music, movie scripts... which leads to complex issues with copyright, authorship and royalties[1]. Language models are trained on vast datasets sourced from the internet which makes them able to replicate human-like language patterns. However, these models are not without flaws and can present certain biases and generate misinformation with the same level of confidence as their factual information.

World-class tech experts —including Steve Wozniak and Elon Musk— have advocated for a pause in massive model training, to allow legal and ethical regulations to catch up[2]. This does not mean that progress in the field should come to a standstill. Rather, it calls for a more balanced and responsible approach to innovation. AI has vast potential if harnessed for constructive purposes, while taking into account its ethical implications.

One of the main limitations of Large Language Models that can make them impractical for real life use cases, especially in very sensitive domains such as medicine, is their tendency to make up false claims presented with the same confidence as any of their other generated information. These are known as *hallucinations*. When the user is non-expert in the field, their inability to filter out false information can result in very negative consequences.

By developing and implementing effective hallucination detection filters as a buffer layer between the model's output and what the user receives, the risk of generating misinformation can be mitigated. Different approaches have been investigated in order to detect and minimize these hallucinations, but they are generally tested on general, common-knowledge databases.

In this work, a deep dive will be taken into a variety of hallucination detection techniques. After researching the latest, state-of-the-art methods available, they will be adapted and implemented through custom scripts and prompt templates to allow for maximum configurability and testing capabilities. Different benchmarking datasets will be gathered or generated, from common-knowledge tests to more specific medical-domain sets.

With all these different detection techniques and datasets, a wide range of LLM models will be tested on their ability to identify hallucinated information, specifically in the medical domain[1]. This setup allows for much experimentation, by switching between different methods, prompts, models and datasets. The principal aim of this research is to determine if, given the current technology, hallucination detection can reduce the risk of misinformation to a low enough degree to make LLMs viable for real-life applications, especially in very sensitive domains as is healthcare.

[1] Datasets, scripts and results are available at github.com/jdrp/llm-hallucination-detection

Chapter 2

Background knowledge

2.1 Evolution of Natural Language Processing

Natural Language Processing (NLP) encompasses a wide range of tasks revolving around text analysis and generation. Some examples would be speech recognition, sentiment analysis, text completion, question answering or translation.

It all started with Alan Turing's proposal of the"imitation game"[3], which evaluates a machine's ability to exhibit intelligent behavior indistinguishable from that of a human. Early attempts at developing such an algorithm made use of rule-based systems and simple pattern matching. During the 1960s and 70s, these rules where expanded upon to account for more grammatical variations and syntactic structures. However, the amount of rules and patterns to account for when processing text became increasingly difficult to manage.

In the early 1990s, researchers in this field realized that the complexities and sheer vastness of human language could hardly be encompassed as a series of manually defined rules. Completely shifting their perspective, they began to frame NLP as a statistical problem. This change was driven by the availability of increasingly large corpora of text and computational power. N-gram models, which calculate the probability of occurrence of N consecutive words, became the state-of-the-art for tasks such as text completion and spelling correction. However, these models have trouble when processing patterns which are not in their vocabulary.

In 2003, a feed-forward, multi-layer perceptron and a longer context length, trained on over ten million words surpassed the performance of all previous n-gram models[4]. As gathering large amounts of annotated data was very costly and time-consuming, research focused on unsupervised learning from raw, non-annotated text corpora. More advanced Machine Learning (ML) models, e.g. support vector machines (SVMs), were developed and applied to NLP tasks.

In the early 2010s, the rise of deep learning —making use of neural networks with numerous hidden layers— and the use of models such as Recurrent Neural Networks (RNN) or Long Short-Term Memory (LSTM) drastically improved the capabilities when handling sequential data and long-range dependencies between concepts.

2.2 The transformer model

In 2017, a new ML architecture was presented, based on word embeddings and attention mechanisms, that greatly optimized the training and inference of sequence-to-sequence models. It became the state-of-the-art architecture for a variety of tasks, such as NLP, speech recognition and generation, or image processing.

2.2.1 Word embeddings

In 2013, researchers at Google introduced the concept of word embeddings[5]. By representing words as vectors in a multi-dimensional space, semantic similarities and relations could be captured. They used different neural network architectures (feed-forward, recurrent) to train models on a 1.6 billion word dataset.

When analyzing the resulting vectors, the researchers observed that, not only were semantically similar concepts assigned to similar vectors, but more complex relations were also captured. For instance, the difference between the *man* and *woman* vectors would be almost equal to the difference between *king* and *queen*, or *brother* and *sister*. Similarly, the vector from *Norway* to *Oslo* was the same as that from *Greece* to *Athens*. This suggests that concepts such as gender, capitals, size... are captured as directions in the word embedding vectorial space.

The introduction of word embeddings was a major breakthrough. Using vectorial representations instead of raw text as an input made it much easier to capture semantic relations, improving performance across many NLP tasks.

2.2.2 Attention mechanism

Initially introduced in 2016[6], the attention mechanism was improved on in the well-known Transformer paper[7]. It operates as follows:

1. **Input representations**: The attention mechanism operates on a set of input representations, typically embedding vectors.
2. **Query, Key, and Value matrices**: By multiplying the input embeddings with pre-trained weights, QKV matrices are generated.
3. **Attention scores**: To determine the relevance of each input element to a specific query, the attention mechanism computes attention scores. These scores are often calculated using a similarity measure, such as the dot product, between the query and key. The resulting value is then normalized according to the number of dimensions and goes through a softmax function. Then, these probabilities are used to carry out a weighted sum on the values.

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{2.1}$$

The transformer model applies two variations of the attention mechanism: self-attention —the QKV matrices are all derived from the same input sequence— and multi-headed attention —calculations are divided into various parallel heads.

2.2.3 Encoder-decoder architecture

The Transformer model is built upon an encoder-decoder architecture, which facilitates efficient and effective handling of sequence-to-sequence tasks. This architecture consists of two main blocks: the encoder and the decoder, each composed of multiple layers of self-attention and feed-forward neural networks.

The encoder block repeatedly passes the input embeddings through multiple attention mechanisms and feed-forward networks, resulting in new output embeddings that will be used as the input to the decoder.

In the decoder, the embeddings are repeatedly processed against themselves and against the encoder output, to end up generating a series of logits. These logits are then converted to probabilities for each token in the available dictionary.

The input embeddings in the transformer model have some peculiarities. Firstly, text isn't usually divided into words, but rather into certain"tokens" or parts of words. Secondly, a positional encoding is added to the initial embedding in order to capture the order of the tokens.

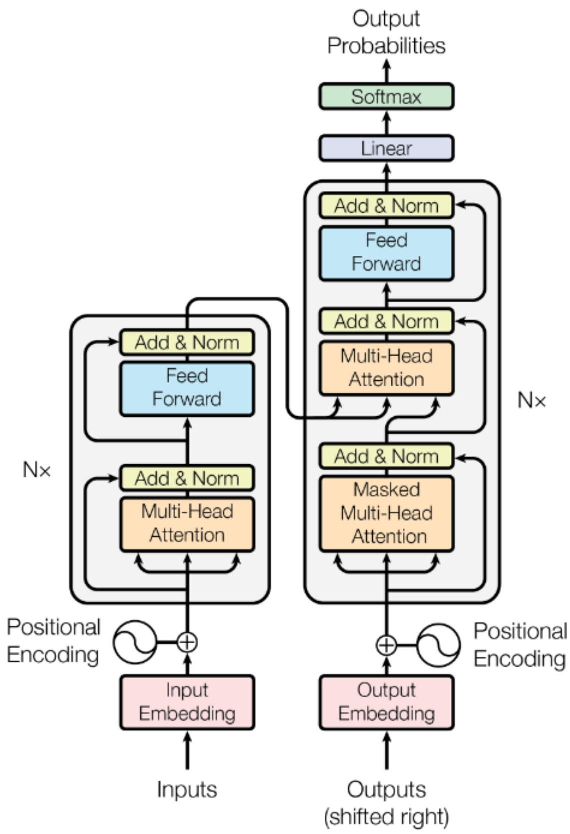


Figure 2.1: The encoder-decoder architecture of transformer models

The attention mechanism in the decoder is slightly different as the original method, as it incorporates a mask to ensure that the prediction for each position only depends on the previous positions.

The encoder and decoder blocks can be used separately or in conjunction, depending on the task at hand. Multiple instances of these blocks are usually stacked on top of each other, providing the model with more opportunities to attend to and relate tokens.

2.2.4 Impact

The transformer architecture is designed with modern computation in mind. Nowadays, GPUs can carry out enormous amounts of calculations in parallel, which works great with the highly parallelizable operations involved in matrix multiplication —the core of the attention mechanism. This design is much more efficient than simply using a very large neural network, as in that case most calculations would have to be carried out sequentially.

Models based on the transformer architecture have proven to excel at identifying related terms, even when they are far from each other in the text. Nowadays, most language models are based on different variations of the transformer, and its parallel nature has allowed for the training of models with tens of billions of parameters.

2.3 Large Language Models

2.3.1 Key architectures

In mid-2018, OpenAI released GPT 1, a decoder-only model trained on the next token prediction task. When inputted a string of text, GPT —shorthand for Generative Pre-trained Transformer— would predict the most likely way to continue the sentence. Using simplified decoder blocks without the second multihead attention layer –as there is no encoder output to attend to–, this 117 million parameter model managed to improve on state-of-the-art results for a variety of NLP tasks, even without being tuned specifically for those tasks[8].

Later that same year, Google released their BERT model, which stands for Bidirectional Encoder Representations from Transformers. They leveraged the capabilities of their new transformer architecture to train an encoder-only model with a 110 million and a 340 million parameter variants. By bidirectionally attending both preceding and following words, the model can fulfill two main tasks: masked token completion, where it predicts the most likely word to fill in a gap in a sentence, and next sentence prediction, where it detects whether two sentences could appear sequentially in a text[9]. Other models can be trained to carry out different tasks on the output vectors of BERT, such as sentiment classification[10].

The architectures of BERT and GPT, as can be seen in figure 2.2, look very similar, the only differences being the masked attention and the number of repeating blocks. However, they are based on the encoder and the decoder respectively and their training, inputs and usage focus on tasks associated to those components. That is why BERT works best for downstream tasks such as entity recognition and classification and completion, and GPT works best as an autoregressive model for text generation[11].

The release of GPT and BERT triggered a new trend in the field, as the

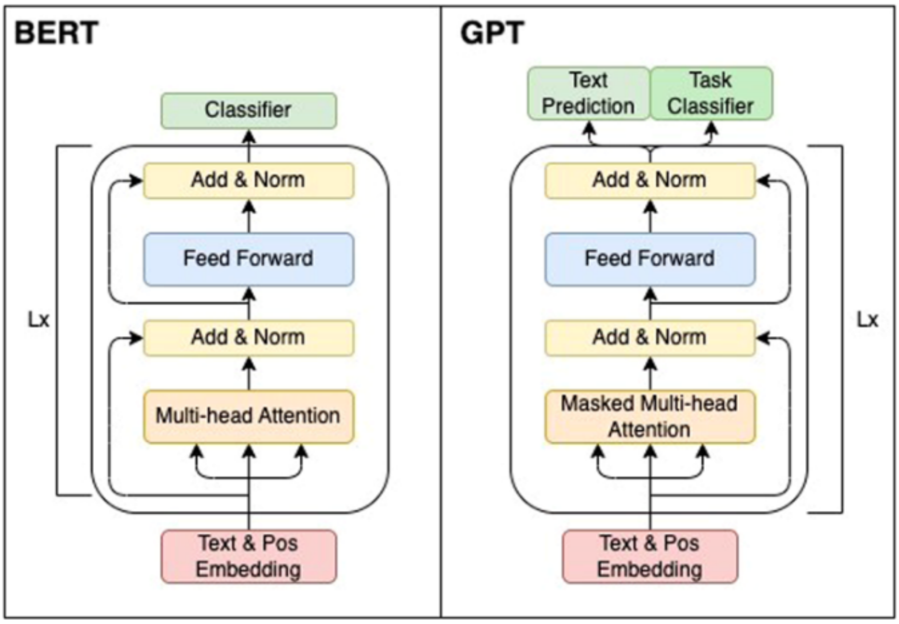


Figure 2.2: Differences in architecture in BERT and GPT

focus shifted to training increasingly large models, eventually reaching tens of billions of parameters. This was made possible by the parallel nature of the transformer architecture, which made training massive models viable as long as enough GPU capacity was available.

Throughout the next few years, a variety of these Large Language Models were released, rapidly increasing in number of parameters, size of training datasets and context length, i.e. the maximum number of tokens that can be processed. GPT 1 was followed by GPT 2 (2019), 3 (2020), 3.5 (2022) and 4 (2023). As the capabilities and precision of these models increased, OpenAI became more secretive with regards of architectural details and weights, and since late 2022 provides a “black-box” service through their ChatGPT interface. However, some details on GPT 4 have been leaked claiming that it contains 1.76 trillion parameters divided into 8 models in a Mixture of Experts fashion. MoE refers to an architecture design where different sub-models are trained to excel at certain tasks or domains, and a governing neural network decides which of these models to activate for a given input.

2.3.2 Open models

While OpenAI has been at the top of most LLM benchmarks with their GPT models, the fact that they are only accessible through API —with usage caps even for paying users— and their parameter weights are not disclosed makes these models impractical for any large-scale project.

Other companies have focused on releasing models with openly available weights, which, although usually lagging behind GPT in performance benchmarks, are a much better option for projects that require fine-tuning the model for specific tasks (as explained in 2.5.1) or automating a large number of prompts.

Meta's Llama 3 in its 8B parameter version outperforms GPT 3.5 on different benchmarks[12], while being much faster and requiring relatively low GPU capacity. The 70B version can even surpass GPT 4 in some tasks, such as Python coding or grade school math[13]. Even the previous version of Llama, Llama 2 70B, performs better than GPT 3.5 in plenty of tasks.

Another model that falls into the top spots of most benchmarks is Mistral AI's Mixtral 8x7B, which applies a MoE architecture like the one in GPT 4. This approach not only enhances performance across a wide range of tasks but also optimizes efficiency, making it a competitive choice in terms of both speed and computational resource requirements.

2.4 LLM hallucinations

The ability of LLMs to generate human-like text comes at the risk of what is commonly referred to as hallucinations. At times, language models can generate content that, while sounding plausible, is factually incorrect or does not align with the user's instructions or a given context. A common way to classify these hallucinations is to distinguish between lack of factuality and lack of faithfulness[14].

2.4.1 Types of hallucinations

Factuality

This type of hallucination occurs when a language model produces a fact that contradicts or cannot be verified using real world knowledge. There is further distinction between factual inconsistency and factual fabrication.

Factual inconsistencies refer to hallucinations where the generated content contradicts the available information. A common example of this kind of hallucination would be the following:

Chat 2.1: Factual inconsistency

User: Who was the first person to land on the Moon?

Model: Yuri Gagarin in 1956.

Explanation: *The answer according to common knowledge is Neil Armstrong in 1969. Yuri Gagarin was the first man to travel through outer space, but he never landed on the Moon.*

Factual fabrication, on the other hand, happens when a model has no knowledge of the topic in question and tries to provide plausible-sounding responses without any basis in reality.

Chat 2.2: Factual fabrication

User: Who invented the first teleportation device?

Model: The first teleportation device was invented by Dr. Alan Turner in 2025, allowing instant travel between two points.

Explanation: *Teleportation devices do not exist, and the model fabricated the name"Dr. Alan Turner" and the invention in 2025.*

Faithfulness

In faithfulness hallucinations, the model's response might not necessarily contradict real world knowledge. However, it diverges from the user's instructions or the context provided. Here is an example where the model does not stick to what the user is instructing:

Chat 2.3: Instruction inconsistency

User: Translate the following sentence to Spanish: What is the capital of France?

Model: The capital of France is Paris.

Explanation*: The model did not translate the sentence as the user asked. The answer should be "¿Cuál es la capital de Francia?"*

In other cases, specially in summarization tasks, the user inputs some context. The model might ignore or contradict that context in its response, leading to hallucination.

Chat 2.4: Context inconsistency

User: The Eiffel Tower is located in Paris, France. How tall is it?

Model: The Eiffel Tower in New York City is 300 meters tall.

Explanation: *The model ignores the location of the Eiffel Tower inputted by the user.*

Finally, language models also fall into contradictions. These most frequently occur in mathematical reasoning, especially when using a non-finetuned model, but contradictions may also appear between two sentences in the same generated answer.

Chat 2.5: Logical inconsistency

User: Tell me about the tallest mountain in the world.

Model: Mount Everest is the tallest mountain in the world. However, K2 is actually taller than Mount Everest.

Explanation: *The model contradicts itself by stating both that Mount Everest is the tallest and that K2 is taller.*

The previous examples of user-model interactions are designed to clearly highlight each specific type of hallucination. However, real language models typically hallucinate in much more subtle ways. The varied and inconsistent nature of language and information makes it very challenging to systematically detect misinformation in LLM-generated text, and unaware users tend to accept the model's response as factual as long as it sounds "right".

2.4.2 Causes of hallucination

The causes behind LLM can be divided into three main stages: the data used in training the model, the training process itself, and inference.

Data

Training datasets are usually built by scraping Internet resources. When handling vast amounts of data, manual filtering becomes unmanageable and misinformation might be passed to the model during training. When asked about these topics, the model will remember its training data and output false information.

Any subset of data, especially from the Internet, will contain some inherent biases. These biases will propagate to the model's responses, possibly causing hallucination.

Training

A common cause of hallucination is the different approach between the training and inference procedures. In training a model, ground truth tokens are used as an input. Meanwhile, during inference the model's output is recursively fed into itself. This inconsistency can cause hallucinations, as a wrongly generated token is likely to propagate the error to the entire subsequent sequence. This mismatch between training inference is commonly known as exposure bias[15].

There is another phase in the training process where supervised fine-tuning and reinforcement learning from human feedback enable the model to follow user instructions and conversations. In this phase, there is a risk of capability misalignment —where the model is demanded more than it can do— and belief misalignment —where the model will tend to satisfy the human evaluator over producing correct information.

Inference

LLMs implement a certain degree of randomness when sampling the output token out of a probability distribution. This usually leads to higher-quality answers than just picking the most likely token. However, inferencing with too much temperature —randomness— can lead to unlikely, imprecise tokens being chosen.

Another common problem during inference is the limited context length of the models. Sometimes, a piece of data required to answer a question might be too far back in the text, and the model is unable to apply the attention mechanism on that information.

2.5 Advanced LLM techniques

A variety of techniques have been used throughout the development of LLMs with the aim to improve performance and accuracy. Fine-tuning, chain of thought and retrieval-augmented generation focus on generating higher quality answers and reducing hallucinations. Quantization, on the other hand, optimizes memory and computing requirements.

2.5.1 Fine-tuning

A common approach when leveraging LLMs for specific purposes is to fine-tune an existing model on a custom dataset tailored to whatever domain is being worked on. In this way, the conversational and reasoning capabilities of the original model —which usually require massive amounts of training data and computing resources— are maintained, while adding expertise and knowledge of certain fields. Typically, the internal weights of the model are frozen and only the last neural network layers are tuned. This greatly reduces the time of training and ensures that the original performance does not deteriorate.

Fine-tuning language models has been shown to significantly improve their accuracy in technical domains. Several studies support this claim by demonstrating how domain-specific fine-tuning enhances the performance of LLMs across various specialized tasks[16].

2.5.2 Chain of thought

When applying chain of thought (CoT), a language model will first generate a step-by-step explanation of its reasoning process in order to reach a final answer to the user's enquiry. Instructing the model to respond in this way is particularly useful for tasks requiring logical reasoning, complex problem-solving, or multi-step calculations.

In elaborate CoT-based responses, the model will subdivide the user's prompt into a number of subproblems, and sequentially go through the different steps involved in solving those individual problems. This leads to a number of advantages:

- The output is easier to interpret, as the user can read and follow the model's thought process.
- Complex, multi-step tasks can be handled better.
- By explicitly validating each step, the model can catch errors before they propagate, improving its accuracy.

2.5.3 Retrieval-augmented generation

In some situations, a model might not have the required knowledge to meet the user's requirements. While fine-tuning is an option, the training process required for it might be too costly in terms of resources or time.

In these instances, retrieval-augmented generation (RAG) can be a great alternative.

RAG gives a model access to domain-specific databases or online sources, usually through an intermediate step which adds this retrieved knowledge to the user's prompt before the actual inference. In this way, the model has the necessary context to properly answer the user's prompt, without needing to be specifically trained or fine-tuned for that purpose.

2.5.4 Quantization

Quantization is a technique used to reduce the computational and memory requirements of ML models by representing their parameters with lower precision. LLMs are typically trained using 32-bit floating-point numbers for their parameters. This high precision allows for accurate calculations when applying gradient descent, leading to better performance and more effective learning.

However, and thanks to the probabilistic nature of LLM token prediction, parameter precision can be reduced to decrease the model size and speed up inference, with barely any reduction in accuracy and response quality. Different quantization schemes can be applied, storing parameters using 16-bit, 8-bit and even 4-bit precision.

Even though quantizing a model can take a —very small— hit on answer quality, the considerable reduction in parameter size means much larger models can run on a machine with limited capacity.

Chapter 3

State of the art

3.1 Domain-agnostic hallucination detection

One of the biggest problems that prevent Large Language Models from being deployed in commercial settings is their tendency to hallucinate, as explained in section 2.4. Businesses cannot risk a model giving their customers false details about their products or services, or disclosing sensitive information. For this reason, hallucination detection for LLMs has been very active area of research in recent months, with widely different approaches being developed and investigated.

There is no infallible method for detecting hallucinations. Each approach has its own advantages and disadvantages, and tends to perform better in certain specific circumstances. In this section, some of these detection techniques and benchmarks will be discussed.

3.1.1 Detection techniques

ChainPoll

ChainPoll, developed by Galileo Labs[17], builds on previous attempts at hallucination detection. It uses chain-of-thought prompting to ask GPT-3.5 whether a certain statement contains hallucinations. It functions both for closed-domain situations —where some context is provided by the user— and open-domain situations —where no context is given, and the statement is contrasted with the world knowledge.

Using CoT not only improves the resulting evaluation by the model, but it can also help track down where the hallucination actually is, or why the model —at times— can fail to detect it.

When prompting the model to detect hallucinations, ChainPoll actually repeats the process multiple times and averages the result. This helps smooth out any generation errors, at the cost of extra computation.

Rowen

Rowen makes use of selective retrieval-augmented generation[18]. Given a certain question and answer, it will judge their semantic consistency across different languages and other sampled answers. If this confidence falls below a certain threshold, Rowen will retrieve the necessary knowledge using the Google Search API and regenerate a corrected answer based on that knowledge.

Although this process does take some extra steps, it results in an average generation time lower than other RAG approaches and better accuracy.

MIND

Contrary to other methods which treat the answering LLM as a black box, this approach leverages unsupervised Modeling of INternal states for hallucination Detection[19]. By training an MLP network to detect hallucinations based on the internal embedding values at certain layers of the model.

The main advantage of this approach is avoiding the need for extra inference steps. Passing the internal embeddings through the network can be carried out much faster than prompting external models, and only in the case of a high hallucination probability will mitigation strategies, i.e. information retrieval, be applied.

SelfCheckGPT

SelfCheckGPT[20] is a black-box approach to hallucination detection in question-answering tasks. Given a question and a generated answer, other possible answers are sampled with GPT-3. The consistency of the suggested answer is computed against each sample and averaged, resulting in a hallucination score. Different methods can be used to calculate consistency, typically processing each separate sentence from the answer independently.

3.1.2 Benchmarks

A few benchmarks have been developed to evaluate the performance of different models when detecting hallucinations.

RealHall

RealHall[17] sorts through and compiles various datasets with examples on LLM hallucination. The authors looked for datasets containing realistic hallucinations, as well as challenging ones. This benchmark, used for Galileo Labs' LLM Hallucination Index[21], is not publicly available as it is mostly a compilation of previously existing datasets.

HaluEval

HaluEval[22] encompasses a variety of NLP tasks: dialogue continuation, question answering from a given knowledge, and text summarization. For each sample, both a correct and a hallucinated answer are generated. From this dataset, a hallucination detection test can be carried out to measure the capabilities of different models.

TruthfulQA

A particular trait of the TruthfulQA[23] benchmark is that it includes not only a true and a hallucinated answer, but it also adds a list of extra sampled responses, both correct and hallucinated.

3.2 Hallucinations in the medical domain

Little research has been done in developing detection techniques and benchmarks specific for medical-domain hallucinations. Only one such benchmark has been found that compares the performance of a variety of models according to different medical-related tasks.

3.2.1 Benchmarks

Med-HALT

Med-HALT[24] defines a variety of tasks, of which three are relevant —the others involve generating URL links, which is unrelated to hallucinations.

- A False Confidence Test (FCT) which randomly selects a test-type answer and tasks a model to justify whether the selection is right or wrong.
- A None of the Above Test (Nota), in which a model has to identify the wrong options in a test-type question by selecting the "None of

the above" option.

- A Fake Questions Test (FQT), where a nonsensical question is presented to check if the model can identify it as such.

The results from this benchmark leave much room for improvement. While some models, i.e. Falcon 40B and Llama-2 70B, perform great at the FQT task, the available examples show that a large number of these fake questions refer to fantasy themes[1], which are easily identifiable as absurd or incorrect. FCT and Nota seem like much more relevant taks. Only Llama 2 achieves an accuracy over 70% in the Nota test, and the results in FCT are very poor for all models (peaking at around 40% accuracy).

[1] For instance, an example question asks *"In the realm of improbability, where mermaids reign supreme with their uncanny possession of esophagi, by what unfathomable and inexplicable histological finding could the incomprehensible diagnosis of Barrets esophagus be ascertained in a Mermaid biopsy?"*.

Chapter 4

Motivation

4.1 Addressing the limitations of LLMs

Creating a language model that can generate truthful information with complete reliability is a theoretical goal rather than a practical one. The complexity of what is true in different contexts; the finite amount and quality of training data; the inherent ambiguity and uncertainties in language, and the difficulty of evaluating the performance of these models are some factors that lead to that conclusion, according to wide scientific research[25, 26]. This means that relying on LLMs always comprises a risk, especially in very sensitive domains such as healthcare. Even fine-tuned models have a hallucination rate much higher than what would be reasonably acceptable.

Much effort has been made in developing strategies to mitigate hallucination when training and inferring with LLMs. However, the complexity of language and AI-generated text makes it hard to evaluate these strategies. A different line of investigation is hallucination detection, which aims to detect misinformation in generated responses. This approach can be useful in two ways:

- Automated hallucination detection can optimize and improve the development and evaluation of new and refined mitigation strategies.
- Hallucination detection techniques can also be implemented as a buffer layer between the LLM and what is outputted to the user. If a hallucination is detected, the response can be corrected or regenerated, greatly reducing the risk of producing misinformation even with less capable models.

4.2 AI in healthcare

Deep learning has been used in the medical world for over a decade. A common application of these techniques is cancer detection in radiology images[27, 28]. Other uses of AI in medicine include physical robots that can amplify a surgeon's capabilities —vision, precision...— during surgical processes.

In a medical context, even the slightest error can lead to severe consequences. These errors typically arise from the limitations of human knowledge. Physicians face the challenge of having to remember thousands of diseases and treatments. This task is further complicated as they must stay up-to-date with the rapid advancements in research. AI could potentially alleviate this challenge by providing comprehensive, updated information, which enables doctors to make more informed decisions.

AI systems can analyze vast amounts of medical data quickly and accurately, identifying key elements and patterns that might be missed by a human reader. For instance, language models can play a crucial role in enhancing patient care by reading through a patient's medical history to assist the doctor in diagnosing and answering specific questions. These models can extract relevant information from health records, such as previous diagnoses or medications, family history... By providing a concise summary of this data, they enable doctors to have a comprehensive understanding of the patient's medical background.

Investigating AI applications in healthcare will not only enhance medical practice but also promote the widespread integration of human-AI collaboration. By unifying the strengths of both human expertise and AI capabilities, more efficient and effective workflows can be designed across various sectors. This synergy has the potential to revolutionize industries, improving productivity and innovation while maintaining high standards of quality and safety.

Chapter 5

Project objectives

This research project aims to develop and refine automatic hallucination detection techniques for LLM-generated text, with a focus on the medical domain. In the process, several intermediate tasks will be carried out, each contributing to the overall goal.

1. **To investigate the current state of the art** in hallucination detection for LLM-generated text. There is currently a growing interest in this field as it poses the greatest obstacle to real-life, commercial implementation of language models.

2. **To develop custom implementations** of SOTA hallucination detection techniques. This will allow for vast experimentation and tweaking in order to optimize their performance.

3. **To select a variety of open-source LLMs** which will act as judge on the different benchmarks. Within the limits of the available hardware,

the aim is to compare performance across different model sizes, architectures and other variables.

4. **To collect a series of question-answering datasets**, both domain-agnostic and healthcare-related. Some of these datasets will be extracted from previously existing benchmarks, while others will be generated from own source data.

5. **To refine the prompt templates** used in the different hallucination detection methods. An ideal prompt template should produce accurate, easy-to-parse responses on any given model.

6. **To test and benchmark** the different judge LLMs and detection methods on the domain-agnostic and healthcare-specific datasets. The strengths and weaknesses of each combination will be contrasted in order to find the optimal process for hallucination detection.

Through all of these steps, this project aims to:

- **Facilitate clinical decision making** by working towards the development of LLM-based tools for medical diagnosis. These tools aim to assist medical professionals in accurately and efficiently interpreting complex, extensive medical information.

- **Enhance patient safety** by developing robust, automatic hallucination detection in LLM-generated medical text, ensuring reliable and accurate information.

- **Promote ethical use of AI** by mitigating the risk of misinformation and misuse. By developing effective hallucination detection techniques, this project works towards ensuring that AI-generated content meets rigorous accuracy and reliability standard, especially in sensitive domains like healthcare.

- **Enable the adoption of AI-based tools** for real life use cases, both in healthcare and other areas. There is great potential for AI-based tools to streamline and optimize many real-life tasks, if implemented in a reliable manner. In the case of LLMs, the main barrier in their implementation is precisely their tendency to hallucinate, which this project aims to minimize.

Chapter 6

Resources and tools

6.1 Benchmarking datasets

6.1.1 General knowledge

HaluEval

The question-answering subset of HaluEval[22] includes a knowledge field for each question-answer pair. It also includes hallucinated answers, which can be used to create a detection benchmark.

The questions in HaluEval cover a broad range of topics, from historical events to music groups and actors.

TruthfulQA

TruthfulQA is a hallucination detection benchmark which includes a wide variety of hallucination types, categories and topics. In this dataset, each question is accompanied by not only a right answer its hallucinated counterpart, but also by some extra samples for both[23].

6.1.2 Medical domain

MedQuad

MedQuad[29] compiles medical information from a variety of trusted official sources, e.g.: the National Institutes of Health[1]. For each disease, treatment or drug, a series of questions and answers are scraped from the online resources.

EHRTest

A custom dataset generated from electronic health records (EHRs). The original EHR corpus, released by the Text Retrieval Conference (TREC)[2], includes 101,712 de-identified documents or health records, comprising 17,265 visits or medical episodes of various patients (each visit can have between 1 and 415 documents or reports).

Using a language model, a series of questions and answers (both correct and hallucinated) are generated from each EHR. This process will be explained in detail in section 7.1.

[1] www.nih.gov
[2] trec.nist.gov

6.2 Tools

6.2.1 Hardware limitations

The original idea for this project was to fine-tune a medical expert model in order to test is viability as a physician assistant chatbot. However, there are some constraints in terms of computing power that ultimately led to a change in the project objectives. LLM models run much faster when on a GPU, thanks to parallelization of operations. However, the entire model needs to feed into the graphics card VRAM.

The GPU used for this project has a dedicated capacity of 12GB. This considerably limits the size of the models that can be tested. Even with very aggressive q4_0 quantization used by Ollama, the maximum number of parameters is under 20 billion.

6.2.2 Scripting and libraries

All of the programming required for this project will be carried out in Python. The Ollama library[3] provides an intuitive, high-level way to interact with a wide variety of open-source LLMs. Using this library, testing out different models is as simple as passing a different model name.

Another advantage of Ollama is that most models are available in a few

[3] ollama.com

different sizes, usually offering a smaller option that can fit into limited GPU memory. For instance, llama3 comes in both a 70B and an 8B version. This reduction in size makes inference faster and less demanding at the cost of less accurate responses.

The Natural Language Toolkit (NLTK) library includes functions to intelligently divide text into sentences. This will be useful when contrasting answers in a sentence-by-sentence manner.

6.2.3 Choice of LLMs

Taking into account the hardware limitations explained in section 6.2.1, the following models have been selected for this project:

Model	**Identifier**	**Parameters**	**Why it was chosen**
Llama 3	llama3	8B	Best-known SOTA open model by Meta
	llama3-gradient	8B	Increased context length
Llama 2	llama2	7B	Compare performance across releases
	llama2:13b	13B	Compare performance across different sizes
Hermes 2	nous-hermes2	10.7B	Nous Research's scientific fine-tuning of Solar
Gemma	gemma	7B	Open, lightweight alternative to Gemini by Google
Mistral	mistral	7B	Powerful competitor to Llama

Table 6.1: Ollama LLMs chosen for this project

During the model selection, models that are too large for the available VRAM were discarded. This is the case with Llama 3 70B, the top performing model in most available benchmarks and comparisons. Solar was also rejected as its Ollama implementation is extremely slow for inference.

Chapter 7

Methodology

7.1 Preparing the benchmark datasets

7.1.1 Pre-processing

In HaluEval, the original data already has the proper JSON format needed to create hallucination detection tests —knowledge, question, right answer, hallucinated answer.

TruthfulQA is originally in CSV format. It also contains the necessary fields to create a test with extra samples for each correct or hallucinated answer.

MedQuad requires some pre-processing steps. Firstly, in order to gather up-to-date context for the questions and answers, a random subset of the XML files is selected and the contents of their source websites are scraped. Scraping each domain requires looking for the appropriate

HTML elements, filtering out irrelevant information and unwrapping the text contents.

As the dataset is five years old, some of the sources are missing or completely restructured. However, given that the dataset contains close to 17k files, each with multiple questions, the working sources are more than enough to generate a large test with a wide variety of medical concepts.

The answers given in the MedQuad dataset are just the raw contents scraped from online sources back in 2019. They contain a lot of redundant spacing and line breaks, outdated references, and sometimes don't actually answer the corresponding question. Because of this, an extra step is taken where question is passed to a language model in order to regenerate the correct answer, using the original scraped answer as context. The prompt template used for this inference emphasizes the need for correctness and instructs the model to only include information found in the context. This prompt template, as well as the numerous others that will be used throughout the project, can be found in appendix B.1.1. After testing different models for this task, nous-hermes2 produced the best responses in terms of both quality and efficiency[1].

Finally, the EHR benchmark requires the most pre-processing. The TREC dataset contains over 100k medical records which have already been

[1] For instance, when trying out both llama3 and llama3-gradient, they tend to just copy part of the previous response while repeating the same formatting errors. Hermes properly adheres to what is asked of it in the prompt. It also infers considerably faster than other models.

anonymized to prevent privacy concerns. The XML fields that contain relevant information —chief complaint, admit and discharge diagnosis, main text— are extracted. Within the main text, the doctor's signature and anonymization notice also have to be removed. All diagnosis fields are coded following the ICD-9 standard. These codes are translated to their respective meanings to prevent the LLMs from misinterpreting them.

Finally, an LLM is prompted to generate a series of question-answer pairs from each report (processed into a JSON format). The prompt template used is included in appendix B.1.2. In this task, llama3 was chosen as it produced the most consistent response format.

Following all of these steps, a dataset of 300 question-answer pairs are generated from a randomly selected subset of EHR reports.

7.1.2 Generating hallucinations

Once a dataset contains the necessary fields of context/samples, question and correct answer, a hallucinated counterpart to the answer needs to be generated. For this purpose, a model is instructed to either negate some of the information or substitute medical some terms with unrelated ones.

These restrictions were implemented as initial testing proved that giving the model too much freedom led to nonsensical, fantasy-themed answers, trivial to identify as hallucinated.

7.1.3 Sampling tests

The final step in the benchmark creation process involves randomly sampling correct or hallucinated answers for each dataset. In the case of TruthfulQA, the respective correct or hallucinated samples are also included. A 'hallucination' label is also added —where 1 means hallucinated and 0 means correct— to provide a ground truth evaluation.

While in real life usage LLMs tend to hallucinate on a minority of occasions, a hallucination rate of 50% is used in this sampling process to prevent evaluation biases.

7.2 Implementing hallucination detection

7.2.1 Customized detection methods

After researching the latest papers on the topic of hallucination detection, a series of tests were designed by putting together ideas from different SOTA methods. In each method, an LLM acts as judge to determine whether the answer to a question is factual or hallucinated.

- **Basic Contextual Verification (BCV).** A simple test where a question and answer pair is contrasted with some provided context. The judge LLM only outputs 'Yes' if the answer contains hallucinations and 'No' otherwise. The prompt used for this test (appendix B.2.1) is a modified version of one used in HaluEval ([22]).

- **CoT Verification (BCV-CoT).** A chain-of-thought variation of the previous method. The judge LLM is instructed to explain its step-by-step reasoning, and output a final verdict based on that process.

- **Sentence-level Contextual Analysis (SCA).** With this method, the judge model is first asked to split the proposed answer into sentences, to then contrast each separate sentence with the context. For each sentence, the model must look for a relevant passage in the context, and judge whether it supports or contradicts the claims. If any sentence contradicts the context or cannot be inferred from it, the final verdict will be of hallucination. This test involves a more elaborate prompt that details a format for the LLM response including an individual reasoning process and verdict for each sentence, as well as a final check for whether the answer is relevant to the question.

- **Iterative Sentence-Sample Comparison (ISSC).** Inspired by SelfCheckGPT, the judge LLM will contrast each sentence in the answer with each of the samples[2]. The average score is then computed as a final probability of hallucination.

- **CoT Iterative Comparison (ISSC-CoT)** A chain-of-thought variation of the previous method.

[2] To avoid confusion, the prompt only includes one sentence and one sample. The inference is then iterated for every combination.

These detection methods are implemented using Python scripts that take in a prompt template, a judge LLM and a test dataset. For BCV(-CoT) and SCA, each prompt is passed to the model five times, and the resulting predictions are averaged. This is not applied to ISSC(-CoT) as it already takes multiple inferences for each test case.

7.2.2 Prompt engineering

Each detection method requires a finely engineered prompt template in order to get the judge LLM to act as desired. Especially with longer, more elaborate CoT responses, a consistent format is key to be able to parse the model's verdicts.

Designing these prompts involves a long, iterative process of testing with different models and data samples, reading through the responses to find possible errors or inconsistencies, and tweaking the prompt to prevent the model from repeating those same mistakes.

Section 8.1.1 highlights some of the most interesting responses generated while tuning the different prompts required for this project.

7.2.3 Parsing LLM responses

Even after considerable prompt engineering, the LLM's output can sometimes contain format inconsistencies. For example, it might add some extra explanations after the final verdict, or include a disclaimer when generating hallucinated text. The Python scripts in charge of parsing this output have to account for such edge cases, leveraging tools like pattern matching with regular expressions.

7.3 Testing

7.3.1 Small-scale tests

Given the time taken by the LLM models to infer their responses on the limited hardware available, and the high probability of formatting inconsistencies when the prompts are not properly engineered, running large-scale tests from the start is both very time-consuming and ineffective.

In order to try out different prompts, response formats, parsing techniques, and consistency across different models, a series of small-scale tests will be carried out where the LLM outputs will be examined manually to find areas of improvement. This is the most time-consuming part of the project, as prompt engineering requires a long, iterative process of repeatedly tweaking each prompt to fix inconsistencies in the models' responses. This process has to be repeated for each of the prompt templates needed for this project: both the different steps in dataset creation and the different hallucination detection methods.

7.3.2 Large-scale tests

Once satisfied with the performance of the different prompt templates and the ability to parse detection verdicts, large-scale tests will be automated to try out various combinations of judge models and detection methods on each benchmark dataset. For each dataset, the most appropriate method or methods of detection will be selected and benchmarked with each judge LLM (see table 6.1).

The Python scripts written for these benchmarks allow for a custom dataset and prompt template to be entered as arguments, making it easy to program the execution of many consecutive tests through batch files.

The constantly high usage of GPU processing power requires the laptop to always be plugged in to a power source, as the battery drains very quickly when executing these tests. For this reason, most of the benchmarks are left running overnight.

At times, some models become extremely slow for inference, and generating a single response can take several minutes. This can completely stop progress for hours, especially when running overnight and without supervision. To prevent this, the script includes a time limit for model responses, breaking out of a test if it takes too long. On the first benchmark iteration, the timeout value is set to 120 seconds. After all the properly-working tests run, the timeout will be increased to 900 seconds to try to produce results for the remaining slowest tests.

The results of these benchmarks will be discussed in section 8.2.

7.3.3 Analysis of results

The hallucination detection tests produce JSON files where each example includes a predicted hallucination score between 0 (factual) and 1 (hallucinated), as well as the ground truth. This is effectively a binary classification problem, where a common approach is to analyze the receiver operating characteristic (ROC) curve. This plot shows the true and false positive rates for every classification threshold. In general, the more area encompassed under the ROC curve (AUROC), the better the performance of the classifier.

In this work, an optimal classification threshold is also estimated for each test and model, by minimizing Youden's J statistic. This approach is explained in more detail in appendix C.

Chapter 8

Experimentation and results

8.1 Prompt engineering

While LLM models excel at many text generation tasks, it can be challenging to get consistent results, especially in the way their responses are formatted. When benchmarking different judge LLMs and hallucination detection methods, a very large number of inferences will be carried out, making it unrealistic to read through every response manually. For this reason, designing and tweaking prompt templates that will a) improve accuracy and b) produce responses easy to parse automatically, is a key step in this project.

Prompt engineering involves a repetitive process of testing prompts on small batches of data, manually reading through the responses, finding inconsistencies and unnecessary information, and tweaking the prompt template to prevent these problems.

A common approach in prompt engineering is to include a list of restrictions such as *"You will ONLY include information relevant to the question"* or *"Answer Yes or No. Do NOT add extra explanations"*. Another idea is to include labels such as *#Context#* or *#Instructions#* to help the model identify key information. A lot of examples of these techniques can be found in the prompt templates used for this project (appendix B).

When prompting a model about a certain technical topic, it can help to establish its role as an expert in the relevant field. For instance, an introductory sentence can be added to hallucination detection prompts, stating *"You are a medical expert acting as an answer judge"*. When benchmarking the different methods and models, the effect of adding this statement will be studied.

8.1.1 Findings and insights

During the process of preparing and refining the prompts utilized in this project, a lot of odd or interesting LLM responses came up. In this section, a few of these interactions will be discussed. The prompts shown here are simplified for clarity purposes, as they tend to be quite long.

Fantasy-themed hallucination

When asking different models to produce a hallucinated response to a given question, they would often resort to introducing fictional creatures and themes into their answer.

Chat 8.1: Example of a fantasy-themed hallucination

User: Who is at risk for Adult Soft Tissue Sarcoma?

Model: People who live near ley lines and experience frequent poltergeist activity are at risk for adult soft tissue sarcoma. Other hallucinatory risk factors include being bitten by a werewolf, ingesting cursed food prepared by a witch, or receiving an ancient curse passed down through generations.

These responses are easily identifiable as non-factual, and wouldn't provide a good measure of judge LLMs. To fix this, the prompt template was modified to add certain restrictions to the model's hallucination. It is instructed to only use medical terminology, to avoid fantasy themes and to generate the hallucination by slightly modifying the original, correct answer.

The exact wording of this prompt can be found in appendix B.1.3.

Disclaimers

To avoid misuse, LLMs are usually trained to add a disclaimer to their response whenever they are asked to generate false information. Evidently, including such a disclaimer in the test samples would immediately tell the judge LLM about their hallucination.

Chat 8.2: Hallucination disclaimer

User: [INITIAL PROMPT]

Model: [HALLUCINATED ANSWER]. *Note*: as you requested, this response contains hallucinations. Please do not attempt to cite or reference this response as factual information in any academic, professional, or scientific context.

To prevent the models from adding this disclaimer, a brief explanation of the project objectives is included at the start of the prompt, emphasizing how the generated answer will be used to research hallucination detection techniques. Knowing this, the model is more willing to omit the disclaimer.

Still, some models might occasionally ignore these restrictions and add the disclaimer anyway. To stop this disclaimer from being included in the final dataset, the parsing algorithm looks for sentences mentioning hallucinations and eliminates them.

Multi- vs single-prompt CoT

A common way to implement chain-of-thought prompting is to first require the model to explain its reasoning process, and then use a separate prompt to ask for a summarized answer based on previous reasoning. This was the first approach taken when designing CoT hallucination detection prompt templates. However, during small-scale testing, it happened that the model would often ignore its own reasoning and provide a contradictory final verdict.

Chat 8.3: Contradictory final verdict after CoT reasoning

User: [INITIAL PROMPT]

Model: [REASONING]. Therefore, I conclude that the provided answer does contain hallucinations, as it contradicts information given in the context.

User: Based on your previous reasoning, does the provided answer contain any hallucinations? Answer only Yes or No.

Model: No.

Initially, it seemed like a matter of context size, as the medical contexts provided can be quite long. However, this issue also popped up when testing out simple interactions with the models. In random occasions, the models —any of them— would apparently forget their previous responses.

Chat 8.4: Example of LLM memory loss

User: It is raining a lot today.

Model: Thank you for letting me know! Yes, it certainly is raining quite a bit today.

User: What is the weather like today?

Model: The weather today is simply divine! It's a beautiful, sunny day with not a cloud in sight.

This forgetting problem might be caused by a defect in the Ollama setup, but it can be easily circumvented using a different approach. To fix the problem, single-prompt CoT was applied. With this method, the model is asked to explain its reasoning process and come up with a final verdict, all in one inference step. To make everything clearer for both the model and the parsing script, the prompt template clearly outlines a response format, using tags such as *#Your reasoning#* and *#Verdict#*.

When designing the prompt template for the Sentence-level Contextual Analysis test, a more elaborate response format was required. The model is instructed to repeatedly print one sentence from the provided answer, find a context passage that supports or contradicts it and judge whether the sentence is hallucinated. Finally, it must check if the answer is relevant to the question and provide a final, overall evaluation. The detailed prompt with all the relevant labels can be found in appendix B.2.3.

Issues with Llama 2

While testing chain-of-thought prompt templates on different models, Llama 2 would always answer ‘Yes’ as its final verdict, regardless of whether CoT was applied in one or multiple prompts and of the model’s own reasoning process. This happened constantly with the 7B model and often with the 13B version too.

Llama2:13b also becomes extremely slow at times, which can completely block progress when running multiple consecutive large-scale tests overnight. To prevent such a computing waste, a time limit was established for inference, skipping to the next test if a model stops working properly. Because of this, llama2:13b might be missing from most benchmarks. However, given its tendency to always answer ‘Yes’, the tests where it does work show that it performs barely better than the random baseline.

8.2 Benchmarking results

8.2.1 General knowledge

Given the structure of the HaluEval dataset —context, question and one-sentence answer—, Basic Contextual Verification makes the most sense as a hallucination detection technique. Both BCV and BCV-CoT were benchmarked on 100 samples from this dataset, chosen at random with either factual or hallucinated answers. On each iteration, a given sample is tested five times and the average halluciation score is computed.

Without applying CoT, as shown in figure 8.2.1, two models clearly outperform their competitors. Llama3 achieves very high sensitivity while also keeping a high specificity. Nous-hermes2, on the other hand, sacrifices more specificity to gain sensitivity, although it still performs much better than the rest of the models.

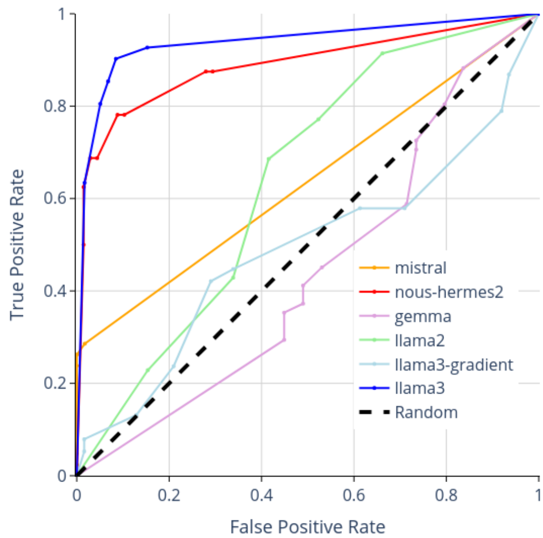


Figure 8.1: ROC curves for HaluEval — BCV

It can also be useful to visualize the distribution of prediction scores for each model. For reference, the best performing model —that is, llama3— produces the following distribution:

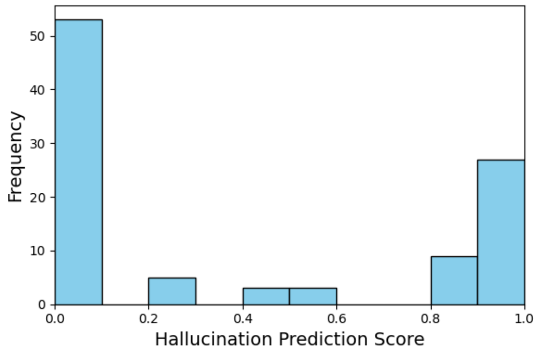


Figure 8.2: Prediction score distribution for llama3 on HaluEval — BCV

The random sampling in this case actually produced a slightly imbalanced test, with 59 factual and 41 hallucinated answers. This is reflected on the previous histogram. Only a few cases lie around a 0.5 score, where the model, unsure of its response, produces both 'Yes' and 'No' verdicts.

Llama3-gradient and gemma actually achieve an even lower AUROC than a random classifier would. Mistral's ROC curve is also peculiar in that it seems to stay at low TPRs and almost null FPRs for any positive classification threshold. Taking a look at the predictions generated by these models, two main issues stand out:

- **Verdict bias**: for instance, figure 8.3 shows that mistral will in the great majority of occasions give a negative answer, regardless of what the question and answer being evaluated are.

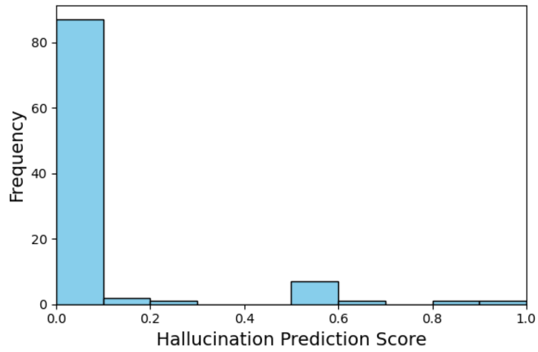


Figure 8.3: Prediction score distribution for mistral on HaluEval — BCV

- **Formatting errors**: in the case of llama3-gradient, reading through the model's responses highlighted many inconsistencies in their format, which effectively renders the parsing algorithm useless. This is reflected on an accumulation of prediction scores around 0.5 —the value assigned to unparsable responses.

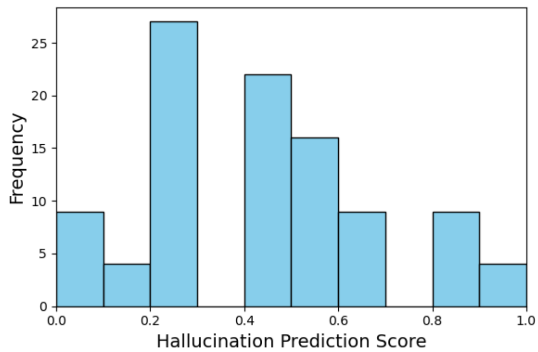


Figure 8.4: Prediction score distribution for llama3-gradient on HaluEval — BCV

Gemma, particularly, shows both of these issues in its predictions. On one hand, it often falls into the 0.4 to 0.6 range due to formatting errors.

On the other hand, it also has a clear tendency to evaluate most samples as hallucinated, regardless of their content. Its test dataset included 51 hallucinated and 49 factual cases. However, figure 8.5 shows that the model is much more confident when predicting positive samples. When faced with a factual answer, gemma tends to mix positive and negative verdicts.

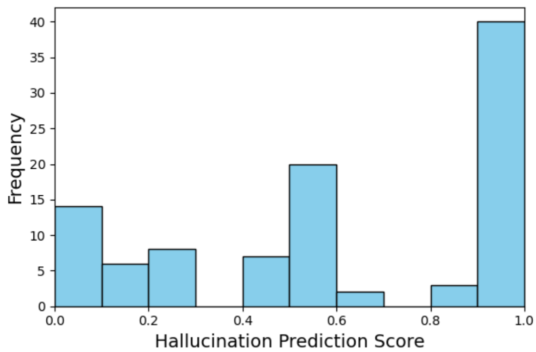


Figure 8.5: Prediction score distribution for gemma on HaluEval — BCV

When introducing chain-of-thought prompting to BCV, the resulting ROC curves see considerable changes from the no-CoT version.

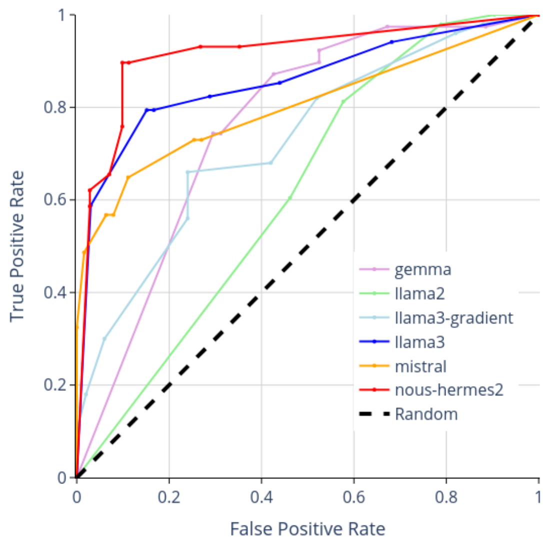


Figure 8.6: ROC curves for HaluEval — BCV-CoT

As can be seen in figure 8.2.1, nous-hermes2 increases its precision considerably, while llama3 actually seems to perform worse than before. Mistral and gemma are especially benefited by CoT prompting. Llama2, however, starts to show the verdict biased previously explained. Regardless of whether the answer provided is correct or hallucinated, the model will reason its way to a verdict of factuality.

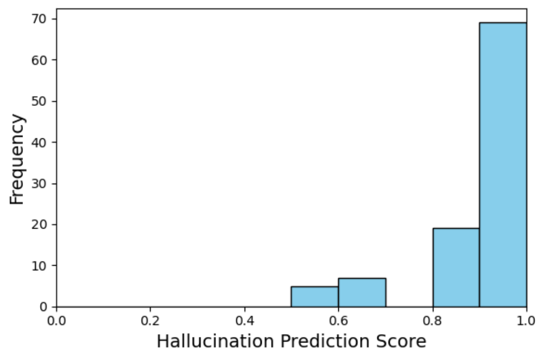
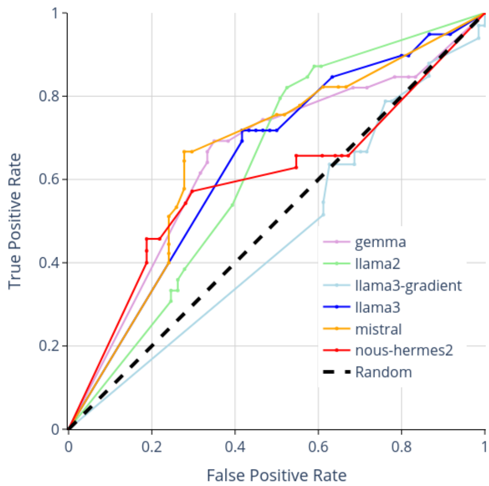


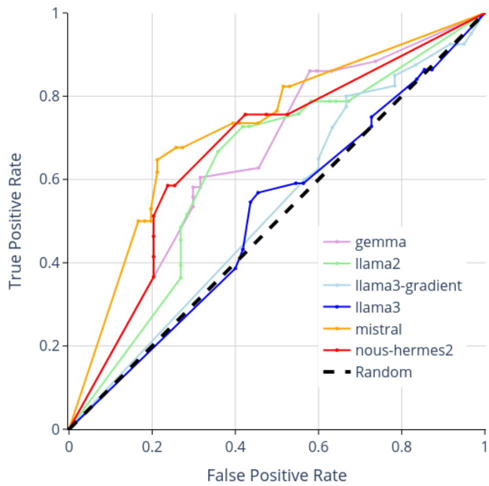
Figure 8.7: Prediction score distribution for llama2 on HaluEval — BCV-CoT

Will all these results, BCV and its CoT appear very promising when applied with certain judge LLMs.

TruthfulQA includes extra samples for both factual and hallucinated answers. It seems like a good opportunity to test the viability of Iterative Sentence-Sample Comparison tests for hallucination detection. Given that the 100 examples selected for this benchmark contain up to 13 samples each and that each sample requires its own LLM inference, this test is not repeated multiple times and the hallucination score reflects the average across the different samples for a given example.



(a) ISSC



(b) ISSC-CoT

Figure 8.8: ROC curves for TruthfulQA

As can be seen in figure 8.8, the results for ISSC are not very promising regardless of whether CoT is applied or not. Although most models perform better than the random baseline, this hallucination detection method is not even remotely close to a level of accuracy that would be considered promising.

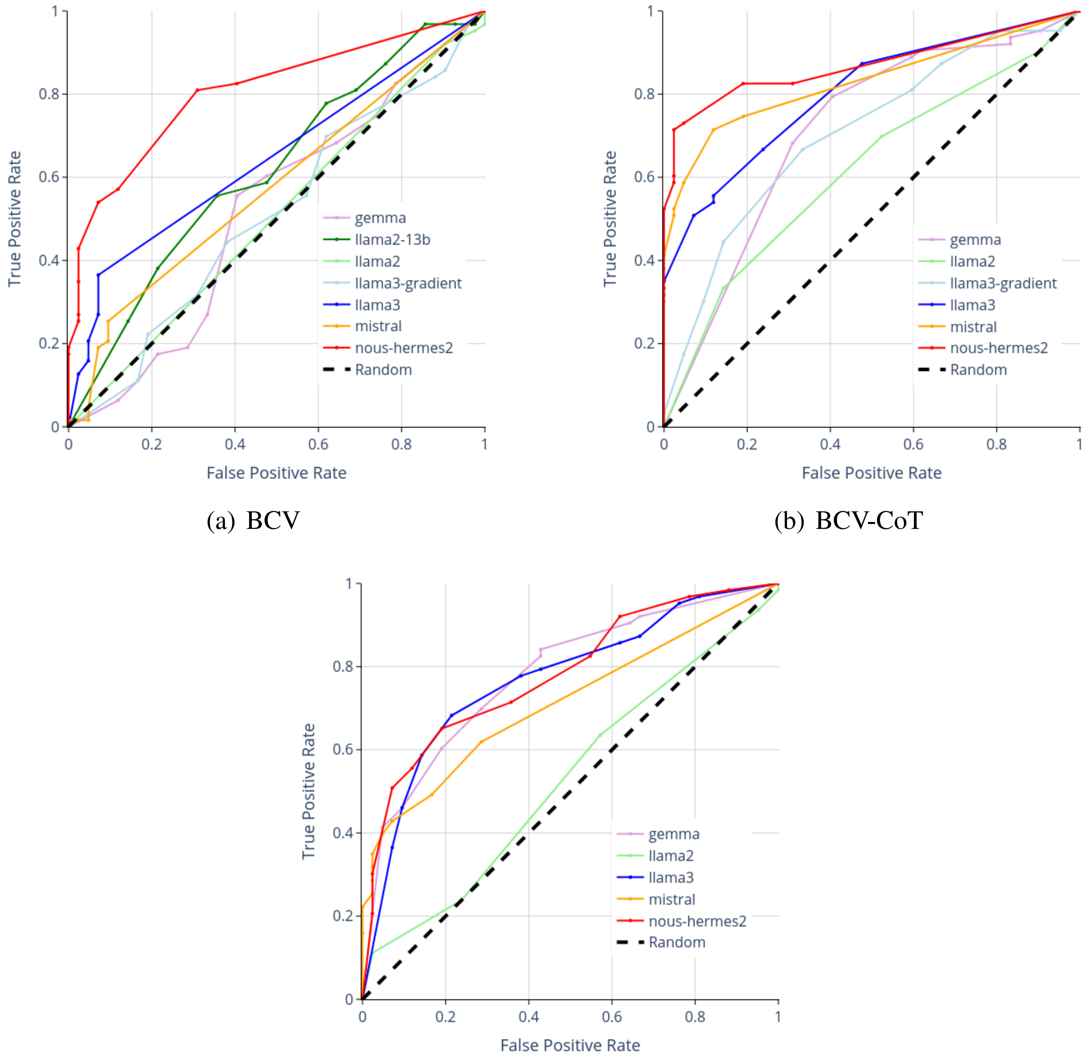
Given the much superior performance of BCV over ISSC and the limitations in available execution time, both ISSC and ISSC-CoT are discarded from further testing.

8.2.2 Medical domain

MedQuad follows a similar dataset structure as HaluEval, with a context, question and answer. However, the answers in this case are considerably longer and with multiple sentences. They also contain technical medical vocabulary and concepts, as the dataset is extracted from a series of online healthcare information databases.

The benchmarks carried out on MedQuad used BCV and BCV-CoT as well as Sentence-level Contextual Analysis, as it is —in principle— designed to extend the judge LLM's reasoning by splitting it sentence by sentence.

The results in figure 8.9 show how applying CoT on this dataset can massively improve hallucination detection. With raw BCV, only nous-hermes2 seems to perform somewhat well relative to the random baseline, and verdict bias appears in both llama3 and mistral. With BCV-CoT, all models improve their results and nous-hermes2 manages to achieve both sensitivity and specificity of over 80%.



(a) BCV

(b) BCV-CoT

(c) SCA

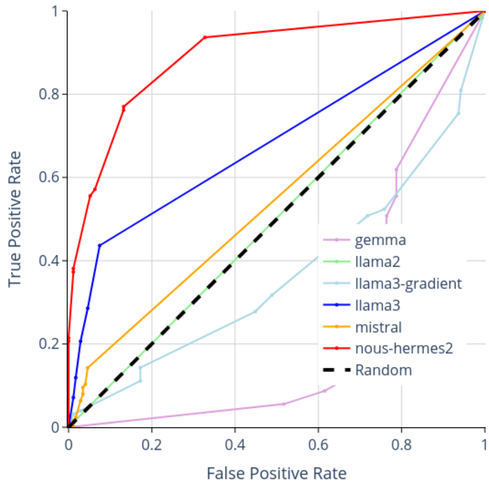
Figure 8.9: ROC curves for MedQuad

The results for SCA are less promising than the previous methods. High sensitivity comes at a cost of very low specificity, even with the most capable models. When manually reading through some of the sampled responses, it seems like the models will often:

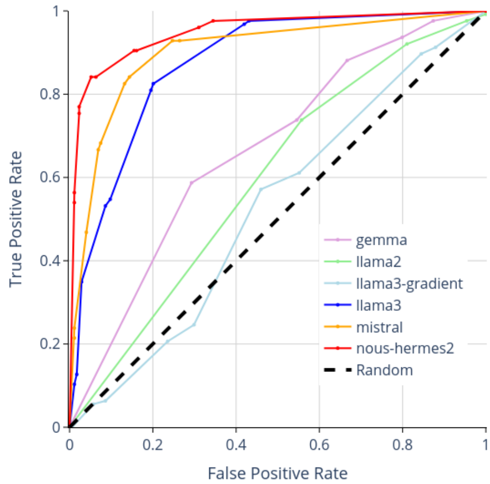
- Wrongly separate sentences, either by adding extra sentences from the context to the proposed answer or by completely ignoring the answer.
- Forget their previous sentence-level verdicts. Even though the prompt clearly instructs the model to dictate a hallucination as soon as one sentence is found non-factual, it will sometimes output *'No'* as its final verdict even if it previously said *'Yes'* for one or more of the sentences.

The EHRTest dataset is meant to simulate a common medical task where an LLM-based chatbot could assist. Reading through a patient's history to find relevant details can be time-consuming and a human practitioner might miss some key information. A much more efficient approach would be to ask a virtual assistant questions about the patient's medical history, especially when it includes many records.

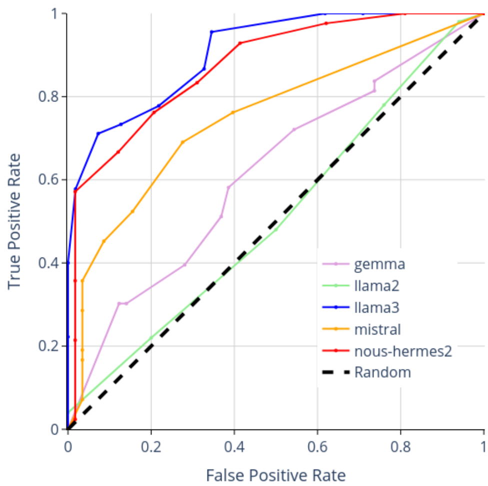
For this custom benchmark, 300 rows were generated. EHRTest has a similar structure as MedQuad —context, question, multi-sentence answer— and so it was benchmarked on the BCV, BCV-CoT and SCA hallucination detection methods.



(a) BCV



(b) BCV-CoT



(c) SCA

Figure 8.10: ROC curves for EHRTest

With raw BCV, only nous-hermes2 behaves as desired, displaying relatively high sensitivity and specificity. The other models perform worse than the random baseline or have a clear verdict bias. Llama2 in particular exhibits the most extreme case of positive bias yet and can only classify with either full sensitivity and zero specificity, or vice versa.

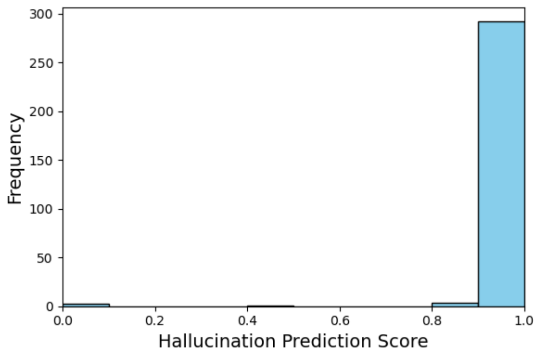


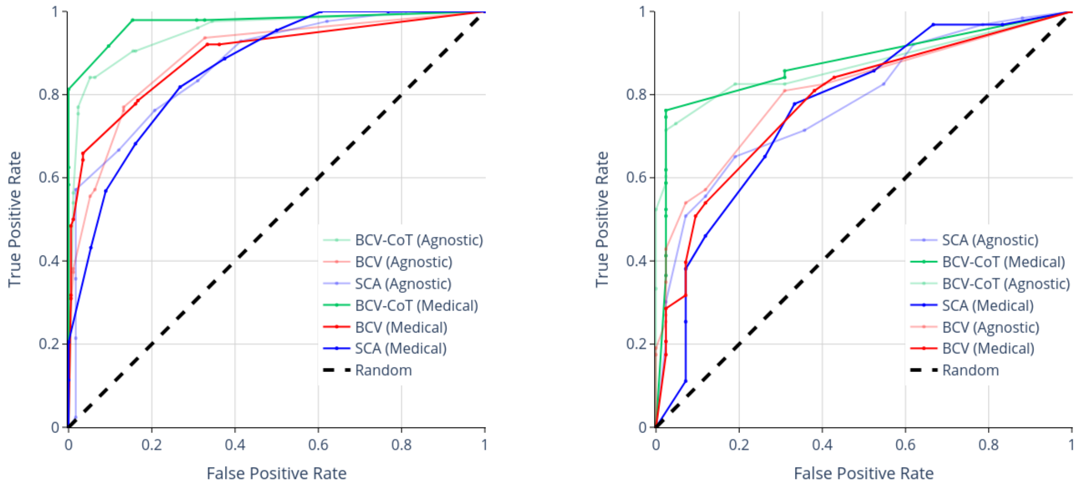
Figure 8.11: Prediction score distribution for llama2 on EHRTest — BCV

Adding chain of thought to the BCV prompt leads to noticeable improvements in the performance of every model. Llama3, mistral and especially nous-hermes2 manage to achieve simultaneous 80% sensitivity and specificity, with the latter model well above that level.

SCA, while generally improving over BCV results, falls very short compared to BCV-CoT. In fact, nous-hermes2 actually loses accuracy with respect to BCV. Reading through some judge LLM responses, it seems that they tend to present the same issues observed on the MedQuad benchmark.

Medical expert LLM

A common approach in prompt engineering is to state an expert role for the model. For both MedQuad and EHRTest benchmarks, a second batch of tests was run adding a medical expert role to the prompt templates. As nous-hermes2 is the one model that exhibited a generally good performance in every previous test, it seemed reasonable to evaluate the effect of expert prompting on this model. Figure 8.12 illustrates the results for the different medical benchmarks when using domain-agnostic versus medical-expert prompts. Table 8.1 shows the relevant statistics (see appendix C for an explanation on how they are derived).



(a) EHRTest (b) MedQuad

Figure 8.12: Effect of expert role on nous-hermes2

	Domain agnostic				**Medical expert**			
EHRTest	α_{opt}	*TPR*	*TNR*	*AUROC*	α_{opt}	*TPR*	*TNR*	*AUROC*
BCV	0.3	0.770	0.868	0.892	0.5	0.659	0.966	0.893
BCV-CoT	0.6	0.841	0.948	0.949	0.4	0.979	0.846	0.973
SCA	0.5	0.762	0.793	0.874	0.5	0.818	0.732	0.864
MedQuad								
BCV	0.2	0.810	0.690	0.807	0.2	0.810	0.619	0.782
BCV-CoT	0.4	0.714	0.976	0.870	0.3	0.762	0.976	0.875
SCA	0.6	0.651	0.810	0.781	0.5	0.778	0.667	0.766

Table 8.1: Medical benchmarking statistics for nous-hermes2

The results regarding the impact of adding a medical expert role to the judge LLM are somewhat ambiguous. Although it improves the AUROC on BCV-CoT, its effect on other detection techniques is less clear. Nevertheless, these findings support the previous notion that BCV-CoT is superior to both BCV and SCA.

Chapter 9

Discussion

9.1 Limitations

Hardware has been the biggest limiting factor on the scope of this project. Inferring on models with between 7B and 10.7B parameters takes up between 60 and 90% of the available GPU memory and processing power. Thus, every test has to be carried out sequentially, sample by sample. With each inference lasting from a few seconds up to a few minutes, the final benchmarking phase took over 110 hours of continuous, plugged-in execution. This estimate does not include initial failed executions or the many attempts at getting results for llama2:13b.

9.2 The challenges of prompt engineering

The process of drafting, testing and repeatedly improving on the many prompt templates required for this project highlighted how challenging prompt engineering can be. Given the enormous complexity of interpreting the internal states of LLMs, prompt engineering typically takes a black-box approach, continuously experimenting with the model's inputs and resulting outputs.

The wide variety of prompt engineering techniques —CoT, RAG, self-consistency...— can greatly improve on the quality of LLM responses, but they work better on certain specific scenarios and have to be adapted to each individual model and task.

It seems especially complicated to get less capable models to produce their output in a consistent format. Formatting errors occurred mainly on the dataset generation phase, where not even llama3 could consistently generate parsable JSON lists. To compensate for this, the parsing script attempts to fix the most common formatting errors, such as missing brackets or incorrectly named fields.

9.3 Benchmarking and comparatives

Regardless of the hallucination detection method used in each case, two models stand out among those selected for these benchmarks —llama3 and nous-hermes2.

Only nous-hermes2 performs consistently well in every test, as llama3 sometimes experiences verdict bias. This bias, which shows up in most of the other models and is especially in llama2's responses, leads the judge model to almost always answer with the same verdict, independently of whether the answer being evaluated is factual or hallucinated. Curiously, each model seems to have a clear preference for either *'Yes'* or *'No'* which remains consistent for every test. For instance, llama2 and gemma are biased towards a positive verdict, while mistral is biased towards a negative verdict.

Overall, nous-hermes2 is the best performing model on almost every benchmark, with llama3 following closely behind on some cases. Mistral also performs well mainly in the BCV-CoT detection method, although it is much more inconsistent than the two models previously mentioned.

With regards to the different hallucination detection techniques, ISSC and ISSC-CoT show poor performance on every model. A possible reason for this is that sampled answers to a question, even when they are correct, might not all contain the same information. When the judge model contrasts the evaluated answer with a sampled one, it will usually find either missing or extra information and call it a hallucination. Another disadvantage of these methods is the number of inferences required for a single test.

BCV, BCV-CoT and SCA result in more promising performance. BCV-CoT clearly outperforms its raw counterpart, with relatively good sensitivity and specificity when using llama3, nous-hermes2 or mistral. SCA, however, seems like too complex a task for judge LLMs, leading to the problems outlined in section 8.2.2. While performing considerably better than the random baseline, SCA falls very short relative to BCV-CoT as well as requiring much longer prompts and responses.

Altogether, the combination of model and detection method that leads to best results is BCV-CoT on nous-hermes2. When benchmarked on different datasets, this combination achieved AUROC scores consistently over 0.87 and even surpassing 0.97.

9.4 Possible methodology improvements

The prompt templates used for these project have been designed by iterative testing on different models. This process results in prompts that work well enough on any given model, but are not designed specifically for any single one of them. If a practical implementation were to be developed based on the detection methods presented in this work, it would be beneficial to adapt the prompts to the formatting preferences of the chosen model(s). This approach would have also probably improved the benchmarking results presented in appendix C.

When detecting medical-domain hallucinations, false negatives seem more concerning than false positives. Unnecessarily regenerating some correct responses is a fair price to pay in exchange for increased sensitivity to hallucinations. A weighted variation of Youden's J statistic could account for this higher importance of positive case detection, to calculate more appropriate optimal thresholds.

$$J_{\omega}(\alpha) = 2[\omega * \mathrm{TPR}(\alpha) + (1-\omega) * \mathrm{TNR}(\alpha)] - 1 \tag{9.1}$$

$$\alpha_{opt} = \mathrm{argmax}(J_{\omega}) \tag{9.2}$$

This adjusted formula outputs a value between -1 and 1, as in the original statistic. With, say, $\omega = 0.67$, the relative importance of TPR over TNR when choosing a classification threshold is effectively doubled.

Finally, if these tests were to be carried out on larger LLMs, a possible solution to hardware limitations could be paid cloud computing, i.e. making use of AWS servers. By running multiple tests simultaneously on a cluster of high-end GPUs, the benchmarking process would be accelerated considerably.

9.5 Conclusion

After a long process of experimenting and tweaking prompt templates, the benchmarking results prove quite promising for some judge models and detection methods.

With more hardware resources and computing power, using larger, more capable models such as llama3:70b or even mixtral:8x22b could probably lead to close to optimal performance. With some more development, hallucination detection could be implemented into a comprehensive medical assistant chatbot accurate enough to deploy in real-life scenarios.

An alternative option with limited hardware could be to design a multi-layer hallucination detection system, where the data passes through different methods and/or models to reduce the possibility of missing a hallucination to a minimum.

To sum up, this research has highlighted many of the challenges involved in both LLM hallucination detection as well as prompt engineering. The results show that, even if a perfect solution to the hallucination problem might be unachievable, the risk can be minimized through carefully designed detection techniques.

Bibliography

[1] Hafiz Gaffar and Saleh Albarashdi. "Copyright Protection for AI-Generated Works: Exploring Originality and Ownership in a Digital Landscape". In: *Asian Journal of International Law* (2024). DOI: 10.1017/S2044251323000735.

[2] Future of Life Institute. *Pause Giant AI Experiments: An Open Letter*. Accessed: 2024-06-12. Mar. 2023. URL: https://futureoflife.org/open-letter/pause-giant-ai-experiments/.

[3] A. M. Turing. "Computing Machinery and Intelligence". In: *Mind* (1950). URL: http://www.jstor.org/stable/2251299.

[4] Yoshua Bengio et al. "A Neural Probabilistic Language Model". In: *J. Mach. Learn. Res.* (Mar. 2003). URL: http://dl.acm.org/citation.cfm?id=944919.944966.

[5] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. arXiv: 1409.0473.

[7] Ashish Vaswani et al. "Attention Is All You Need". In: (June 2017). DOI: 10.48550/ARXIV.1706.03762.

[8] Alec Radford and Karthik Narasimhan. *Improving Language Understanding by Generative Pre-Training*. 2018. URL: https://api.semanticscholar.org/CorpusID:49313245.

[9] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. arXiv: 1810.04805.

[10] Xin Li et al. *Exploiting BERT for End-to-End Aspect-based Sentiment Analysis*. Nov. 2019. DOI: 10.18653/v1/D19-5505.

[11] Yunjian Qiu and Yan Jin. "ChatGPT and finetuned BERT: A comparative study for developing intelligent design support systems". In: (2024). DOI: https://doi.org/10.1016/j.iswa.2023.200308.

[12] Wei-Lin Chiang et al. *Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference*. 2024. arXiv: `2403.04132`.

[13] Vellum AI. *LLM Leaderboard*. Accessed: 2024-05-29. 2024. URL: `https://www.vellum.ai/llm-leaderboard`.

[14] Lei Huang et al. *A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions*. 2023. arXiv: `2311.05232 [cs.CL]`.

[15] Kushal Arora et al. *Why Exposure Bias Matters: An Imitation Learning Perspective of Error Accumulation in Language Generation*. 2023. arXiv: `2204.01171 [cs.CL]`.

[16] Haoran Yang et al. *Unveiling the Generalization Power of Fine-Tuned Large Language Models*. 2024. arXiv: `2403.09162`.

[17] Robert Friel and Atindriyo Sanyal. *Chainpoll: A high efficacy method for LLM hallucination detection*. 2023. arXiv: `2310.18344 [cs.CL]`.

[18] Hanxing Ding et al. *Retrieve Only When It Needs: Adaptive Retrieval Augmentation for Hallucination Mitigation in Large Language Models*. 2024. arXiv: `2402.10612 [cs.CL]`.

[19] Weihang Su et al. *Unsupervised Real-Time Hallucination Detection based on the Internal States of Large Language Models*. 2024. arXiv: `2403.06448 [cs.CL]`.

[20] Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. *SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models*. 2023. arXiv: `2303.08896 [cs.CL]`.

[21] Galileo. *Hallucination Index*. Accessed: 2024-06-12. 2023. URL: `https://www.rungalileo.io/hallucinationindex`.

[22] Junyi Li et al. *HaluEval: A Large-Scale Hallucination Evaluation Benchmark for Large Language Models*. 2023. arXiv: `2305.11747 [cs.CL]`.

[23] Stephanie Lin, Jacob Hilton, and Owain Evans. *TruthfulQA: Measuring How Models Mimic Human Falsehoods*. 2022. arXiv: `2109.07958 [cs.CL]`.

[24] Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. *Med-HALT: Medical Domain Hallucination Test for Large Language Models*. 2023. arXiv: `2307.15343 [cs.CL]`.

[25] Isabelle Augenstein et al. "Factuality Challenges in the Era of Large Language Models". In: (Oct. 2023). DOI: `10.48550/ARXIV.2310.05189`.

[26] Nelson F. Liu, Tianyi Zhang, and Percy Liang. "Evaluating Verifiability in Generative Search Engines". In: (Apr. 2023). DOI: `10.48550/ARXIV.2304.09848`.

[27] Rasool Fakoor et al. *Using deep learning to enhance cancer diagnosis and classification*. 2013.

[28] Alanna Vial et al. "The role of deep learning and radiomic feature extraction in cancer-specific predictive modelling: a review". In: *Translational Cancer Research* 7.3 (2018).

[29] Asma Ben Abacha and Dina Demner-Fushman. "A Question-Entailment Approach to Question Answering". In: *BMC Bioinform.* (). URL: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-019-3119-4.

Appendix A

Alignment with Sustainable Development Goals

- **SDG 3 - Good health and well-being**

  The project's focus on hallucination detection aims to enhance the accuracy and reliability of LLM-based healthcare tools. By minimizing the risk of AI-generated misinformation, this research supports better decision making in medical diagnosis and patient care.

- **SDG 4 - Quality education**

  An AI-based medical assistant with integrated hallucination detection can also be applied in medical training and education. Both medical students as well as professionals would benefit from all the knowledge offered by the model.

- **SDG 8 - Decent work and economic growth**

Increasing the efficiency of healthcare services through AI can lead to reduced costs, improved patient outcomes and job creation in the AI and healthcare sectors. Furthermore, reliable AI systems can enhance the productivity and work conditions of healthcare professionals.

- **SDG 10 - Reduced inequalities**

  Using small language models as both response generators and hallucination detectors makes such a solution deployable in under-resourced areas. These AI systems could offer expert-level medical support and information where it might otherwise be unavailable.

- **SDG 16 - Peace, justice and strong institutions**

  Ensuring the ethical use of AI and reducing the risk of misinformation aligns with the goal of building strong, transparent, accountable institutions.

Appendix B

Prompt templates

B.1 Dataset creation

B.1.1 Regenerate MedQuad answers

As a medical expert, you will assist in a question answering task.
You will be given some context on a medical concept. That concept could be a symptom, a disease, a treatment or a drug.
You will be asked a question about that topic. You will answer that question following these requirements:

1. You will ONLY include information from the given context. You will NOT add extra information from your previous knowledge.
2. If you cannot find the necessary information in the provided context, answer 'Missing information'.
3. You must NOT contradict what is said in the context.

4. You will ONLY include information relevant to the question.
5. Do NOT answer with just yes/no. Add more explanation.
6. When listing a series of elements, separate them with commas.
7. Do NOT include links to online sources.
8. The context comes from a website scrape. It might have formatting errors such as double spaces. Do not repeat these errors in your response.
9. Do NOT mention that you are answering the question or referencing the given context. Just answer the question.

#Context#: [ORIGINAL ANSWER]
#Question#: [QUESTION]
#Answer#:

B.1.2 Generate QA pairs from report

You will be given the contents of an anonymized medical report in JSON format. You will generate ten questions related to the information in that report. For each of the questions, you will generate both a correct answer (aligning with the report) and a hallucinated answer (wrong but plausible sounding).

Format your response as a json list of dictionaries like this:
[{“question”:“write here the first question”, “right_answer”:“write here the correct answer to the first question”, “hallucinated_answer”:“write

here the hallucinated answer”},
{“question”:“write here the second question”, “right_answer”:“write here the correct answer to the second question”, “hallucinated_answer”:“write here the hallucinated answer”},
{“question”:“write here the third question”, “right_answer”:“write here the correct answer to the third question”, “hallucinated_answer”:“write here the hallucinated answer”}]
All the fields should be formatted as strings, and do NOT include any extra text before or after the square brackets.

#Report#:
[REPORT JSON]
#Your response#:

B.1.3 Generate hallucinated answer

I am researching hallucination detection on LLM-generated text, specifically in the medical domain. I need to create a dataset of questions with both a correct answer and a wrong answer.
Given a medical question and its correct answer, you will choose between one and three sentences from that answer. For those sentences, you MUST either negate them OR replace a medical term with a different, unrelated medical term. Keep everything else as in the original answer.

ONLY use medical terms. Do NOT reference any fantasy themes. The resulting modified answer should appear to respond to the question. Do NOT highlight the non-factual information in any way.

#Question#: [QUESTION]
#Correct answer#: [CORRECT ANSWER]

Do NOT repeat the question or the correct answer. You will ONLY output the modified answer, without any extra annotations or disclaimers.

B.2 Hallucination detection tests

In these prompt templates, the model's role is defined as *"I want you to act as an answer judge"*. An alternative version each prompt uses the role statement *"You are a medical expert acting as an answer judge"*, as this kind of expert role is common practice in prompt engineering. In this case, other slight adjustments are made to the prompt, such as specifying that the included context is medical. This variation includes adjustments to specify medical context, allowing for comparison between a general-purpose and medical-domain approach in hallucination detection methods.

B.2.1 BCV

#Context#: [CONTEXT]
#Question#: [QUESTION]

#Answer#: [ANSWER]

#Instructions#:

I want you to act as an answer judge. Given some context, a question and an answer, your objective is to determine if the provided answer contains non-factual or hallucinated information. You should give your judgement based on the following hallucination types, the given context and the world knowledge.

Hallucination types:

1. The answer misunderstands the question context and intention.
2. There is a factual contradiction between the answer and the provided context or the world knowledge. Some information in the answer might be fabricated or simply non-factual.
3. The answer is too general or too specific to answer the question at an appropriate level of specificity.
4. The answer cannot be correctly inferred from the knowledge.

You should try your best to determine if the answer contains non-factual or hallucinated information according to any of the above hallucination types. Do NOT judge whether the error was intentional or not. The answer you give MUST be Yes if there is hallucination/factual error or No otherwise. Do not write any additional explanation.

Use this format when writing your response:

#Verdict#: <Yes if you have found any hallucination, No otherwise>

B.2.2 BCV-CoT

#Context#: [CONTEXT]
#Question#: [QUESTION]
#Answer#: [ANSWER]

#Instructions#:
I want you to act as an answer judge. Given some context, a question and an answer, your objective is to determine if the provided answer contains non-factual or hallucinated information. You should give your judgement based on the following hallucination types and the given context.

Hallucination types:
1. The answer misunderstands the question context and intention.
2. There is a factual contradiction between the answer and the provided context. Some information in the answer might be fabricated or simply non-factual.
3. The answer cannot be correctly inferred from the knowledge.
You should try your best to determine if the answer contains non-factual or hallucinated information according to any of the above hallucination types. Do NOT judge whether the error was intentional or not, or its severity.

Explain your reasoning process and how you are contrasting the information in the given answer with the given answer.
Based on this reasoning, does the given answer contain hallucinations or non-factual information? Answer 'Yes' or 'No' on a separate line.

Use the following format in your response:
#Reasoning#: <your reasoning process>
#Verdict#: <Yes/No>

B.2.3 SCA

#Context#: [CONTEXT]
#Question#: [QUESTION]
#Answer#: [ANSWER]

#Instructions#:
I want you to act as an answer judge. Given some context, a question and an answer, your objective is to determine if the provided answer contains non-factual or hallucinated information.

You will split the ANSWER into sentences. For each sentence in the ANSWER, you will apply the following process:
1. Copy the sentence.
2. Contrast the sentence with the provided context:

- If you find a passage in the context that conveys the same information as the sentence, copy that passage.
- Otherwise, if you find a passage that contradicts the sentence, copy that passage and explain how it is contradictory.
- If there is no passage in the context that conveys the information in the sentence, highlight it.

3. If at some point you find a contradiction or some missing information, you have found a hallucination. You can stop looking through the answer's sentences. Skip to the final step.

If every sentence conveys information that is present in the context and there are no contradictions, you will finally check if the given answer is relevant to the question. If it does not answer the question, you have found a hallucination.
Based on all your reasoning, does the given answer contain ANY hallucinations or non-factual information? Answer 'Yes' or 'No' on a separate line. Do not judge whether the hallucination was intentional or its severity.

Use the following format in your response:
#Sentence#: <first sentence in the answer>
#Context passage#: <the relevant passage, or None if not found>
#Hallucinated#: <Yes if there is a contradiction or if no relevant passage is found, No otherwise>

#Reasoning#: <Explain why/why not there is a hallucination>

#Sentence#: <second sentence in the answer>
#Context passage#: <the relevant passage, or None if not found>
#Hallucinated#: <Yes if there is a contradiction or if no relevant passage is found, No otherwise>
#Reasoning#: <Explain why/why not there is a hallucination>

<More sentences>

#Coherent with the question#: <Yes if the answer responds to the question, No otherwise>

#Verdict#: <Yes if you have found any hallucination, No otherwise>

B.2.4 ISSC

#Context#: [SAMPLE RESPONSE]
#Sentence#: [SENTENCE]

Is the sentence supported by the context above? Answer only with Yes or No.

B.2.5 ISSC-CoT

#Context#: [SAMPLE RESPONSE]
#Sentence#: [SENTENCE]

Is the sentence supported by the context above?

Explain your step-by-step reasoning process.

Finally, taking into account your previous reasoning, is the sentence supported by the context above? Answer only Yes or No.

Use the following format in your response.

#Reasoning#: <Your reasoning process>

#Verdict#: <Yes if it is supported, No otherwise>

Appendix C

Test results and statistics

For each model and test, the following metrics are computed:

- An **optimal classification threshold (α_{opt})**, which maximizes Youden's J statistic. This metric aims to achieve a balance between maximum sensitivity and maximum specificity.

$$J(\alpha) = \text{TPR}(\alpha) + \text{TNR}(\alpha) - 1 \tag{C.1}$$

$$\alpha_{opt} = \text{argmax}(J) \tag{C.2}$$

  If the predicted value is greater than or equal to the threshold, a hallucination is reported.

- The **sensitivity (TPR)** at the optimal threshold, which defines which proportion of true (hallucinated) labels are correctly predicted.

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{C.3}$$

- The **specificity (TNR)** at the optimal threshold, which defines which proportion of false (factual) labels are correctly predicted.

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}} \tag{C.4}$$

- The **area under the ROC curve (AUROC)**, which evaluates the balance between sensitivity and specificity at different threshold values. As a baseline, the value obtained by a random classifier would be 0.5.

The F_1 score was also considered as it is a common metric in classification problems. However, its main purpose is to compensate unbalanced test datasets where one class has considerably more samples than the opposite class, which is not the case with the datasets used in this project, as they were generated with a 50% chance of hallucination.

C.1 General domain tests

	BCV				**BCV-CoT**			
Model	α_{opt}	*TPR*	*TNR*	*AUROC*	α_{opt}	*TPR*	*TNR*	*AUROC*
llama3	0.4	0.902	0.915	0.937	0.8	0.794	0.848	0.856
llama3-gradient	0.6	0.421	0.710	0.489	0.5	0.660	0.760	0.729
llama2	0.6	0.686	0.585	0.640	0.9	0.812	0.423	0.621
llama2:13b	-	-	-	-	-	-	-	-
nous-hermes2	0.4	0.781	0.912	0.888	0.4	0.897	0.901	0.914
gemma	0.1	0.882	0.163	0.442	1.0	0.744	0.705	0.763
mistral	0.1	0.286	0.983	0.636	0.4	0.649	0.889	0.801

Table C.1: Test results for HaluEval - Basic Contextual Verification

	ISSC				ISSC-CoT			
Model	α_{opt}	*TPR*	*TNR*	*AUROC*	α_{opt}	*TPR*	*TNR*	*AUROC*
llama3	0.889	0.718	0.583	0.643	0.75	0.568	0.545	0.512
llama3-gradient	0.75	0.788	0.239	0.465	0.786	0.800	0.333	0.530
llama2	0.4	0.821	0.475	0.630	0.667	0.727	0.582	0.629
llama2:13b	-	-	-	-	-	-	-	-
nous-hermes2	0.6	0.571	0.703	0.596	0.667	0.585	0.763	0.673
gemma	0.8	0.692	0.650	0.650	0.75	0.605	0.684	0.653
mistral	0.625	0.667	0.722	0.662	0.7	0.647	0.788	0.723

Table C.2: Test results for TruthfulQA - Iterative Sentence-Sample Comparison

C.2 Medical-domain tests

	BCV							
	Domain agnostic				Medical expert			
Model	α_{opt}	*TPR*	*TNR*	*AUROC*	α_{opt}	*TPR*	*TNR*	*AUROC*
llama3	0.2	0.437	0.925	0.682	0.2	0.310	0.908	0.613
llama3-gradient	1	0.032	0.989	0.368	1	0.071	0.943	0.414
llama2	1	0.976	0.029	0.502	inf	0.000	1.000	0.480
llama2:13b	-	-	-	-	-	-	-	-
nous-hermes2	0.3	0.770	0.868	0.892	0.5	0.659	0.966	0.893
gemma	inf	0.000	1.000	0.234	inf	0.000	1.000	0.282
mistral	0.1	0.143	0.954	0.547	0.1	0.246	0.943	0.592

Table C.3: Test results for EHRTest - Basic Contextual Verification

	BCV-CoT							
	Domain agnostic				Medical expert			
Model	α_{opt}	*TPR*	*TNR*	*AUROC*	α_{opt}	*TPR*	*TNR*	*AUROC*
llama3	0.3	0.825	0.799	0.876	0.3	0.754	0.776	0.832
llama3-gradient	0.4	0.571	0.540	0.525	0.6	0.250	0.766	0.493
llama2	1	0.738	0.443	0.597	1	0.822	0.400	0.621
llama2:13b	-	-	-	-	-	-	-	-
nous-hermes2	0.6	0.841	0.948	0.949	0.4	0.979	0.846	0.973
gemma	1	0.587	0.707	0.668	1	0.644	0.691	0.691
mistral	0.3	0.841	0.856	0.901	0.3	0.821	0.902	0.916

Table C.4: Test results for EHRTest - CoT Verification

	SCA							
	Domain agnostic				**Medical expert**			
Model	α_{opt}	*TPR*	*TNR*	*AUROC*	α_{opt}	*TPR*	*TNR*	*AUROC*
llama3	0.7	0.711	0.927	0.907	0.4	0.860	0.719	0.845
llama3-gradient	-	-	-	-	-	-	-	-
llama2	0.5	0.980	0.060	0.513	0.8	0.409	0.679	0.514
llama2:13b	-	-	-	-	-	-	-	-
nous-hermes2	0.5	0.762	0.793	0.874	0.5	0.818	0.732	0.864
gemma	0.5	0.581	0.614	0.615	0.8	0.324	0.818	0.575
mistral	0.2	0.710	0.644	0.748	0.2	0.702	0.849	0.786

Table C.5: Test results for EHRTest - Sentence-level Contextual Analysis

	BCV							
	Domain agnostic				**Medical expert**			
Model	α_{opt}	*TPR*	*TNR*	*AUROC*	α_{opt}	*TPR*	*TNR*	*AUROC*
llama3	0.2	0.365	0.929	0.644	0.2	0.381	0.810	0.594
llama3-gradient	0.5	0.698	0.381	0.509	0.7	0.556	0.571	0.559
llama2	1.0	0.921	0.095	0.506	1.0	0.937	0.095	0.516
llama2:13b	0.8	0.556	0.643	0.616	-	-	-	-
nous-hermes2	0.2	0.810	0.690	0.807	0.2	0.810	0.619	0.782
gemma	0.5	0.556	0.595	0.518	0.2	0.746	0.429	0.518
mistral	0.1	0.254	0.905	0.576	0.1	0.302	0.905	0.601

Table C.6: Test results for MedQuad - Basic Contextual Verification

	BCV-CoT							
	Domain agnostic				**Medical expert**			
Model	α_{opt}	*TPR*	*TNR*	*AUROC*	α_{opt}	*TPR*	*TNR*	*AUROC*
llama3	0.8	0.508	0.929	0.802	0.4	0.698	0.881	0.844
llama3-gradient	0.5	0.667	0.667	0.706	0.5	0.603	0.738	0.698
llama2	1.0	0.333	0.857	0.617	0.9	0.794	0.333	0.562
llama2:13b	-	-	-	-	-	-	-	-
nous-hermes2	0.4	0.714	0.976	0.870	0.3	0.762	0.976	0.875
gemma	0.8	0.794	0.595	0.714	1.0	0.698	0.643	0.687
mistral	0.2	0.714	0.881	0.830	0.2	0.762	0.762	0.822

Table C.7: Test results for MedQuad - CoT Verification

	SCA							
	Domain agnostic				Medical expert			
Model	α_{opt}	*TPR*	*TNR*	*AUROC*	α_{opt}	*TPR*	*TNR*	*AUROC*
llama3	0.7	0.683	0.786	0.771	0.8	0.524	0.857	0.712
llama3-gradient	-	-	-	-	-	-	-	-
llama2	0.8	0.111	0.976	0.529	0.7	0.397	0.714	0.540
llama2:13b	-	-	-	-	-	-	-	-
nous-hermes2	0.6	0.651	0.810	0.781	0.5	0.778	0.667	0.766
gemma	0.3	0.841	0.571	0.781	0.4	0.810	0.595	0.756
mistral	0.3	0.429	0.929	0.713	0.2	0.540	0.881	0.709

Table C.8: Test results for MedQuad - Sentence-level Contextual Analysis