Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

Building an Interpretable Natural Language AI Tool based on Transformer Models and Approaches of Explainable AI

Bachelorarbeit/Masterarbeit

Name des Studiengangs

Wirtschaftsinformatik

Fachbereich 4

vorgelegt von

Lennard Zündorf

Datum:

Berlin, 14.02.2024

Erstgutachter: Frau Prof. Dr. Simbeck

Zweitgutachter: Herr Prof. Dr. Hochstein

Abstract

English

The rapid advancement of Artificial Intelligence, particularly in the context of transformer-based Large Language Models, has ushered in a new era in consumer and business software. However, a significant aspect often overlooked is the integration of Explainable AI, specifically interpretable machine learning approaches. This thesis explores the relevance and applicability of existing XAI methods to contemporary transformer-based models, widely used in applications such as ChatGPT. Through the development of an explainable application prototype, this study assesses the feasibility of integrating XAI techniques into practical, industry-level applications. To this end, open-source technologies were employed to create a usable chatbot application prototype that provides decision-making explanations. Despite the successful development of a functional prototype, challenges remain in achieving user-friendly interpretability, particularly with complex models. Thus, this study demonstrates the feasibility of integrating XAI into chat applications but also highlights the vital areas for future research in enhancing post hoc interpretability methods. The prototype reveals limitations in current post hoc interpretability methods and underscores the need for improved approaches, especially considering the increasing size and complexity of LLMs.

Deutsch

Die rasante Entwicklung der künstlichen Intelligenz, insbesondere im Zusammenhang mit transformer-basierten Large Language Models (LLMs), hat eine neue Ära in der Verbraucher- und Unternehmenssoftware eingeläutet. Ein wichtiger Aspekt, der jedoch oft übersehen wird, ist die Integration von Explainable AI (XAI), insbesondere von interpretierbaren machine learning Ansätzen. Diese Arbeit untersucht die Relevanz und Anwendbarkeit bestehender XAI-Methoden auf moderne transformer-basierte Modelle, die in Anwendungen wie ChatGPT weit verbreitet sind. Durch die Entwicklung eines erklärbaren Anwendungsprototyps bewertet diese Studie die Machbarkeit der Integration von XAI-Techniken in praktische, industrielle Anwendungen. Zu diesem Zweck wurden Open-Source-Technologien eingesetzt, um einen brauchbaren Chatbot-Anwendungsprototyp zu erstellen, der Erklärungen zur Entscheidungsfindung liefert. Trotz der erfolgreichen Entwicklung eines funktionalen Prototyps bestehen weiterhin Herausforderungen bei der benutzerfreundlichen Interpretierbarkeit, insbesondere bei komplexen Modellen wie Mistral 7B. Der Prototyp zeigt die Grenzen aktueller Post-hoc-Interpretationsmethoden auf und unterstreicht den Bedarf an verbesserten Ansätzen, insbesondere angesichts der zunehmenden Größe und Komplexität von LLMs.

Contents

Table of Abbreviations IV

Table of Figures V

Table Directory VI

1 Introduction 1

1.1 Objectives of This Work 1

1.2 The Importance of Explainable AI 2

2 Fundamentals 3

2.1 Transformer Models 4

2.1.1 Model Differences 5

2.1.2 Large Language Models 6

2.2 Explainable AI 9

2.2.1 Taxonomy of Explainable AI 10

2.2.2 Overview of Interpretability Methods 14

2.2.3 Evaluating Interpretability Methods 20

2.3 Chatbots 21

3 Current State of Research 22

4 Implementation 23

4.1 Used Technologies and Methods 24

4.1.1 Notable Used Technologies 24

4.1.2 Code Quality 26

4.1.3 Evaluation Framework 27

4.2 LLM Interpretability 28

4.2.1 Exploration 29

4.2.2 Evaluation 29

4.3 Application Prototype 30

4.3.1 Application Architecture 31

4.3.2 Interpretability Implementation 32

4.3.3 User Interface 34

4.3.4 Hardware and Hosting 34

5 Results 35
5.1 LLM Interpretability 35
5.1.1 Explanation Quality 35
5.1.2 Runtime 37
5.2 Implementation Evaluation 39
5.2.1 Qualitative Evaluation 39
5.2.2 Framework Based Evaluation 41
5.3 Discussion 43
5.3.1 Explanation Quality 43
5.3.2 Helpfulness of Explanations 44
5.3.3 Runtime and Scalability 45
6 Conclusion 47
Publication Bibliography VIII
Glossary XVI
Annex XVII

Table of Abbreviations

Abbreviation	Full Word
AI	Artificial Intelligence
LLM	Large Language Model
XAI	Explainable Artificial Intelligence
NLP	Natural Language Processing
UX/UI	User Experience / User Interface
7B	Seven billion (parameters)
GODEL	Grounded Open Dialogue Language Model
LlaMa (2)	Large Language Model Meta AI
GPT	General Pre-Trained Transformer

Table of Figures

Figure 1: Visual depiction of the transformers architecture. 5
Figure 2: Graphic showcasing taxonomy of XAI approaches. 14
Figure 3: Visual depiction of SHAP values. 18
Figure 4: Graphic depiction of application architecture. 31
Figure 5: Screenshot of custom explanation visualization. 33
Figure 6: Screenshot of interactive shap text graphic. 33
Figure 7: Sequence attribution graph for Mistral with KernelSHAP. 36
Figure 8: Graphic displaying runtime of different interpretability method and model combinations. 38
Figure 9: Graphic displaying runtime of Shapley Value Sampling with different models. 39
Figure 10: Screenshot of SHAP based explanation. 41
Figure 11: Graphic displaying evaluation scores and score delta. 42

Table Directory

Table 1: Overview of different large language models. 9

Table 2: Overview of different interpretability approaches. 15

Table 3: Overview of average helpfulness scores and delta. 43

1 Introduction

The absence of explanation and interpretability of Artificial Intelligence (AI) has historically been recognized as a barrier to the adoption of advanced machine learning models into consumer and business software (Arrieta et al. 2019). However, in the rise of AI chatbots and underlying Large Language Models (LLM), businesses and research organizations seem unconcerned with Explainable AI. This is demonstrated by the fact that neither AI-powered software such as ChatGPT[1], Claude[2], HuggingChat [3]nor widely used AI infrastructure providers like Azure OpenAI Services[4] or Amazon AWS Bedrock[5] have integrated principles of Explainable AI (XAI) into their services - specifically, the integration of approaches of interpretable machine learning like the SHAP framework (Lundberg and Lee 2017). This group of companies includes model and software developing firms like OpenAI as well as Fortune 500 companies utilizing these models, as more than 90% of them are building with models of for example the GPT Family (Field 2023). Since 2022, many new and more powerful LLMs have been released, propelling progress in Natural Language Processing (NLP) (Zhao et al. 2023b; Loukides 2023). These breakthroughs have built upon what is, technologically speaking, an already established advancement in AI: the transformer architecture (Lee 2024; Woodie 2021). At the same time, research has advanced in other connected fields like AI Governance AI Safety and Ethical AI. However, a comprehensible explanation for Large Language Model outputs remains elusive (Zhao et al. 2023a).

Given these developments, at least two pertinent questions arise. Firstly, the question of how relevant and applicable existing XAI methods are to contemporary models used for widely used available applications like ChatGPT. Secondly, there is the challenge of how these approaches can be integrated into applications and provide value to application users, if method should prove applicable, as per the core premise of XAI.. The aim of this study was to explore, evaluate, and implement these questions. It specifically investigates the potential for implementations of XAI principles and existing approaches in practical applications based on models with a transformer architecture.

1.1 Objectives of This Work

This study presents an explainable application prototype powered by modern transformer-based LLMs and machine learning interpretability approaches. Furthermore, it provides an overview of the

---

[1] SEE HTTPS://CHAT.OPENAI.COM/
[2] SEE HTTPS://HUGGINGFACE.CO/CHAT/
[3] SEE HTTPS://CLAUDE.AI/
[4] SEE HTTPS://AZURE.MICROSOFT.COM/EN-US/PRODUCTS/AI-SERVICES/OPENAI-SERVICE/
[5] SEE HTTPS://AWS.AMAZON.COM/BEDROCK/

prevailing gaps and challenges in deploying interpretability approaches, together with modern models. This incorporates XAI methodologies that explicitly focus on enhancing the interpretability of a model's output by utilizing interpretable machine learning. This investigation aimed to assess the feasibility of integrating XAI techniques into practical, industry-level applications, such as ChatGPT, by determining the attainability of such an implementation and identifying the associated challenges. The presented prototype specifically combines state of the art Large Language Models, approaches to model interpretability from the field of XAI and a modern user experience/interface (UX/UI) into a usable chatbot application oriented at current market leaders. As preparation to creating this application prototype, extensive testing and evaluation of different existing approaches was done.

The research approach involved leveraging open-source technologies and adapting and extending existing contributions to Large Language Models, XAI, Model Interpretability, and UI for Machine Learning applications. A specific focus was placed on the utilization of open-source software and the most recent technology stack. This was done because the collaborative nature of open-source software fosters innovation and accelerated the development of effective XAI approaches. Focusing on open sources also addresses another key research question: the feasibility of other companies adopting these tools, ensuring broad accessibility, and democratization in the field of AI.

1.2 The Importance of Explainable AI

The importance of Explainable Artificial Intelligence in today's context, particularly with the rise of generative AI and Large Language Models, lies in its ability to demystify the decision-making processes of advanced systems (Molnar 2022; Zhao et al. 2023a). As AI methods become increasingly complex and powerful, understanding their inner workings, and ensuring that they make decisions in a fair, ethical, and transparent manner is crucial (Zini and Awad 2023; Arrieta et al. 2019). Through 2023, generative AI and LLMs have continued to evolve and find applications in various fields (Zhao et al. 2023b). Therefore, the ability to explain and understand their output has become increasingly significant. XAI not only facilitates trust in these systems but also ensures that their integration into different sectors is responsible and aligned with ethical standards. This aligns well with the broader movement towards Responsible AI, where understanding and transparency are key to the successful adoption and ethical deployment of AI technologies (Arrieta et al. 2019). In regulatory contexts, XAI is becoming integral to compliance with emerging standards for "trustworthy AI." As the EU AI Act effectively requires certain AI systems to be transparent and explainable, the Act indirectly supports the development and implementation of XAI, ensuring that AI technology is used responsibly and ethically (Antonini 2023).

Overall, it can be noted that XAI is pivotal in today's AI landscape, particularly with the advancement of generative AI and Large Language Models. This positions XAI not only as an ethical imperative but as a strategic component in responsible AI development and deployment.

2 Fundamentals

The objectives of this thesis are supported by the following foundational elements, which focus on the fields of study addressed by the research question. The following theoretical fundamentals include a concise summary of natural language processing, a description of transformer models and an overview of the emerging category of Large Language Models. Furthermore, the core of this work, Explainable AI and methods of interpretable machine learning, will be further explained, exploring the central concept of Explainable AI and model interpretability, and explaining the fundamental principles and complex details of the selected interpretability approaches. Finally, a brief overview of chatbot applications is provided, along with a concise definition of their associated requirements.

This study can be understood somewhat as part of the field of Natural Language Processing (NLP) which *"[...] investigates the use of computers to process or to understand human (i.e., natural) languages to perform useful tasks."* (Deng and Liu 2018). Natural Language Processing entails the creation of algorithms and systems capable of comprehending, interpreting, processing, and simulating human language in a meaningful and valuable manner (Deng and Liu 2018; Lee 2024). As an interdisciplinary field, NLP draws on linguistics, computer science, and cognitive science to decipher language details (Lee 2024). Machine learning in NLP focuses on enabling computers to analyze, understand, and derive meaning from the human language. This approach facilitates the automatic extraction of linguistic patterns and features, which are crucial for understanding and generating human language (Deng and Liu 2018). It is used in various applications, including automatic summarization, translation, sentiment analysis, and speech recognition (Lee 2024). Machine learning in NLP incorporates various techniques, such as deep learning neural networks, reinforcement learning, and unsupervised learning, to address the complexities and refinements of natural language. These techniques enable the development of sophisticated and accurate NLP applications (Deng and Liu 2018). One of the prime examples of NLP applications is ChatGPT (Lee 2024). Within this specification, NLP covers various specific tasks and applications. These applications include machine translation, information extraction, sentiment analysis, and question answering (Lee 2024). Commonly, language generation is also included in this list of tasks and has recently become one of the most popular applications (Deng and Liu 2018; Amatriain et al. 2023).

Modern AI-based chat applications rely on machine learning algorithms to support language generation tasks, specifically, different forms of language modeling, machine translation, and question answering (Loukides 2023). Text generation is a category of functions that have the goal of automatically producing human-readable text that is coherent, contextually relevant, and grammatically correct. Generation involves creating new text outputs, ranging from simple sentences to complex paragraphs or documents (Deng and Liu 2018). The output is based on specific input cues or data. In language modeling and chatbot applications, these input cues are usually called prompts (Yenduri et al. 2023).

The tasks required to power a chat application, language modelling, can be performed using different machine learning algorithms and approaches (Lee 2024). However, the primary solution for this is transformer-based models. These sophisticated models and their distinct architecture have set new benchmarks in the field of NLP, particularly in handling tasks such as language generation, due to their unique capabilities (Amatriain et al. 2023; Yenduri et al. 2023). These will be further explained hereafter. They can be considered the underlying technology in most recent advantages in AI based NLP, i.e. the introduction of models like GPT-4, LlaMa 2 and Mistral, or groundbreaking applications like ChatGPT and Microsoft Copilot[6] (Lee 2024).

2.1 Transformer Models

Transformer models are the latest architectural advancements in deep learning models, and the critical novel feature of the transformer architecture is the principle of self-attention and its dependency thereon (Yildirim and Asgari-Chenaghlu 2021; Vaswani et al. 2017). Vaswani et al. introduce this as a mechanism that allows each position in a sequence to address all other positions, thereby allowing the model to learn contextual relationships within the sequence (Vaswani et al. 2017). Through self-attention in encoding, decoding, and encoder–decoder attention, transformers can maintain the context between different tokens across all forward feeding steps of a Neural Network (Vaswani et al. 2017). Therefore, they can reliably derive dependencies between the input and output, which is an essential advancement over other deep learning architectures such as primary Recurrent Neural Networks (RNNs) and Long Short-Term Memory RNNs (LSTMs) (Rothman and Gulli 2022).

---

[6] SEE HTTPS://COPILOT.MICROSOFT.COM/

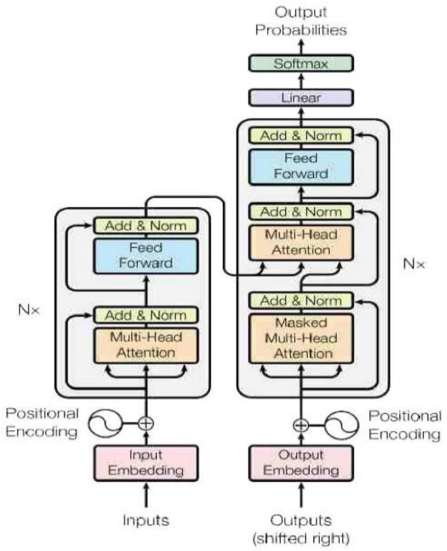


FIGURE 1: VISUAL DEPICTION OF THE TRANSFORMERS ARCHITECTURE (VASWANI ET AL. 2017).

The original transformer architecture is depicted on the left (figure 1). This architecture includes an encoder and decoder block (left-to-right) and visible self-attention heads inside the blocks – “Masked Multi-Head Attention” (Rothman and Gulli 2022). Inside the attention heads the importance of tokens is determined through similarity values, which informs model predictions by evaluating the attention placed on each token (Vaswani et al. 2017; Rothman and Gulli 2022). By combining the self-attention values of the decoded sequence (inside the lower attention head) and referencing the encoder-decoder attention (with the upper attention head), the decoder can reference more information and (global) context in a prediction (Vaswani et al. 2017; Amatriain et al. 2023). Conversely, the encoder in traditional architecture can only access its own self attention.

Any model implementing this architecture, or an improvement thereof, can be considered a transformer model (Amatriain et al. 2023). The architecture, presented in 2017, forms the basis for all further developments, leading to the current state of AI and Large Language Models (Yenduri et al. 2023). While the transformer architecture can also be used for computer vision or audio tasks (Amatriain et al. 2023), the focus will be on language transformers. The word transformer will henceforth be used to reference for these language models based on the transformer architecture.

2.1.1 Model Differences

Transformer language models exhibit three primary architectural distinctions that correlate with the tasks to which they are best suited (Amatriain et al. 2023). The critical architectural variation is related to the attention mechanisms available to the model. Transformers possess three distinct attentions: encoder, decoder, and encoder-decoder attention (Vaswani et al. 2017; Yenduri et al. 2023). Consequently, depending on the architecture and pretraining, the model can be fine-tuned for various tasks (Amatriain et al. 2023).

Encoder-decoder models utilize the complete potential of the initially introduced transformer architecture (Amatriain et al. 2023). They are also known as sequence-to-sequence models. The original architecture allows the encoder to access all input tokens, whereas the decoder can only access tokens preceding the current one and generates output auto-regressively (Yildirim and Asgari-Chenaghlu 2021), which allows the model to generate the output sequence using both the encoder’s output and the current decoder attention (Amatriain et al. 2023). As a result, the model can generate the output text based on the entire given input and existing output and is typically used for tasks such as summarization or text generation based on the input (Amatriain et al. 2023; Yildirim and Asgari-Chenaghlu 2021).

Decoder-only transformer models employ a unidirectional transformer architecture that was initially introduced by the GPT model (Alec Radford et al. 2018). This design replaces encoder-decoder blocks with decoder-only blocks, ensuring that each token solely attends the preceding tokens (Yenduri et al. 2023; Alec Radford et al. 2018). Additionally, these models are called autoregressive models, as they are limited to accessing the previous sequence of tokens, precluding future tokens during training (Yildirim and Asgari-Chenaghlu 2021). By utilizing the pretrained weights of the neural network, these models can generate the following output tokens (Yenduri et al. 2023). Their nature makes them particularly well suited for language generation tasks that fall within the domain of causal language modeling (Amatriain et al. 2023; Yildirim and Asgari-Chenaghlu 2021). Thus, they are widely used in chat applications (Hariri 2023).

Models that utilize only an encoder are commonly referred to as bidirectional transformers (Yenduri et al. 2023), or alternatively, auto-encoding models (Yildirim and Asgari-Chenaghlu 2021). The BERT model, a bidirectional transformer with context-aware embeddings, can maintain attention on tokens before and after a given token, which is made possible through encoder pretraining by masking specific input tokens (Devlin et al. 2018). Bidirectional transformers are particularly well suited for classification tasks and masked language modeling (Yenduri et al. 2023).

2.1.2 Large Language Models

Large Language Models are exclusively based on transformer architecture and possess a model size that can exceed hundreds of millions of parameters (Naveed et al. 2023; Zhao et al. 2023b). However, there is no clearly defined threshold for the model size that translates to categorization as LLM (Zhao et al. 2023b). Large Language Models have been trained using substantial amounts of text data (Zhao et al. 2023b). Because of their exceptional language understanding capabilities and advanced functionality, LLMs are the foundation for contemporary AI-driven chat applications, such as ChatGPT, Claude, Huggingchat, and Microsoft Copilot (Hariri 2023; Naveed et al. 2023).

Large-language models display advanced capabilities compared to smaller models (Zhao et al. 2023b). These capabilities are commonly described as emergent abilities and are mostly displayed because of scaling in the model size and training data (Wei et al. 2022).

One of these emergent abilities is that LLMs are particularly adept at understanding complex linguistic constructs. This allows them to engage in meaningful dialogues and maintain contextual relevance in text generation (ibid.; Naveed et al. 2023; Zhao et al. 2023). Their proficiency in natural language processing can be displayed as a comprehensive understanding of a nuanced language (Wang et al. 2023). Furthermore, LLMs are able to transformer knowledge between fields and learn in context (Zhao et al. 2023b). These are only a few emerged abilities LLMs display; overall, it is important to understand that models of this category can be considered the cutting edge of machine learning capabilities in NLP (Wang et al. 2023; Wei et al. 2022).

Pre-Training and Fine-Tuning

Large language models are usually trained on text and fine-tuned for specific applications (Amatriain et al. 2023). Pre-training and task-specific fine-tuning are the core principles of both transformer models and LLMs (Yildirim and Asgari-Chenaghlu 2021; Rothman and Gulli 2022). Studies on BERT (Devlin et al. 2018) and GPT (Alec Radford et al. 2018) introduced the first application of pre-training and fine-tuning to the transformer architecture (Amatriain et al. 2023).

Although BERT and the original GPT are too small to be considered large language models (Zhao et al. 2023; Wang et al. 2023), they are the base for bigger models and represent two essential architectural differences in LLMs. The theories introduced by these models can be found in most large language models and they were the first foundational models that could be fine-tuned for specific NLP tasks (Amatriain et al. 2023; Yildirim and Asgari-Chenaghlu 2021).

A foundation or foundational model is a basic model trained to handle natural language (Loukides 2023). With the corresponding data, these can be fine-tuned for specific tasks (“downstream tasks”) within the scope of NLP tasks (Rothman and Gulli 2022). These include text classification, text generation, or knowledge-based question answering (Amatriain et al. 2023; Yildirim and Asgari-Chenaghlu 2021). Finetuning can be performed with different approaches. For use in chat applications, the typical approach to fine-tuning is reinforcement learning from human feedback (RLHF), which is used in GPT models and based on reinforcement learning with a reward model that itself is trained on human preferences (Christiano et al. 2017). Other instruction-tuning approaches include instruction tuning on a instruction dataset and tunes the model based on a input prompt and an expected model response (Li et al. 2023; Zhang et al. 2023). Chat versions of LlaMa 2 have been fine-tuned using a combination of RLHF and instruction tuning, the Mistral 7B model meant for chat applications is fine-tuned using instruction tuning. Both models are used throughout this work.

Scaling

The scaling of the model size is an essential factor in the advanced capabilities of current Large Language Models (Wei et al. 2022; Zhao et al. 2023b). The work presented by Google DeepMind (Hoffmann et al. 2022) and previous work by OpenAI (Kaplan et al. 2020) were pivotal in understanding scaling laws, which are guidelines for optimally balancing the model size, training data, and computational resources (ibid.; Zhao et al. 2023). The Chinchilla Model introduced by DeepMind in 2022 demonstrated that with more training data, smaller models could outperform larger models while using less computing power (Hoffmann et al. 2022).

This research has led to two significant trends: developing smaller yet highly efficient models, such as the 7 and 13 billion parameter-sized LlaMa 2 models (Touvron et al. 2023) or the Mistral 7 B model (Jiang et al. 2023). These models require less computational power and are more cost-effective than larger models (Zhao et al. 2023). However, it has motivated the creation of larger models that leverage increased amounts of data, such as the PaLM (Anil et al. 2023). The accessibility and high performance of smaller models are particularly beneficial for the open-source and research community (Naveed et al. 2023). They offer a more practical and economical approach to developing powerful AI tools, leading to widespread adoption and customization (Zhao et al. 2023). Seven billion parameter sized models are also the base for the implementation presented in this thesis.

Architectural Improvements

Most modern LLMs display architectural improvements over the first introduced transformer (Naveed et al. 2023). Key points in research have been the improvement of attention, enlargement of the model's context windows, and further enhancement of language understanding. In addition, research has focused on performance and efficiency (Zhao et al. 2023b).

An example of one of the most modern and improved transformer-based models is the Mistral 7B model. Mistral AI's 7B model combines two advanced approaches to attention: Longformer (Beltagy et al. 2020) and Sparse Transformers (Child et al. 2019) to Sliding Window Attention (Jiang et al. 2023). This increases efficiency and runtime performance (Jiang et al. 2023). Other common improvements in architecture include RMSNorm (Zhang and Sennrich 2019), SwiGLU activation functions (Shazeer 2020) and Rotary Embeddings (Su et al. 2021). All of these are proven to improve language understanding and make Llama2 (Touvron et al. 2023) and Mistral 7B some of the most advanced models (Jiang et al. 2023; Touvron et al. 2023). This fact informed the implementation decision for Mistral 7B as the base model of the implementation, as well as the evaluation of LlaMa 2.

Overview of Selected Models

The following table (table 1) briefly gives an overview of selected models and critical characteristics. It only includes models used in the context of this work and two references: BERT and GPT-4, which are respectively the first transformer-model and the currently leading LLM. Scoring was not included in the comparison, as the different classes of models are only partly comparable in scoring because of their architecture and intended task (Amatriain et al. 2023). All these models are suitable for chat applications, except (the original) BERT model (Yenduri et al. 2023; Amatriain et al. 2023).

Model Name	Created By	Release Year	Availability	Architecture	Model Size (Parameters)	Typical Task
BERT	Google AI Language	2018	open source	bi-directional	340 million	text classification
GPT-2	OpenAI	2019	open source	auto-regressive	1.5 billion	text generation
GODEL	Microsoft, Columbia University	2022	open source	encoder-decoder	770 million	generative question answering
LlaMa 2	Meta GenAI	2023	open source	auto-regressive	7 - 70 billion	text generation
Mistral AI 7B	Mistral AI	2023	open source	auto-regressive	7 billion	text generation
GPT-4	OpenAI	2023	closed source	[unclear]	[unclear]	text generation

TABLE 1: OVERVIEW OF DIFFERENT LARGE LANGUAGE MODELS (DEVLIN ET AL. 2018; JIANG ET AL. 2023; OPENAI 2023; TOUVRON ET AL. 2023; RADFORD ET AL. 2019; PENG ET AL. 2022).

2.2 Explainable AI

Explainable AI refers to the movement towards better understandable Artificial Intelligence and the work contributing to this goal. It is perceived as crucial for building trust in AI systems and overall, the goal of XAI is to make AI systems more transparent, understandable, and ultimately more trustworthy (Masís 2022; Molnar 2022). As first defined by Gunning and Aha in the context of the DARPA's eXplainable AI program: *"[…] [the] goal is to create a suite of new or modified ML techniques that produce explainable models that, when combined with effective explanation techniques, enable end users to understand, appropriately trust, and effectively manage the emerging generation of AI systems."* (Gunning and Aha 2019, p. 45). The fundamental premise of Explainable AI, and specifically Interpretable Machine Learning, is the creation of frameworks to enhance explainability and allow humans to understand model output (Bhattacharya 2022; Molnar 2022).

XAI encompasses a variety of different approaches and principles (Gunning and Aha 2019). While this work focuses on interpretable machine learning, it is vital to understand the overall goals of XAI and ways to achieve this. Various terminologies have been used to describe the goals of XAI (Miller 2017; Arrieta et al. 2019). However, across all these different goals, it is essential to understand that the receiver of explanations is a human (the audience) (Arrieta et al. 2019; Doshi-Velez and Kim 2017). Four terms to describe the goals of XAI are:

Explainability: *"Explainability is the notion of explanation as an interface in the context of AI."* (Arrieta et al. 2019)**.** Explainability is given when an AI system can convey the rationale behind its decisions, actions, or outputs understandably to humans (Arrieta et al. 2019; Kochkach et al. 2023). In XAI, explainability emphasizes the system's decision-making ability and provides insights into the logic and reasoning underlying those decisions (Bhattacharya 2022; Masís 2022). Notably, there is a difference in meaning between explainability and explanation (Molnar 2022), which will be expanded on later.

Transparency: In the context of XAI, transparency refers to the clarity and comprehensibility with which an AI system reveals its internal mechanisms, algorithms, and data processing (Miller 2017; Arrieta et al. 2019). This can be achieved through external measures.

Interpretability: Interpretability refers to the extent to which the causes of a model's decisions are understandable by humans (Kim et al. 2016; Arrieta et al. 2019). It also encompasses the ability of individuals to consistently predict the outcomes of the model, with higher interpretability facilitating greater comprehension of the model's decision-making process (Molnar 2022; Miller 2017).

Understandability: Understandability refers to the ease with which a user can comprehend and make sense of the information and explanations provided by an AI system (Molnar 2022). In the context of XAI, understandability focuses on ensuring that AI's explanations are user-friendly, considering the user's technical knowledge, background, and expertise (Arrieta et al. 2019; Doshi-Velez and Kim 2017).

The core concepts of XAI can be applied across the entire stack of AI applications, ranging from model creation and dataset composition to the display of output and overall accountability (Rothman 2020; Molnar 2022; Bhattacharya 2022). The goal is to reach a state in which AI systems are transparent to human users through understandable, interpretable, and explainable outputs (Arrieta et al. 2019). Additionally, many different approaches, methods, and corresponding technologies can be used to achieve the goals of explainable AI within an AI project or application (Rothman 2020; Masís 2022). This study focuses on enabling XAI on a machine-learning level or as close to a model as possible. It aimed to reach this through what can be described as Interpretable Machine Learning (Molnar 2022; Masís 2022). The connected approaches and methods will be outlined accordingly.

2.2.1 Taxonomy of Explainable AI

While a survey of all significant research is not the goal of this work, it will prove worthwhile to establish some basics on taxonomy and categorization based on the most critical work in the field

(Arrieta et al. 2019; Doshi-Velez and Kim 2017; Molnar 2022). This will inform the following overview of methods and their comparison and ensure a collective understanding throughout this thesis. Establishing a clear categorization is especially important because the Taxonomy of Explainable AI has historically lacked formality, and in literature, specific terms can be used differently (Arrieta et al. 2019; Molnar 2022), possibly further complicating the research on and adaption of XAI

As mentioned, explainable and Interpretable AI are often used interchangeably, and Interpretable AI usually refers to the same goals and ideas as Explainable AI. Therefore, throughout this thesis, Interpretable and Explainable AI are used as synonyms. Different from this are the terms interpretability, interpretability method, and explanation (Molnar 2022; Kochkach et al. 2023). This approach aligns with the common usage of terms in literature (Molnar 2022; Arrieta et al. 2019; Holzinger et al. 2022; Miller 2017). Interpretability in machine learning is : “[…]*the ability to explain or to present in understandable terms to a human […]*” (Doshi-Velez and Kim 2017). An Interpretability Method is an approach rooted in machine learning that aims to make a machine-learning model explainable (Bhattacharya 2022; Masís 2022; Molnar 2022). An explanation is the output of any interpretability method or transparent model to rationalize a model's output/prediction (Molnar 2022; Miller 2017). An explanation typically connects the values of an instance's features with the prediction made by its model in a way that is easy for humans to understand. Alternatively, some explanations comprise a group of data instances (Molnar 2022; Bhattacharya 2022).

Properties of Interpretability Methods

Interpretability methods can display many properties which describe the method's quality (Molnar 2022; Schwalbe and Finzel 2023) or the quality of the generated explanations (Molnar 2022). Both are important for evaluating interpretability methods. These properties are crucial for understanding and evaluating the method's effectiveness and suitability for various scenarios (Schwalbe and Finzel 2023). As a means of simplification, only a subset of commonly used properties outlined in literature (Molnar 2022; Holzinger et al. 2022) are mentioned below and used throughout this work.

Algorithmic Complexity: Algorithmic complexity represents the computational undertaking needed to generate an explanation with the specific interpretability method (Molnar 2022; Schwalbe and Finzel 2023). This is intricately connected to the runtime and potentially can scale with the number of features/tokens and complexity of a model (Holzinger et al. 2022). This is especially true for large language models that can process substantial data and have a corresponding high model size (Madsen et al. 2022).

Fidelity: Fidelity refers to how accurate the explanations generated by an interpretability method are to the actual decision-making process of the model (Molnar 2022). Commonly also referred to as faithfulness (Schwalbe and Finzel 2023)- It assesses the accuracy of the explanation to the model's behavior in specific instances (Schwalbe and Finzel 2023). High Fidelity is one of the most essential properties of an interpretability function, as the explanation closely aligns with the model's actual functioning (Molnar 2022). In a local instance, the explanation accurately reflects why the model made a particular decision for a given data point(Schwalbe and Finzel 2023).

Consistency: Consistency ensures that the explanations remain stable across different instances or versions of the model (Schwalbe and Finzel 2023), as well as across different models trained for the same task with the same data (Molnar 2022). It implies that similar input instances receive similar explanations, maintaining uniformity in the interpretability method's output (Schwalbe and Finzel 2023). This consistency is crucial for the reliability and trustworthiness of the explanations provided by an interpretability method (Molnar 2022).

Stability: Stability, closely related to consistency, refers to the robustness of the explanation against variations in the input (Molnar 2022). It assesses how much the output of an explanation method changes with its input, ensuring that explanations are dependable and not overly sensitive to insignificant changes in the data. Stable explanations are essential for maintaining the validity and reliability of the interpretability method over different scenarios and data variations.

The properties, combined with the categorization of methods are the most coherent way to determine a methods quality and fitting to certain model and application settings. Accordingly, they have been used to determine the most fitting approach to model interpretability.

Categorization of Interpretability Methods

In Explainable AI, there are typically two different ways to reach interpretability in machine learning, however some sources suggest a more detailed categorization (Schwalbe and Finzel 2023), showcasing the diverse understanding of the taxonomy of XAI. Similar to the differentiation outlier earlier, a more straightforward categorization for Taxonomy was chosen, rooted in different surveys(Arrieta et al. 2019; Holzinger et al. 2022).

Intrinsic Interpretability or **Transparent Models:** Interpretability is intrinsic to the model, when the model is interpretable by design (Molnar 2022) or self-explainable (Schwalbe and Finzel 2023). Transparent Models are also named white-box or glass-box models (Linardatos et al. 2020). A model with intrinsic interpretability however might not be understandable and still call for an interpretability approach to reach the goals of XAI (Molnar 2022). Making a model transparent through adapting the architecture can also be an approach to interpretability but is usually considered a post hoc approach (explained below) (Schwalbe and Finzel 2023).

Post Hoc Interpretability: Opposite to transparent models, non-transparent models are called black box models (Arrieta et al. 2019; Molnar 2022). The existence of black box models calls for methods to explain predictions of an AI after they have already been made. These are Post Hoc Interpretability methods (Arrieta et al. 2019; Holzinger et al. 2022; Madsen et al. 2022). Post Hoc Interpretability methods are meant to explain a model prediction decision *after the fact*, considering the input and output of a model, and aim to clarify the model's decision-making process by generating explanations for individual predictions or highlighting the model's overall behavior (Molnar 2022). Generated explanations can have various forms and showcase multiple properties (Schwalbe and Finzel 2023).

Black box models include transformer models, which are per definition not transparent because of their inherent nature as deep learning models (Zini and Awad 2023). As this work focuses on the interpretability of transformer architecture-based models, the focus will be on Post Hoc Interpretability going further. Therefore, they are exclusively referenced in the following and references to in short as interpretability method or interpretability approach henceforth.

Various approaches exist to further categorize Post Hoc Interpretability Methods. Fundamentally, a method for Post Hoc Interpretability is defined by at least three characteristics (Arrieta et al. 2019; Molnar 2022; Holzinger et al. 2022):

1. **Model Specific** or **Model Agnostic (Portability)**: This characteristic determines whether an explanation method is tailored to a specific model, or models with the same architectural elements, or is versatile enough to be applied to various models (Molnar 2022; Schwalbe and Finzel 2023). Model-specific methods are designed to collaborate with a particular model type, unlocking detailed insights specific to that model's structure (Molnar 2022). In contrast, model-agnostic methods offer broader applicability, allowing for explanations across different model types and architectures, and enhancing their utility in diverse applications (Arrieta et al. 2019)
2. **Local** or **Global Explanation (Locality)**: This classification addresses the scope of the explanation that is provided by an interpretability method (Arrieta et al. 2019; Molnar 2022). Local explanations focus on individual predictions, shedding light on why a model made a specific decision for a single instance. This is useful for understanding case-specific model behavior (Holzinger et al. 2022). Global explanations, on the other hand, provide insights into

the overall decision-making process of the model, offering a comprehensive view of how the model operates across all instances (Molnar 2022).

3. **Explanation Outcome (Expressive Form)**: This category refers to the format in which the explanation method presents its findings (Molnar 2022). The output form of an explanation varies widely, ranging from numerical values like attribution values, rules and heatmaps to example-based explanations. The nature of the output determines how accessible and understandable the explanation is for different audiences, playing a critical role in the practical usability of the explanation method in real-world scenarios (Schwalbe and Finzel 2023).

Considering the outlined categorization, Interpretability Methods can be classified as follows (figure 2). In this graphic, the post hoc interpretability approaches would be listed below their specific expressive form.

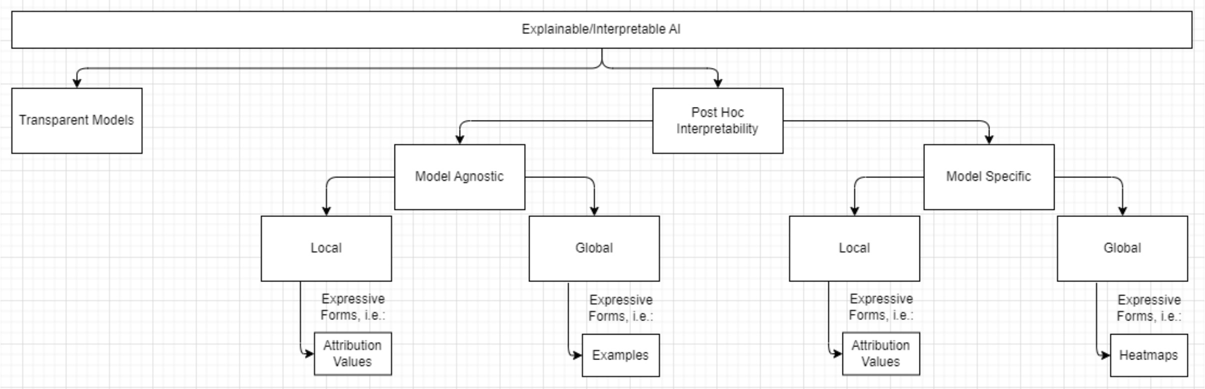


FIGURE 2: GRAPHIC SHOWCASING TAXONOMY OF XAI APPROACHES.

Furthermore, there are two primary approaches to model interpretability: gradient-based methods and perturbation-based methods. The former is based on altering input features and assessing the network's performance on each possible (random) combination of input features - perturbations, which can be computationally intensive and requires multiple passes through the model. In contrast, gradient based approaches use backpropagation to calculate gradients and estimate attribution scores (Ancona et al. 2019). Based on this fact, gradient-based approaches are model specific to neural network-based models, like transformer models.

2.2.2 Overview of Interpretability Methods

In the following, we will look at different techniques for model interpretability, specifically Post Hoc Interpretability methods fit for transformer models, which limits the model specific methods available. Still the overview is not intended to be complete, as the number of different methods and further improvements is enormous (Kochkach et al. 2023; Arrieta et al. 2019). The overview will include different model-specific and model-agnostic methods. Every presented method can be applied to transformer models.

Method Name	Portability	Locality	Expression Form	Key Properties	Description
LIME	Agnostic	Local	Attribution Values	Local Fidelity	Interpretability through the local fitting of surrogate models
Shapley Value (Sampling)	Agnostic	Local	Attribution Values	Fidelity and Consistency	Game-theory based attribution method through perturbation.
KernelSHAP	Agnostic	Local	Attribution Values	Local Fidelity and Efficiency	Combination of LIME and Shapley Values to estimate SHAP Values.
PartitionSHAP	Agnostic	Local	Attribution Values	High Efficiency	Runtime efficient variant of SHAP based on Owen Values.
Attention Visualization	Specific	Local/Global	Visualization	[unclear]	Visualization of underlying attention of the model to different tokens.
Attention Attribution	Specific	Local/Global	Attribution Values	[unclear]	Attribution of attention based on attention heads and dependencies.
GradientSHAP	Specific	Local	Attribution Values	Fidelity	SHAP variant based on integrated gradients.

TABLE 2: OVERVIEW OF DIFFERENT INTERPRETABILITY APPROACHES (MOLNAR 2022; HOLZINGER ET AL. 2022; LINARDATOS ET AL. 2020).

This table (table 2) gives a brief overview of the following interpretability approaches and their characteristics. The table additionally includes a concise description of the model interpretability approach, their properties, and how explanations are created. More in-depth explanations will follow.

Model Agnostic Methods

The concept of making interpretability and explanations independent from a model was first introduced in 2009 (Baehrens et al. 2009). While model-agnostic approaches are the most flexible, they can also lack performance compared to model-specific approaches (Molnar 2022). They are typically more widespread in adoption because of their versatility and ease of application across diverse types of models (Molnar 2022; Holzinger et al. 2022).

Some of the more common Model Agnostic Interpretability Approaches are LIME (Ribeiro et al. 2016a), the SHAP framework (Lundberg and Lee 2017) and Shapley Values, estimated through i.e. Sampling (Erik Štrumbelj and I. Kononenko 2010) .

LIME (Local Interpretable Model-agnostic Explanations) is an approach that explains local decisions with a surrogate model (Holzinger et al. 2022, p. 16) The overall goal of LIME is to use an interpretable model to generate explanations. This model is trained on perturbations of input and output of the to-be-explained model (Ribeiro et al. 2016a). This is achieved through local fitting of an interpretable surrogate model (i.e., a linear regression model), which is how explanations are locally generated for each feature (Ribeiro et al. 2016b). LIME is a local perturbation-based approach that provides explanations for each model decision and no global explanations (Holzinger et al. 2022). Furthermore, LIME can be used for Text, Tabular, and Image data (Molnar 2022). LIME local fidelity is remarkably high. However, it is highly dependent on the fitting of the surrogate model, which can require more effort and data than other approaches (Holzinger et al. 2022). LIME explanations can be unstable, especially if LIME is not optimized (Molnar 2022).

Game Theoretic Approaches use game theory as a theoretical approach to understanding social interactions with competing actors. It studies optimal decision-making by independent and competing agents in a strategic setting. Cooperative game theory, a branch of game theory, emphasizes that coalitions of players are the primary decision-making units that can enforce cooperative behavior. Shapley values, named in honor of Lloyd Shapley, who introduced them in 1951, are a solution concept in cooperative game theory. In this context, the Shapley value acts as a formula to measure each player's contribution to the overall game. It addresses the challenge of fairly distributing the "payout" among players (or features in machine learning), considering their varying contribution levels and influence. The Shapley value is calculated based on the marginal contribution of a variable or feature across all subsets of features, aiming to distribute the total gains among the players.

This theory can be utilized for model interpretability; in this context, Shapley Values are used to determine the relative impact of each feature or variable on the output of a model by comparing their effects against the average across all different outcomes. Each model feature acts like a player, and Shapley Values are used to calculate these features' impact on the model output. These are the generated explanations. There is a large number of game theory-based approaches (including SHAP) that are all based on some form of Shapley Values (Linardatos et al. 2020).

The main advantages of Shapley's values are the fair distribution between features, fidelity, contrastive explanations, and the solid theory behind the approach (Molnar 2022). The disadvantages of Shapley Values are most notably the algorithmic complexity and their complex interpretation (Holzinger et al. 2022). One common criticism of Shapley Values and all methods based on perturbation is the random combination of features. As different perturbations are created and examined, unrealistic and potentially wrong feature combinations can appear. This is based on the theory that each player (or feature) is independent. However, this is commonly not the case, especially in the field of NLP. This leads to sampling and evaluation of entirely wrong feature combinations.

Shapley Values are extremely exhaustive since the number of calculations exponentially increases for each additional model feature. As expressed mathematically, this is 2^{F}, with F being the amount of model features, which in a LLM setting are the input tokens. Considering this, it is typically only feasible to estimate this value, as accurately calculating them is too computationally intense. One approach to estimation is Shapley Value Sampling, which samples permutations of the input features to determine the values (Erik Štrumbelj and I. Kononenko 2010).

An evolution of Shapley Values is Owen Values. This concept extends Shapley Values by accounting for the presence of a coalition structure. This structure divides the players into distinct groups or coalitions, and the Owen value determines the value or payoff for each player, considering their contribution and membership in specific coalitions. This makes the Owen Values particularly applicable to scenarios where players (or model features) are naturally grouped or have existing partnerships or alliances. Owen values are then exclusively calculated based on these grouped features instead of all individual features, which significantly limits the number of needed calculations as they do not increase exponentially.

The SHAP Framework (SHapley Additive exPlanations) was introduced in 2017 by Lundberg and Lee (Lundberg and Lee 2017). The framework presents a form of attribution values (SHAP values) and new associated methods to calculate these. These methods are model agnostic and model specific approaches, which each combines the benefits of Shapley Values with the benefits of each underlying interpretability method (Lundberg and Lee 2017; Edoardo Mosca et al. 2022). For this, the framework relies on the work of essential approaches, namely LIME, DeepLIFT, and LayerWise Relevance Propagation (LRP) (Linardatos et al. 2020).

SHAP values allow the attribution of each feature’s importance based on the expected output of the original model (the base values) (Lundberg and Lee 2017). SHAP values have several advantages over the feature significance measurements of other methods (Molnar 2023). They have local accuracy, missingness, and consistency (Linardatos et al. 2020).

Through SHAP values the authors suggested to unify different interpretability approaches into one consolidated metric for assessing feature significance, which the named methods all aim to estimate (Lundberg and Lee 2017). The authors also suggested several ways to highlight the SHAP values graphically; an example from the original paper can be seen below (figure 3).

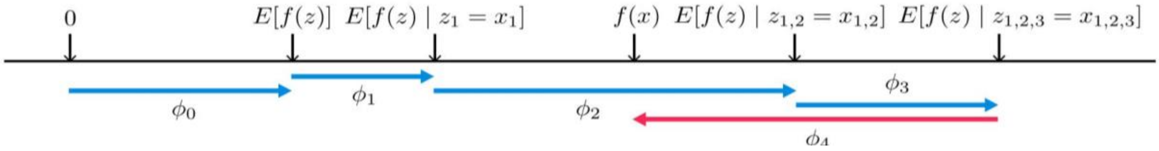


FIGURE 3: VISUAL DEPICTION OF SHAP VALUES (LUNDBERG AND LEE 2017)

Lundberg and Lee introduced new methods to approximate SHAP values based on existing Interpretability Methods. Generally speaking SHAP methods all have similar advantages, which are the same as the advantages of Shapley Values (Holzinger et al. 2022). Additionally, they display a more efficient estimation through different methods and inherit some benefits of the approach they are adopted from (Edoardo Mosca et al. 2022; Lundberg and Lee 2017). Across all these methods, the SHAP Values are also comparable (Molnar 2023). KernelSHAP and PartitionSHAP will be explained thoroughly in the following as they were used throughout the implementation part of this thesis.

A core principle common to most SHAP interpretability approaches is masking, a method used to simulate all the different possible outcomes or subsets of features and used in KernelSHAP and PartitionSHAP. All different subsets (perturbations) of features are simulated by masking different input features. With text input, this is commonly done by replacing the value with an empty string. For each subset, attribution values can be calculated using different interpretability methods. The individual impact of each feature on the output can be calculated by comparing these values to a base value across all different feature variations. The masking approach was further formalized by Scott Lundberg and Su-In Lee, together with Ian C. Cover, in their paper “Explaining by Removing: A Unified Framework for Model Explanation” (Covert et al. 2020).

KernelSHAP is a model agnostic interpretability approach that combines Shapley Values and LIME (Molnar 2022). It uses improved principles of LIME to estimate SHAP values. Different model features are masked based on an initially calculated attribution, and model predictions are generated for each created perturbation of features (input tokens). A single perturbation is sometimes called a coalition and is a combination of different model features out of which one or more can be masked out to focus on the non-masked-out feature. For masking, KernelSHAP replaces a feature value with a randomly sampled value from the training dataset. In, i.e., an NLP setting, this can also be an empty string value. Based on predictions for different perturbations, a group of weights is calculated using the SHAP

Kernel. This SHAP Kernel is a formula used to weigh different coalitions according to their importance, and, in combination with LIME, it can generate Shapley Values. This LIME is fitted with a weighted Linear Regression based on the calculations done through the SHAP Kernel. LIME then generates the feature importance values, SHAP Values, for all input tokens.

PartitionSHAP is a newly introduced SHAP variant based on Owen Values. For this, PartitionSHAP clusters input features. While this could be done based on, i.e., grammatical structure, it is usually done based on a similar pattern. For these unions and perturbations, PartitionSHAP calculates Shapley Values, utilizing masking throughout to create different feature subsets. The given PartitionSHAP implementation replaces string tokens with an empty string or the following: "…." Different masking strategies are employed for other models and data types, as specific models cannot manage missing features. Because of the Owen values approach to game theory, it is computationally less intensive to calculate SHAP Values. This makes PartitionSHAP highly runtime efficient and a fast model agnostic approach. However, Owen's values are more granular, and PartitionSHAP explanations can tend to be less accurate than methods like KernelSHAP. Otherwise, it still has the benefits of SHAP Values and is a valuable addition to model agnostic interpretability approaches.

Model Specific Methods

Model-specific interpretability methods are designed to uncover the internal mechanics of a model, offering explanations for its outputs in a way closely tied to the model's architecture. They typically leverage machine learning models' specific concepts and internal workings. In the realm of transformer model interpretability, various methods are employed.

Attention Visualization is a method where the attention values between tokens in self-attention layers are considered a relevancy score. For this, it utilizes the potential inherent interpretability of transformer models by outputting value representation of the transformer models attention to specific tokens (Yeh et al. 2023; Vig 2019). Specifically, these are the self-attention values created by a transformer model, representing the relationship between different words (Vaswani et al. 2017). The attention values allow a transformer to make informed predictions based on input tokens and (potentially) previously decoded tokens (Amatriain et al. 2023; Vaswani et al. 2017). Attention visualization can be done on encoder, decoder and cross (encoder-decoder) attention, depending on transformer model specifics (Vig 2019). However, presenting model features is not necessarily understandable to the audience. To reach interpretability, further processing and visualization are needed (Molnar 2022). Attention visualization can deliver straightforward, achievable explanations that have an accessible expression. However, they lack actual insights, as there is no actual connection possible between input and output values, especially with auto-regressive generative models (Zhao et al. 2023a). Furthermore, a model's attention values do not accurately represent a token's importance to a model output (Hao et al. 2020; Zhao et al. 2023a). Therefore, attention visualization is often critiqued for its simplicity and potential to highlight irrelevant tokens. It tends to focus solely on attention scores and ignores other computational aspects of the network (Hao et al. 2020). Subsequently, other connected approaches have been introduced, such as attention attribution, which seeks to use attention values as the base of attribution. It does so through refining the selection of attention values, based on different attention heads and extracting dependencies.

The two most common attention visualization implementations are BERTViz (Vig 2019) and the more recent AttentionViz (Yeh et al. 2023). These build on previous research on attention visualization, most importantly Tensor2Tensor visualization (Vig 2019).BERTViz[7] introduced improvements to previous work, notably adopting decoder-only and encoder-decoder Models (Vig 2019).

2.2.3 Evaluating Interpretability Methods

Interpretability methods can be evaluated through several measures, one of which is through the properties of Interpretability Methods and the generated explanations (Molnar 2022; Schwalbe and Finzel 2023). The properties combine qualitative and quantitative measures, as well as some properties that are not necessarily suitable for evaluation, like expressive form. Selected properties have been outlined in Properties of Interpretability Methods, but many others exist (Schwalbe and Finzel 2023). These properties can accurately describe the quality of an Interpretability Method and generated explanations, and there is existing work on evaluating different approaches on fidelity, consistency, or other properties. Nonetheless, evaluating the properties and objective helpfulness of an interpretability method can prove challenging. Some recent contributions have introduced ways to evaluate explanations with reference datasets. These contributions (Hedström et al. 2023; Arras et al. 2022), as of December 2023, have so far focused on Visual Models or, in some cases (Hedström et al. 2023), not progressed to NLP tasks.

However, evaluation of Interpretability methods through properties or with testing datasets does not necessarily cover how valuable explanations are for a human user, which can be considered the primary audience of explanations and XAI (Arrieta et al. 2019). To evaluate methods considering the needs of this audience, a measure of accuracy of consistency across different cases is typically not enough, as high fidelity or strong consistency do not necessarily reflect on the value to the explanation audience. Doshi-Velez and Kim have suggested a framework that allows for the evaluation of

---

[7] SEE HTTPS://GITHUB.COM/JESSEVIG/BERTVIZ

interpretability method implementations based on numerous factors and their target audience. It thoroughly outlines evaluation steps for any implementation. This framework was used as the underlying framework to evaluate the quality of the created prototype in this work. The framework fundamentals consist of three evaluation steps: application-, human-, and functionally- grounded evaluation of interpretability methods. In this context, application-grounded evaluation is the evaluation in the context of the real-world application. This means, i.e., that an experiment is run with the exact intended usage, and real-world users of a system evaluate the result (Doshi-Velez and Kim 2017). The human-grounded evaluation covers a simplification of the application-grounded evaluation. The evaluation is done by simply grading an explanation performed outside of the usual application context (Doshi-Velez and Kim 2017). Functional evaluation is done through evaluation by proxy; by comparing an explanation to another one and excluding any human rating (Doshi-Velez and Kim 2017). While the first two steps evaluate subjective helpfulness for different audiences, the last evaluation step is on a more technical level and focuses on properties of the used interpretability method.

2.3 Chatbots

The chosen sample use case to further evaluate interpretability options for LLMs is that of a chatbot application. The most prominent example of a chatbot application is ChatGPT, which was released in November 2022 (Loukides 2023; Hariri 2023). A chatbot is a computer program designed to simulate conversation with human users, especially over the internet. It operates through a combination of pre-set scripts and various forms of AI, utilizing technologies such as natural language processing (NLP) and machine learning (ML) to process and understand human language (Adamopoulou and Moussiades 2020). Since, 2022, chatbots have developed to incorporate AI and have become reliant on large language models (Loukides 2023; Hariri 2023)

Chatbot applications display a few key characteristics. Firstly, they are designed to understand and respond to natural language inputs, allowing for more intuitive user interaction (Galitsky 2019; Adamopoulou and Moussiades 2020). These applications also prioritize user engagement, often employing conversational tactics to maintain a natural and fluid dialogue (Jain et al. 2018). Connected to these three main user expectations can be defined as the core expectations set towards chatbot applications:

User Experience/Interface: The design of the chatbot user interface (UI) is pivotal in shaping the user experience. Interfaces that are intuitive and straightforward are preferred, facilitating interactions that are uncomplicated and demand minimal learning or effort from users

(Jain et al. 2018; Galitsky 2019) . The aim is to develop user-friendly platforms that enable effortless navigation and interaction, thereby making chatbots accessible to a wide spectrum of users with varied levels of technical proficiency (Jain et al. 2018).

Language Understanding: A fundamental expectation for modern chatbots is the capability to process and understand natural language (Abdulla et al. 2022). This necessitates the integration of advanced Natural Language Processing technologies like Large Language Models, which empower chatbots to interpret and respond to the colloquial and diverse language utilized by different users (Abdulla et al. 2022; Hariri 2023).

Personalization: Personalization in chatbot interactions is important to users and a key premises of commonly available tools (Hariri 2023). Customized chatbots that recall previous interactions and accordingly customize responses provide a more individualized experience (Abdulla et al. 2022)

3 Current State of Research

An ample number of theories, surveys on taxonomy and modern approaches to machine learning interpretability exist (Arrieta et al. 2019; Holzinger et al. 2022; Molnar 2022). This includes work applicable and specific to transformer models, as outlined throughout the fundamentals part of this work. A good amount of groundbreaking work can be found around the theories and methods of XAI (Lundberg and Lee 2017; Ribeiro et al. 2016b), as well as applications thereof (Palacio et al. 2021; Madsen et al. 2022; Masís 2022). Nonetheless, a notable gap can be observed when researching the adoption onto Large Language Models within the established definition, or when looking into implementation with chat applications. A similar gap, however smaller, exists in XAI research utilizing generative text models. Several interpretability approaches, both model specific and model agnostic, are applicable to transformer models and generative text models. Nevertheless, most work or demonstrations of new interpretability approaches are utilizing models like BERT or GPT-2. These cannot or barely be considered Large Language Models, because of their model size and lack of emergent capabilities (Zhao et al. 2023b). BERT is a commonly used and utilized language model for research purposes within other XAI methods and implementations of SHAP (Mosca et al. 2022). This is probably mainly because of its wide adoption and low computational requirements. Other work references GPT-2 as showcase for text generation models (Molnar 2023). However, modern generative models are not commonly used in these circumstances. A commonly used NLP task to display XAI methods is hate speech detection or other classification tasks like sentiment detection (Remmer 2022; Mosca et al. 2022). However, implementations of model interpretability on text generations are typically hard to find, and limited significant research exists. This is also the case because of the relative novelty of widely and openly available LLMs like LlaMa, LlaMa 2, Mistral AI, OpenChat, etc. (Zhao et al. 2023b).

An outlier to this lack of research is recent work from the team behind captum. Captum is an open-source library for model interpretability, build for Pytorch based models and part of Meta Open Source (Kokhlikyan et al. 2020). A recent paper introduces novel features for LLM interpretability and showcases application into a model like LlaMa 2 (Miglani et al. 2023). This work is referenced in the presented implementation.

Overall, a clear research gap in the field of XAI focusing on the application of interpretability of generative text models is evident. This is especially the case for comparing how well different approaches fit LLMs and if existing approaches can fulfill the expectations for Explainable AI in a production setting. Specifically, a usable chatbot application prototype, which is the topic of this thesis.

4 Implementation

The implementation is split into two main parts that build on each other. First is the exploration and evaluation of different interpretability approaches on Large Language Models (LLM Interpretability). Building on this is the conception, design, and implementation of an application prototype (Application Prototype). The first is fundamental to the second as it thoroughly explores several concepts and implementations and lays the groundwork for the application prototype. The exploration and evaluation produced conclusive qualitative and quantitative insights on applicability of different methods that were used for implementation and individually analyzed. The prototype created based on these insights was evaluated using a specifically created evaluation framework and individual qualitative assessment. The results of this evaluation provide further insights into the possibilities of interpretability for LLMs and withing chatbot applications.

In the following, an overview over specific used technologies, technological approaches and methods will be given. This includes the used evaluation framework. Furthermore, notable implementation details of both implementation steps will be outlined separately.

4.1 Used Technologies and Methods

A customary technology and tool stack was used for implementation. This included Python, Pytorch, the Huggingface Platform,[8] and Google Colab[9]. All of these are standard technologies/tools for their respective tasks. They are commonly included in literature covering the application of LLMs for both scientific and industry usage (Jha 2023; Masís 2022; Rothman 2020; Rothman and Gulli 2022; Tunstall et al. 2022). Other common standards include using Docker, GitHub, and PyCharm as IDEA for programming. A focus was put on utilizing open-source technologies, for reasons outlined earlier. Furthermore, using open-source software allows for easy access and, if needed, customization of software without having to develop an entirely new system.

4.1.1 Notable Used Technologies

Additionally, this work is based on four open-source libraries for python that are less commonly used within machine learning and a UI toolkit for machine learning applications. Most importantly these are the used libraries for interpretability implementation: shap[10], captum[11] and inseq[12]. shap and captum are the two most widely used open-source libraries for interpretable machine learning. Furthermore, this work explored the insights provided by attention visualizing using another library, BERTViz. Moreover, the prototype is based on Gradio[13] for user interface functions. All these used technologies and tools will be shortly described in the following.

shap

The shap package originates from the foundational work and code of Lundberg and Lee, which was introduced alongside the SHAP framework (Lundberg and Lee 2017). Over time, it has been enhanced by contributions from various developers in the open-source community and follow-up work by Scott Lundberg. As of December 2023, the package supports eleven methodologies for computing SHAP-based explanations. It offers user-friendly classes tailored for diverse interpretability methods, which facilitate the generation of explanations and the visualization of feature attribution. It implements model specific methods like DeepSHAP and GradientSHAP, as well as model agnostic methods like KernelSHAP and PartitionSHAP. Depending on the method, it supports input in textual, tabular, and visual form. The package includes various visualization capabilities to create interactive graphics in

---

[8] SEE HTTPS://HUGGINGFACE.CO/

[9] SEE HTTPS://COLAB.GOOGLE/

[10] SEE HTTPS://SHAP.READTHEDOCS.IO/EN/LATEST/

[11] SEE HTTPS://CAPTUM.AI

[12] SEE HTTPS://INSEQ.ORG/

[13] SEE HTTPS://WWW.GRADIO.APP/

html and JavaScript as well as graphics. These features have mostly been contributed by developers such as Scott Lundberg, as is the case with most of the package features.

SHAP is one of the more complete interpretability frameworks available, especially because of the array of plotting and visualization tools available. However, the shap package has not seen significant advancements in recent years, with the latest updates primarily focusing on bug fixes. No substantial updates have been implemented to adapt the package to the rapidly evolving landscape of Large Language Models since the acceleration in their development. Nonetheless, shap does support a limited amount of auto regressive models like GPT-2, which are wrapped in special classes to gain access to prediction probabilities. The shap package was used in exploration of different methods, evaluation of LLM Interpretability and as a supplementary implementation of model interpretability in the application prototype. The prototype also uses the advanced visualization provided by shap.

captum and inseq

The captum library is a library by Meta Open Source, specifically the same team working on PyTorch (Kokhlikyan et al. 2020). Within this ecosystem, Captum offers a suite of interpretability methods that are integrated with PyTorch. Based on the possibilities of PyTorch models, captum also supports diverse input types like text and visual or tabular data. However, the library is dependent on PyTorch and therefore only supports neural network models. The library supports sixteen post hoc interpretability methods and, most significantly, a framework for LLM interpretability that was released in December 2023 (Miglani et al. 2023). Examples of available interpretability methods include LIME, KernelSHAP, GradientSHAP and Shapley Value Sampling. The newly introduced LLM Attribution classes simplify the incorporation of various interpretability techniques into autoregressive text generation models, such as Llama 2 (Miglani et al. 2023), which is achieved through directly accessing model functions and effective masking of input features, like the shap package. Captum additionally offers a limited visualization capability. Captum was extensively used for exploration and evaluation alongside the shap package. Furthermore, it provides the main interpretability method implementation in the application prototype.

Inseq is another initiative that aims to provide access to multiple interpretability methods, with a specific focus on sequence-to-sequence models (Sarti et al. 2023). It relies heavily on captum and augments it with additional visualizations, as well as support for certain models. Based on this it was evaluated for the usage with GODEL.

Overall, captum provides an extensive set of available interpretability approaches, profiting from a tight integration with PyTorch and backing of the respective team. For PyTorch based neural network models it can be considered the prime solution to post hoc interpretability. This is exemplified by the newer approach to interpretability for Large Language Models, which to date is unique in the field. Inseq provides various capabilities specific to sequence-to-sequence models that would not be available otherwise and supports these with improved visualization based on captum. However, its limitations with auto-regressive models mean it is not fit for most other models.

BERTViz

BERTViz, first introduced in 2017, is an open-source library providing graphical visualizations of transformer model attention in HTML and JavaScript (Vig 2019). It uses model functions to output attention values together with the model output and visualizes these. Three different views of model attention can be created, depending on the used model. Since the introduction of GPT-2 in 2019 there has been no significant improvement of BERTViz and overall, the value as interpretability method is severely limited, since attention values do not accurately represent the importance of tokens to the model output. It was therefore not used as part of the implementation or evaluated further.

Gradio

Gradio combines Python and JavaScript packages to a complete library focused on the creation of user interfaces for machine learning applications (Abid et al. 2019). It enables building UI in pure python, out of which it compiles HTML, JavaScript and CSS and other underlying technologies for routing, etc. For this, Gradio predefines various components and other functionalities tailored towards machine learning applications and additionally enables the creation of custom components (Abid et al. 2019). Gradio's design allows for minimal transitions between programming languages and effortless integration with hosting services such as Hugging Face Spaces. It is one of the more widely used UI libraries along with i.e., Streamlit for Python or Shiny for R.

4.1.2 Code Quality

Black[14] and Pylint[15] were utilized as code quality tools to ensure high code quality across Python code in the application prototype modules and Jupyter Notebooks. black is a code formatter for Python that utilizes an opinionated code style based on the PEP 8 Python style guide. black detects and automatically fixes typical errors and inconsistencies in code formatting. Pylint is a static code analyzer focused on errors, code smells, and possibilities for refactoring. Both tools were run in a CI/CD scheme using the pre-commit[16] package and GitHub Actions. Combining black and Pylint ensures code quality based on standard Python style guides like PEP8. Several minor rule changes, and ignored issues were

---

[14] SEE HTTPS://BLACK.READTHEDOCS.IO/

[15] SEE HTTPS://PYLINT.READTHEDOCS.IO/

[16] SEE HTTPS://PRE-COMMIT.COM/

additionally configured. However, these custom rules are conservative, and a high code quality is maintained at an elevated level.

4.1.3 Evaluation Framework

The evaluation framework used to investigate the quality of the implementation is loosely oriented on the framework proposed by (Doshi-Velez and Kim 2017), as outlined and described in Evaluating Interpretability Methods. In this work, two evaluation steps have been covered: application-grounded and human-grounded evaluation. Additionally, a comparative analysis was performed using the (also implemented) visualization approach. This allowed for isolated evaluation of the quality of explanations and the head-to-head comparison to a different method. Attention Visualization can be considered a baseline of interpretability for transformer models as it is the simplest, most easily usable, out-of-the-box approach. The limited value of generated explanations is recognized in the comparative analysis through the scoring method. This way, a rich set of evaluation results covering several factors of good XAI could be compiled, and the method was evaluated for fidelity, consistency, and stability. Furthermore, the helpfulness of explanations to different audiences was measured. This way, the evaluation adheres to all the critical premises of XAI. A scientifically sound approach was ensured by relying on fundamental work on the topic (Doshi-Velez and Kim 2017), referenced by some practical sources in the field (Molnar 2022). These three tiers of evaluation will be outlined shortly. While functionally grounded evaluation should be considered an important set of evaluation and in certain cases might be the only viable evaluation method (Doshi-Velez and Kim 2017), it was not used as evaluation step in this work. The reason for this is the focus on the helpfulness to the audience and the thoroughness of the previous evaluation before building the application prototype. The latter already represents thorough testing of the interpretability method implementation, as the specifics are similar in evaluation and the final prototype.

For evaluation of the prototype, a total of ten test inputs were used with the main model and SHAP based interpretability approach. The ten test inputs vary in topic, message, and system prompt to account for variety in inputs. For comparability, two inputs always cover the same topics, although with different messages or system prompts.

Application-Grounded Evaluation

This evaluation tier was fulfilled through rating of the explanations by the application creator. In this scenario, he fulfills the application or model creator role. This persona's stated goal is to further understand or enhance application and model. Each prompt and the resulting explanation were rated for helpfulness towards reaching this goal on a scale from 1 to 5. This evaluation resulted in an application-grounded helpfulness score. A higher score on this scale is better.

Human-Grounded Evaluation

Human-grounded evaluation was performed similarly to application-grounded evaluation but with a different audience in mind. The rating was done from the perspective of a human without any prior knowledge of the model or specific knowledge of the field of NLP and machine learning. Essentially, the rating was still performed by the author but by thinking of the position of the described persona. The goal of this persona was to gain a better understanding of the model's reasoning. For this, explanations should show a clear connection between input and output and can be easily understood. Using this approach, the implementation could be evaluated from an end-user perspective without leveraging extensive user interview, which was deemed out of scope for this work. The human-grounded evaluation step resulted in the human-grounded helpfulness score. As with the application grounded evaluation, this is represented on a scale of 1 to 5. Respectively, a higher score is better.

Comparative Analysis

The application-grounded and human-grounded evaluations were also performed on the implemented attention visualization to inform a comparative analysis. Evaluations were done in the ways described for application- and human-grounded evaluations. The delta between the application-grounded and human-grounded evaluations was calculated to formalize the comparison between methods. The delta was calculated by subtracting the rating for attention visualization from the rating of the main interpretability method. The reason for this is that it is to be expected and desired that the SHAP implementation performs better. Therefore, a positive delta is anticipated. A higher delta value is better.

4.2 LLM Interpretability

In preparation for the conception, design, and development of the application prototype, research to evaluate various approaches to post hoc interpretability was conducted. This involved exploring several different implementations, using the previously described libraries for model interpretability as a starting point. Furthermore, selected approaches and models were evaluated together on quality and performance using a standardized approach. The insights gathered from the exploration of different methods were combined with the evaluation. These generated results are the basis for implementation decision for the prototype, but the results also represent significant insights into the interpretability of Large Language Models.

4.2.1 Exploration

The application of various interpretability methods, libraries, and models was investigated thoroughly through practical testing. The methods explored included LIME, KernelSHAP, PartitionSHAP, PermutationSHAP, GradientSHAP, Integrated Gradients, Shapley Value Sampling, and various forms of attention visualization. The objective was to determine the efficacy of the approaches with Large Language Models like GPT-2, Mistral, and Llama 2. The explored implementations are the basis for a formalized comparison of different methods on various Large Language Models.

Through exploration of PartitionSHAP with Mistral and Llama 2 it was determined that the PartitionSHAP implementation from the shap library was not functional because of an issue in calculating log odds, which is the probability to predict each given token and is needed when calculating SHAP values for the different perturbations of the model input. This issue could not be fixed because of the theoretical and technical complexity of PartitionSHAP. Nevertheless, it was possible to implement three other methods with these larger models. These are LIME, KernelSHAP and Shapley Value Sampling.

4.2.2 Evaluation

Evaluations were conducted on three distinct models: GPT-2, Mistral 7B, and LlaMa 2 7B. Specifically, these are the openly available version of GPT-2, Mistral 7B Instruct v0.2, LlaMa 2 7B Chat HF (HF = Huggingface version). Additionally, some tests were run with GODEL v1.1 Large. These models operated under identical settings and received the same input. Four interpretability methods—LIME, KernelSHAP, Shapley Value Sampling, and PartitionSHAP—were assessed, focusing on their explanation quality and runtime efficiency. However, PartitionSHAP was only applied to GPT-2, as its implementation with the other models was not feasible. The tests were performed on all three models after each other. This was possible though using the maximal available hardware acceleration option from google colab and a common approach to reduce memory footprint without losing performance that is provided by the transformer's library, 8-bit Matrix Multiplication (Dettmers et al. 2022). The used virtual machine had 80GB of RAM and 40GB of GPU memory available on a Nvidia A100 GPU. This significantly shortened runtime to an average of 1/10 of utilizing only a CPU. Another full testing run was done using only a CPU and in a sequential way as the hardware setup was only able to manage one model at a time. The runs were aborted after a maximum of 4 hours, after which the Shapley Value Estimation with Mistral and LlaMa models was not finished. The data presented in the results reflects on the generated explanations and the performance of different methods depending on the utilized hardware acceleration.

The evaluation algorithm instantiated interpretability classes for the requested models and model-method combination, from which it generated explanations for the model inputs. For GPT-2 this was a simple message. For Mistral 7B and LlaMa 2 7B the input was formatted according to prompt conventions and including a standardized system prompt, as the used models were finetuned for chat. Across all explanation runs the runtime was measured. A standardized plotting function was used to display the individual explanations, ensuring compatibility across different methods. Explanations were displayed in sequence and token to token level, depending on the used implementation. Sequence wise explanations show the attribution of only the input tokens. Token to token explanations show attribution values of each input token relative to each output token in a heatmap. Style wise the created graphs were oriented on the coloring defined by SHAP, to further standardize the output. This means that positive attributions are magenta and negative attribution sky blue. A total number of 4-6 graphs with explanations were created for each model run as well as a table of runtimes for each different approach.

Furthermore, the evaluation generated numerous insights into the applicability of different approaches on models and the use case of chatbot applications.

4.3 Application Prototype

The application prototypes combine the gathered insights into a prototypical chatbot application that is enhanced with model interpretability. This allows the user to better understand model output through an easily accessible UI implementation. The goal of the application was to implement a prototype as close as possible to standards established by commonly used applications like ChatGPT and enhance this with understandable explanations that fulfill the goals of XAI as close as possible.

4.3.1 Application Architecture

The application architecture is as displayed in the following graphic. The prototype is a module-based python application that adheres to the principles of object-oriented programming. The application architecture is shown below (figure 4).

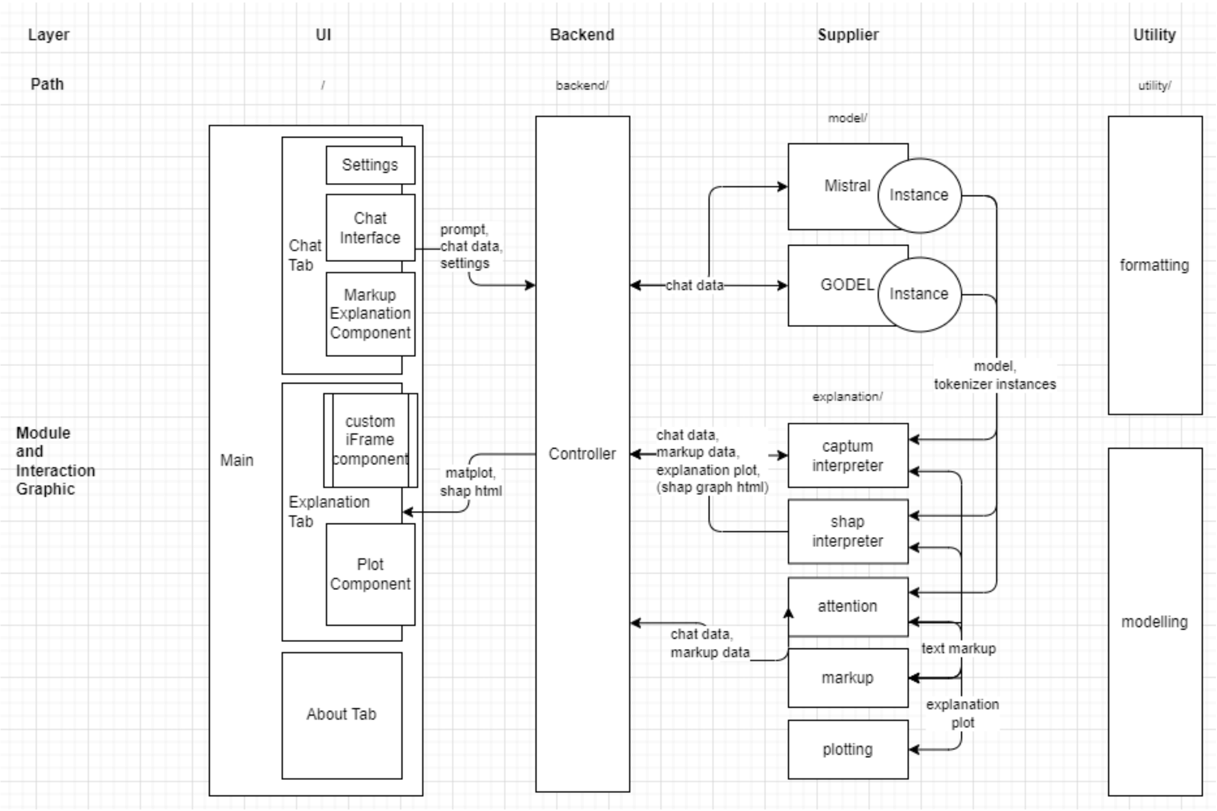


FIGURE 4: GRAPHIC DEPICTION OF APPLICATION ARCHITECTURE.

The application is divided into a main application file defining the UI and referencing the backend controller, which is used for any interference and itself depends on other modules in the supplier layer. The implement and provide the used models, implementations of interpretability approaches and connected modules for visualization. Additionally, the application relies on various utility functions.

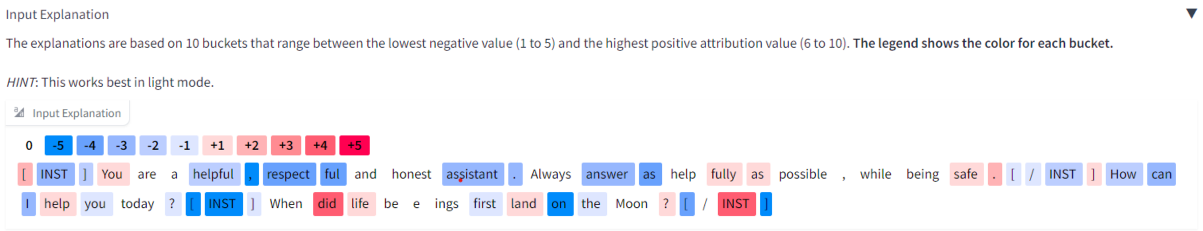
The application prototype employs two distinct models: Mistral 7B Instruct v2.0 and GODEL v1.1 Large. Mistral 7B Instruct 2.0, henceforth referred to as 'Mistral', is a version of the Mistral 7B model, specifically fine-tuned for chat interactions through instruction tuning. GODEL diverges in architectural design, as it is an encoder-decoder model for sequence-to-sequence text generation. Its highly reliant input text for answer generation and contrasts with Mistral, which leverages its extensive pretraining. Mistral was chosen as the main underlying Large Language Models. Mistral 7B is among the highest performing models, outputting high quality text of high conversational quality. Through this it maintains remarkable efficiency during runtime. In total it best fulfills two of user needs defined earlier for chatbots: language understanding and personalization. It furthermore employs several significant improvements outlined earlier and represents the widely used class of decoder only models, especially in chatbot settings. Additionally, GODEL was added for three main reasons. Firstly, GODEL's comparatively smaller size, compared with counterparts like Mistral or LlaMa 2, allows it to operate on local systems with a satisfactory performance and leads to better compatibility with free or low-cost hosting services. While Mistral is capable of local deployment, it demands robust hardware and an investment in hosting infrastructure for optimal performance. Moreover, GODEL is paired with PartitionSHAP, which is the most efficient interpretability approach and intended to be included as a showcase. Lastly GODEL is highly dependency on input data, yielding more detailed explanations for specific inputs and therefore presenting the case for XAI in a different model context.

4.3.2 Interpretability Implementation

Two interpretability approaches were utilized, as the two models present different possibilities. For Mistral, a KernelSHAP implementation from the captum library was used, using the SHAP framework to interpret the model's predictions. In the case of GODEL, the used interpretability approach was PartitionSHAP. Additionally, a simple visualization of model attention was implemented.

As KernelSHAP is implemented for Mistral it similarly is the main interpretability approach used in the prototype. KernelSHAP and PartitionSHAP provide high quality explanations within a reasonable performance, especially compared to Shapley Value Sampling. Compared to methods like LIME or other model agnostic and model specific approaches they have proven to be more versatile and significantly easier to implement while maintaining high quality. Both approaches benefit from the foundational principles of SHAP and Shapley values, ensuring high consistency in explanation delivery and robust expressive power when visualized effectively. A potential confusion arising from negative SHAP values was mitigated with an abstraction technique. Attention visualization was chosen as an additional reference interpretability method, because of its easy implementation and the possibility of abstraction into the same user interface. The implemented approaches present a total of three different visualizations of explanations, depending on the specific method. These will be described in the following.

The main explanation interface abstracts SHAP and attention values in the form of marked up text. It employs eleven distinct 'feature importance buckets', spanning a scale from -5 to +5. Each input token is allocated to a bucket based on its combined SHAP or attention value. The bucketing algorithm programmatically establishes thresholds that evenly distribute the range of values, splitting them into negative impact (-5 to -1) and positive impact (+1 to +5) buckets, with a separate category for values converging towards zero. For attention values, which inherently cannot be negative, the corresponding negative buckets are omitted. These buckets are utilized for SHAP values to display feature importance comprehensively. End-users can easily grasp the relative importance of input tokens, with the most impactful tokens highlighted in magenta and the least impactful, or those with negative SHAP values, marked in blue. An example is visible below (figure 5). This color scheme aligns with the conventional SHAP value color-coding, leading to an intuitive understanding of feature importance akin to the color-coded graphs used in methodological evaluations.



Input Explanation

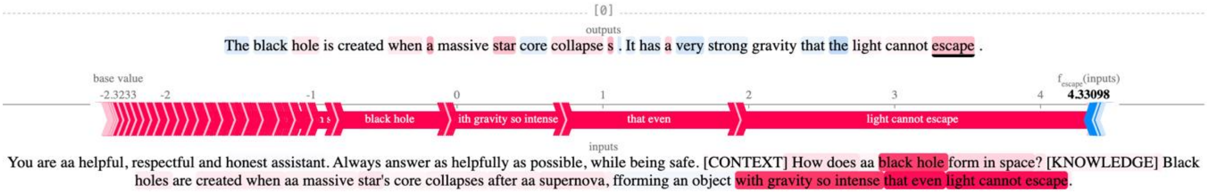
The explanations are based on 10 buckets that range between the lowest negative value (1 to 5) and the highest positive attribution value (6 to 10). **The legend shows the color for each bucket.**

HINT: This works best in light mode.

[INST] You are a helpful , respect ful and honest assistant . Always answer as help fully as possible , while being safe . [/ INST] How can I help you today ? [INST] When did life be e ings first land on the Moon ? [/ INST]

FIGURE 5: SCREENSHOT OF CUSTOM EXPLANATION VISUALIZATION.

The application's second visualization feature for interpretability is exclusively devised for SHAP value attributions created using GODEL and PartitionSHAP and is derived from the interactive plotting capabilities of the shap package. The shap text graphics allow for a granular, token-to-token attribution that precisely shows the SHAP values associated with each token. For this, the shap package generates an interactive HTML-based graph. This visualization is a scientifically more precise rendition of the model's decision processes. However, it was observed that the complexity inherent in understanding SHAP values might mean that this display is confusing users lacking expertise in XAI. To avoid potential confusion, this advanced visualization has not been incorporated into the application's primary interface tab but into a secondary one. Nevertheless, for professional users with a foundational understanding of SHAP values and a background in XAI, this level of detail is valuable. It provides an in-depth comprehension of the interpretations of AI decisions. Therefore, this advanced visualization is offered as an option within the application but is conditionally available — it is present only when interacting with GODEL and when utilizing explanations derived from PartitionSHAP. An example of the view is visible below (figure 6).



The black hole is created when a massive star core collapse s . It has a very strong gravity that the light cannot escape .

You are aa helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe. [CONTEXT] How does aa black hole form in space? [KNOWLEDGE] Black holes are created when aa massive star's core collapses after aa supernova, fforming an object with gravity so intense that even light cannot escape.

FIGURE 6: SCREENSHOT OF INTERACTIVE SHAP TEXT GRAPHIC.

Thirdly, the application also displays an explanation graph for input token attribution. The shown graphs are based on the graphics created for the evaluation of different approaches and included as an additional explanation output for knowledgeable users. Similarly, it is displayed in the “Explanations” tab meant to address users with more background knowledge. In difference to the interactive shap graphic it is displayed for KernelSHAP and PartitionSHAP based SHAP values and therefore also both models. The graphic has not been implemented for attention visualization because of a lack of informative value.

4.3.3 User Interface

The UI leverages Gradio to provide an easy user experience. The emphasis was placed on developing a simple yet informative user interface that aligns with the accessibility standards of conventional chatbot applications and the defined user need of high user experience. For this, Gradio provides a suite of fitting components and utilities. Which are used throughout the entire application for the UI alongside utility functions like examples and informative messages to users. However, Gradio lacks the support of interactive HTML elements to display interactive explanations from the shap library. A custom component was devised to serve as a container for such functionality, based on a provided workflow from Gradio. This resulted in a custom iFrame component with only minor modifications from an existing template component.

4.3.4 Hardware and Hosting

The application necessitates a substantial amount of RAM for operations, as it is designed to load two full models into memory. Despite these requirements, the application can run locally on machines equipped with 16 GB of RAM or more. The incorporation of a GPU can significantly enhance the efficiency of the interpretability features. For demonstration purposes, the application was hosted on Huggingface Spaces, which provides an accessible platform for deploying machine learning models, along with hardware acceleration capabilities. During the showcase and evaluation phase, the application was operated on a virtual machine configured with a 4-core CPU, 15 GB of RAM, and a Nvidia T4 GPU equipped with 16 GB of memory. The operational cost for this setup is $0.60 per active hour. Within this environment, the Mistral model was configured to run exclusively on the GPU. This was achieved through the utilization of the same 8-bit Matrix Multiplication as in the evaluation step. Conversely, the GODEL model was allocated to run on the CPU using standard RAM, a configuration that suits the less demanding nature of the smaller-sized model and the PartitionSHAP interpretability method.

5 Results

The forthcoming section will provide a concise review of the evaluations conducted on Large Language Model interpretability and the application prototype. A more comprehensive examination will be dedicated to the application prototype, considering its significance as the central implementation of this thesis. The evaluation of the prototype will be structured around the predefined framework and will additionally include a compilation of additional findings.

Throughout the evaluation the two steps focus on different previously defined goals, desirable properties, and user expectations. The research done on LLM interpretability focuses on desired interpretability approach properties as defined. Henceforth the prototype evaluation focuses on the overall goals of XAI and the previously defined user expectations towards chatbot applications. All these properties, goals and user expectations will be outlined again shortly.

5.1 LLM Interpretability

The interpretability of Large Language Models, specifically the chosen examples GPT-2, Mistral 7B, LlaMa 2 7B and partly also GODEL, was evaluated with the focus on the potential usage in chat applications. For this, an evaluation focused on the desirable properties of interpretability methods was performed, separated into qualitative evaluation of the explanations and performance evaluation through the runtime of different methods. Refer to the repository listed in the annex for all results and a selection of results for the mistral model.

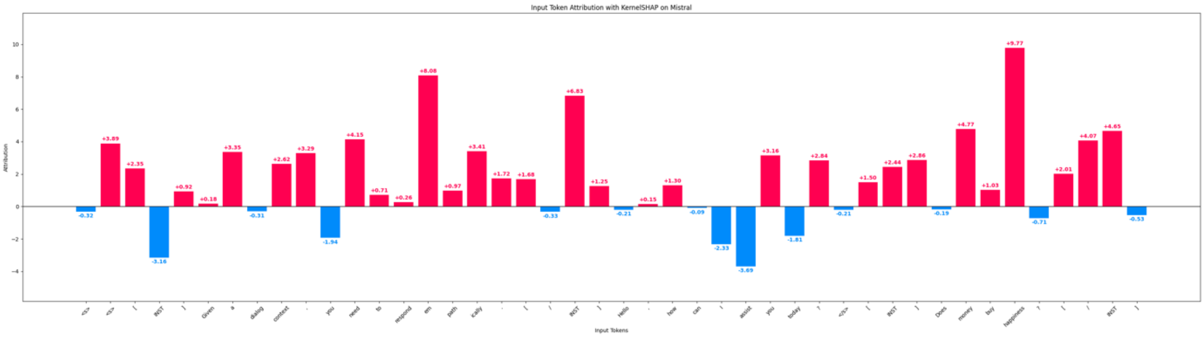
5.1.1 Explanation Quality

The most important consideration for the quality of explanations are the fidelity and consistency of explanations. Fidelity, which is the measure of how accurate an interpretability method or the explanation reflects the internal decision-making process of the model, is a general concern and should be fulfilled as much as possible by any method. Consistency describes how well a method can handle variations in input and consistently provide similar results. Consistency is especially important in the context of the chatbot use case as the textual input can vary greatly. The results and observations connected to these two properties will be outlined in the following.

Fidelity

Fidelity must be mostly assumed based on theoretical foundations and is complicated to test in an interpretation setting, especially since only limited approaches to objective evaluation of fidelity exist. Based on the properties of LIME, KernelSHAP and Shapley Value Sampling an acceptable to good fidelity can be assumed. Upon further comparing the results of LIME and KernelSHAP with that of Shapley Value Sampling, which is the approach with the highest theoretical fidelity of the three, this assumption could be confirmed (see annex for all attribution graphs for Mistral). It can be observed that similar or the same words, i.e., "happiness," or "money" are attributed with the higher attribution scores across all three methods and the simplified sequence attribution graphs, i.e., for Mistral (see Annex). Upon comparison with a the more detailed token to token attribution graphic that can be created using Shapley Value Sampling, the matching attribution between methods can be further confirmed. Similar observations could be made with the results of GODEL and PartitionSHAP, where the attribution of words like "happiness" or "money" is high. It must be noted that the results of a post hoc interpretation of the Mistral model and the GODEL model cannot be directly compared as the models display two different architectural styles. Nevertheless, like the other approaches, it can be assumed that PartitionSHAP does have adequate fidelity based on theoretical foundations.

However, another observation about the handling of input features by the method was made in the process which can impact the reliability of the theoretical fidelity of all methods used on LLMs.



MFIGURE 7: SEQUENCE ATTRIBUTION GRAPH FOR MISTRAL WITH KERNELSHAP.

The graphic above (figure 7) shows the explanation graph of the original input "*Does money buy happiness?"*, combined with prompting tokens and a system prompt for the Mistral Model. Upon inspecting the shown explanations, it's clear that token attribution is calculated on tokens such as "[", "INST ", "/", etc. This is done for each token separately. However, correct would be using "[INST/]" as one token and additionally not attributing anything to the value at all, as it is a token used as part of the prompting of the model. At least when trying to attribute the input and not the impact of wrong prompt formatting. This behavior of explanations persists across all models that need a specially formatted prompt and rely on prompt tokens, i.e., also LlaMa 2 and GODEL. Additionally, it is visible that connected words are separated by the model tokenizer. The separation of words and tokens into fragments is most certainly tokenizer behavior, which is expected when using the model in a normal way, as the model will have access to the entire input sequence (all features/tokens) when generating output text. In this case, the splitting of words into tokens does not impact model behavior. When used as part of an interpretability method this is not necessarily the case; as perturbation-based approaches like the tested LIME, KernelSHAP and Shapley Value Sampling evaluate every (or most) possible combination of input features through masking out input features/tokens. Due to the described token behavior this means that fragments are handled as independent tokens. Which, apart from the attribution of fragments, can lead to two different behaviors. Firstly, evaluation of feature combinations where a token used for prompting can be partly or completely masked out as part of the perturbation process and the model is used in a way it is not designed for. Secondly, sentence combinations that are not represented in English language or have an entirely different meaning might be created as a perturbation.

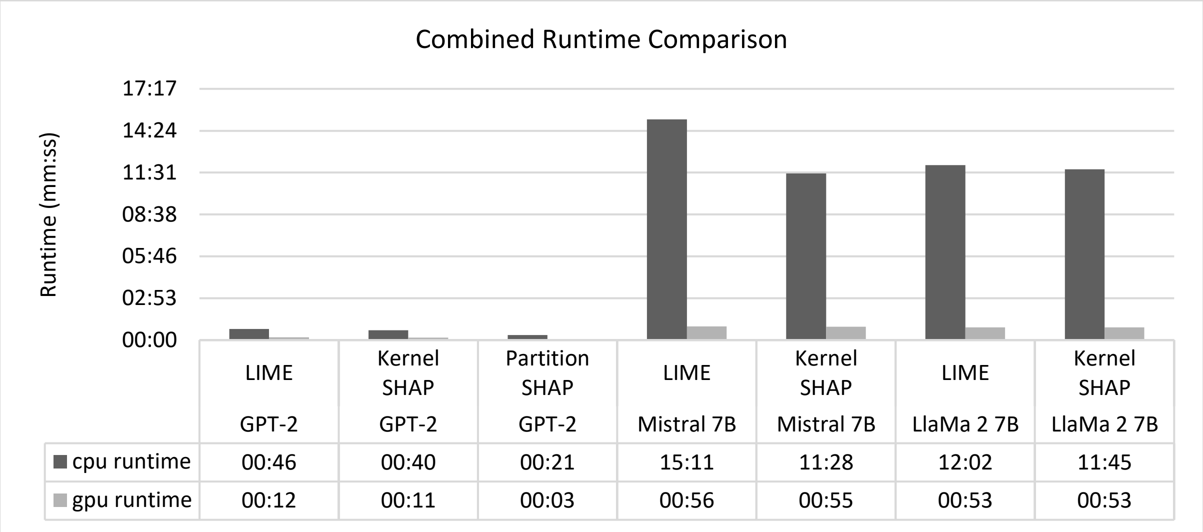
Considering the way tokens are handled and the fact that perturbation-based methods rely on feature independence, which is not inherently present in natural language contexts, it is difficult to definitively assess the fidelity of these interpretability methods. However, when viewed through a logical lens, the overall trends and indications provided by interpretability methods look promising.

Consistency

When comparing the output of different methods both on the same model and on different models a lack of consistency between explanations stands out. This is the case when comparing different methods and when comparing explanations of the same method. The lack of consistency is quite strong, visible i.e. on the attribution of the word *"happiness"* across LIME and KernelSHAP with Mistral and LlaMa 2 models: The theoretically comparable attribution ranges from +2.22 to +30.65, although it must be noted that the trends align across all methods and models – *"happiness"* is important to the model output. Still there is a wide range in the attribution between methods. Furthermore, using the same input several times on the same method and model also shows discrepancies (see annex for graphs on Mistral). It is questionable if the variance in attribution stems from the interpretability method or the variance in model predictions. However, the hyperparameter controlling the randomness of the model is set to a default of 1.0, which should lead to stable answers. This would indicate inconsistencies of the interpretability methods. The overall reason for this lack of consistency remains unclear. However, a technical issue with the implementation code in this thesis is out of question since the value is created by the libraries and not manipulated further.

5.1.2 Runtime

The results below highlight runtime in on the same prompt and with the same model settings. Models that include prompting abilities (like Mistral and LlaMa 2) were prompted as needed by the model, which can lead to difference in the amount of input tokens. This represents a factor potentially leading to minor inaccuracy in the results, however overall trends are still reflected accurately in the results. Runtime was evaluated from the first call of the class creating interpretations until the calculation finished. Since it was not possible to implement PartitionSHAP on Mistral or LlaMa they have not been evaluated in combination. Nevertheless, the PartitionSHAP results on GPT-2 are included to display the significance of runtime difference relative to other methods. The first showcased results exclude Shapley Value Sampling for clarity, as the runtime can far exceed that of other methods. Shapley Value Sampling results are therefore displayed separately further below.

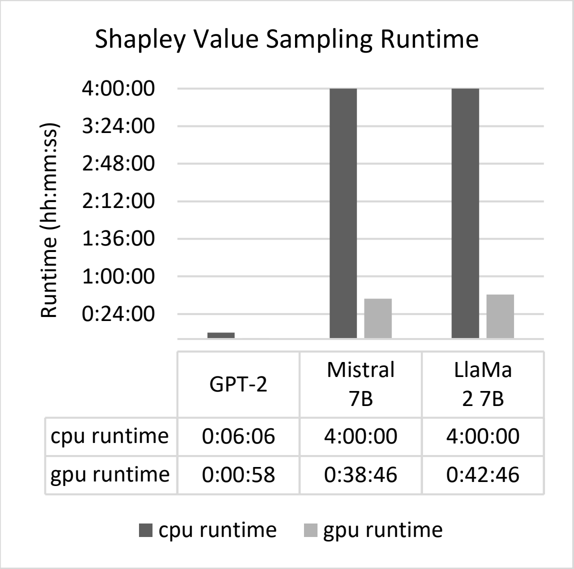


	LIME GPT-2	Kernel SHAP GPT-2	Partition SHAP GPT-2	LIME Mistral 7B	Kernel SHAP Mistral 7B	LIME LlaMa 2 7B	Kernel SHAP LlaMa 2 7B
■ cpu runtime	00:46	00:40	00:21	15:11	11:28	12:02	11:45
▨ gpu runtime	00:12	00:11	00:03	00:56	00:55	00:53	00:53

FIGURE 8: GRAPHIC DISPLAYING RUNTIME OF DIFFERENT INTERPRETABILITY METHOD AND MODEL COMBINATIONS.

Three key observations could be made from the data shown above (figure 8). Upon first analysis, an evident correlation between the size of a model and the runtime of its interpretability methods is clear. This correlation is demonstrated by the contrast in runtime between the 1.5 billion parameter sized GPT-2 model and the larger seven billion parameter models such as Mistral and LlaMa 2. The runtime increase between GPT-2 and Mistral is distinct, with a runtime percentage increase of 1620% on CPU and 400% on GPU for KernelSHAP. A similar picture is presented by the results for LlaMa 2. When the runtimes for GPT-2 are examined in isolation, PartitionSHAP emerges as significantly more efficient than LIME and KernelSHAP, particularly when a GPU is used for acceleration. On average, runtime with PartitionSHAP is between 30% to 50% of other methods under evaluation. Lastly, the deployment of GPU-based hardware acceleration has been shown to drastically reduce the runtime of interpretability methods, even for larger models like Mistral and LlaMa 2. When comparing the performance on a CPU alone to that with GPU acceleration it can be observed that the runtime for KernelSHAP and LIME can be reduced to below one minute. This is a strong improvement from the pure CPU runtime, which exceeds 10 minutes for KernelSHAP and LIME. In certain instances, the runtime with GPU acceleration can decrease to as low as 6% of the runtime required on a CPU.

Shown on the right (figure 9) is a singular graphic for the runtime of Shapley Value Estimation across all three evaluated models and for both CPU and GPU acceleration. The generation of explanations with these methods was stopped after a maximum of 4 hours, at which point the calculations using a CPU were not finished. Again, it is visible that a GPU can significantly increase the efficiency of any method and model. Compared to the other methods, the runtime of Shapley Value Sampling is clearly far out of range of any other methods, as the runtime increases more than 2307% compared to KernelSHAP with Mistral.



	GPT-2	Mistral 7B	LlaMa 2 7B
cpu runtime	0:06:06	4:00:00	4:00:00
gpu runtime	0:00:58	0:38:46	0:42:46

FIGURE 9: GRAPHIC DISPLAYING RUNTIME OF SHAPLEY VALUE SAMPLING WITH DIFFERENT MODELS.

5.2 Implementation Evaluation

The focus of the entire prototype evaluation was how the defined goals of XAI and the user expectations towards chatbots can be fulfilled by the prototype. Based on this, the evaluation of the implementation, specifically the application prototype, was done using the established evaluation framework. Through the framework it was possible to evaluate the XAI goal of *explainability*, which is how well an AI system can explain model decisions and provides insights into the logic and reasoning underlying those decisions. The quality of explanations was additionally evaluated based on a few examples and separated from the testing as part of the framework. Apart from general explanation quality it was evaluated how well the defined user expectations towards *UX/UI*, *language understanding* and *personalization* could be fulfilled. It is encouraged to verify certain results by trying out the prototype application by hand. All evaluations, if not specifically noted otherwise, were run in an accelerated environment as outlined in the implementation section and using the Mistral 7B Instruct model. GODEL was also used in the specified accelerated virtual environment.

5.2.1 Qualitative Evaluation

The qualitative evaluation focuses on the fulfillment of user expectations. Three of the most important user expectations towards chatbots have been previously defined – *UX/UI*, which is the expectation of intuitive and stringboard user experience through UI, *language understanding*, describing the capability to process and understand language in a way similar to humans and *personalization*, which describes how well the application can represent a conversation context. The fulfillment of these three expectations is important to evaluate if an interpretable chatbot application can still maintain a performance on the level of typical openly available software and to draw conclusions towards the practicability of such implementation in other software. The fulfillment of a good user experience was less focused on the quality of the user interface, as high-quality components are provided by Gradio.

Additionally, some insights on explanation quality in the prototype could be compiled. These will also be referred to below.

User Expectations

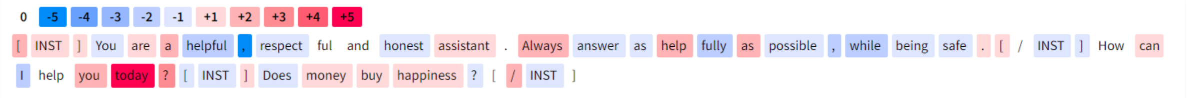
One of the most notable features of good user experience apart from a simplified user interface is the performance of the application. In the context of a chatbot this specifically is the speed to answer. When using the application, set to Mistral and without any explanations, answers are generated withing a few seconds. The answers are of a high quality, which was to be expected from a model that scores high on most common benchmarks (Jiang et al. 2023). The answer to the example "*Does money buy happiness?*" reads as follows: "*Money can provide certain material comfort s and financial security that can contribute to overall well - being and happiness. However, beyond a certain point, additional wealth does not necessarily lead to greater happiness.[…]*." Upon starting a conversation and asking follow up questions, the model gives reasonable responses. From this conversation and similar ones, the prototype, using Mistral in a normal response mode, has a good UX/UI and the model shows adequate language understanding and personalization.

When using the explanation method implementation to augment Mistral model responses the time to answer increases significantly. In the testing setup it takes 62 seconds to generate an answer including an explanation. This time increases with more input tokens, i.e., when starting a conversation, as the model takes all (or most) of the conversation history as input. Upon evaluation a runtime of up to 250 seconds could be recorded. Considering this fact, it can be determined that the user experience is significantly impacted when additionally generating explanations. Otherwise, the model still displays the same qualities since model capabilities are not impacted by the interpretability approach.

Mistral model responses can appear as unfinished and cut off, which is an error in the model usage of Mistral itself, that could not be mitigated. The model setting should not lead to the shown behavior. Another issue with the application quality is the formatting of output text. Because of this, some text can appear with wrong whitespaces or missing special characters such as hyphens or apostrophes. Both issues can be considered as prototype issues that wouldn't persist on a production application.

Explanation Quality

When examining explanations more closely for transparency it is evident that the generated explanations attribute importance to input tokens in an expected fashion. This aligns with the insights generated in the first evaluation of LLM interpretability. This means that i.e. for the example *"Does money buy happiness?"* the most important words by their combined SHAP value are *"money"* and *"happiness"*. This is clearly visible in the interface highlighting the most important tokens, as shown below (figure 10). When also adjusting the system prompt a difference in input token attribution can be observed.



[INST] You are a helpful , respect ful and honest assistant . Always answer as help fully as possible , while being safe . [/ INST] How can I help you today ? [INST] Does money buy happiness ? [/ INST]

FIGURE 10: SCREENSHOT OF SHAP BASED EXPLANATION.

With a model like GODEL, the picture presents itself differently. As the model is basing most of its input on additional knowledge and less on pre-training the explanations clearly show which parts of the additional knowledge have the largest impact on the answer. With the example of "*Does money provide happiness?"* SHAP clearly attributes a significant part of the answer generated by GODEL on the additional knowledge (*"Some studies have found a correlation between income and happiness, but this relationship often has diminishing returns. From a psychological standpoint, it's not just having money, but how it is used that influences happiness.").* This clearly represents the models' internal workings as generative sequence to sequence model.

5.2.2 Framework Based Evaluation

The evaluation framework was developed to formalize a testing process that can assess the helpfulness of explanations and thereby how well the XAI goal of explainability was fulfilled. In the testing framework this is formalized by a helpfulness score that encompasses the fulfillment of explainability, as well as the three subgoals of XAI, as defined in the fundamentals – *transparency*, *interpretability*, and *understandability*. Using the described evaluation framework, the prototype was evaluated on three metrics across ten different combinations of input message, system prompt and conversation history.

These metrics are:

1. Application-grounded helpfulness score (label α).
2. Human-grounded based helpfulness score (label η).
3. Scoring delta between SHAP and Attention Visualization.

The following graphic (Figure 11) depicts the application-grounded and human-grounded evaluation scores for each input and method. Refer to the annex for detailed graphs, showing application- and human-grounded evaluation separately. This is combined with the delta between the two methods as a measure of competitive performance. Shown are the scores for each of the ten input combinations, labeled alphabetically from A to J. The entire evaluation was done using Mistral and KernelSHAP, henceforth references of SHAP explanations refer to explanations created by KernelSHAP on the Mistral Model.

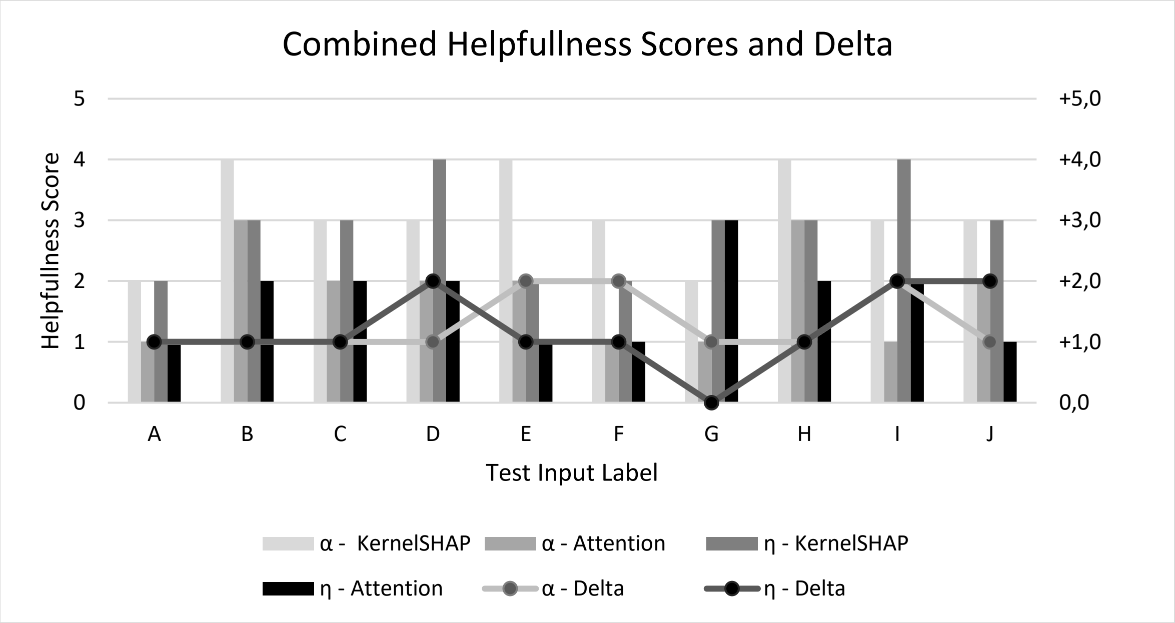


FIGURE 11: GRAPHIC DISPLAYING EVALUATION SCORES AND SCORE DELTA

These combined insights display the helpfulness across a range of different scenarios and two different audience personas. It also gives an indicator for the performance compared to the easiest approach to interpretability, attention visualization. Several observations can be made based on the two displayed graphics. Firstly, the helpfulness scores vary between the different inputs, across both SHAP explanations and attention visualization explanations. This is the case for application- and human-grounded evaluation. Reasons for this could be too high variation between the different test inputs or a lack of consistency of explanations. When further comparing the rating for different inputs it becomes clear that it is a lack of consistency in interpretations, as the ratings for more similar inputs, like A and B, still differ strongly (see annex for all table of all test input and model responses). This aligns with insights gathered in earlier evaluations. What is unclear is if this is an inconsistency in model interpretations or a representation of accurate explanations and a variation in the model, impacted by the slight differences in test input. Additionally, it is evident that the SHAP implementation consistently outperforms the Attention Visualization implementation, visible by the positive delta value. This was to be expected, as Attention Visualization is a strongly criticized approach providing the bare minimum of interpretability.

	Application Grounded Eval	**Human Grounded Eval**
SHAP	3,1	2,9
Attention Visualization	1,8	1,9
Delta	+1,3	+1,0

Table 3: Overview of average helpfulness scores and delta.

The helpfulness scores can also be displayed in an averaged form across all tested inputs, as visible in the table on the left (table 3). This table further enforces some of the observations made based on the graphic. In the table, a clear difference between application and human grounded evaluation is evident, even when averaged. Additionally, the delta between scoring of the SHAP implementation and Attention Visualization is clearly visible, as it is above +1 for both evaluation steps.

Overall, it can be determined that the SHAP implementations perform fine on the established scale. The application reached an average of 3,1 of 5 in application-grounded helpfulness rating and 2,9 of 5 in human-grounded helpfulness scoring. However, if the max score of 5 represents perfect explainability, it can be stated that the goal of explainability could not be fully reached. This is the case for both knowledgeable and not knowledgeable users, represented by application- and human-grounded evaluation respectively.

5.3 Discussion

The shown results present conclusive evidence to discuss the viability of implementing post hoc interpretability and achieved explanation quality with Large Language Models. It also shows the feasibility of a Chatbot application that explains the impact of input features on the model output of an auto-regressive Large Language Model in an understandable way and whether they can maintain the user expectations connected to them. However, the results also show several interesting issues that need to be further discussed, which will be done in the following.

5.3.1 Explanation Quality

As previously noted, a significant issue exists with the explanation quality of interpretability methods applied to Large Language Models due to their indiscriminate attribution of importance to all tokens, which is particularly problematic when using a masking technique. This indiscriminate attribution often results in the fragmentation of special tokens and misuse when masking is applied to parts of prompt tokens. A similar problem arises with split words, where masking can lead to the formation of non-dictionary words, which are not recognized as part of English or any other language. This issue stems from the feature independence assumption that underpins methods like KernelSHAP, LIME, and Shapley Value Sampling, and represents a core criticism of perturbation-based interpretability methods. Moreover, these methods typically fail to account for grammatical dependencies between words when masking, which can alter the intended meaning of word combinations, such as "bad luck." However, in NLP the relationship between words and their contextual meaning is critical.

In theory, the clustering feature of PartitionSHAP could be adapted to align with grammatical features, but as of now, there is no such implementation available in the public domain. Additionally, a new approach, GrammarSHAP, was introduced as an interpretability approach that takes grammatical structures into account; however, its code has not been made publicly available, and its performance lags considerably behind that of PartitionSHAP (Mosca et al. 2022). Overall, adjustments to interpretability methods are needed to better account for use cases in NLP as current principles. Most importantly, the principle of feature independence cannot be applied to natural language input. Furthermore, this partly wrongful attribution of token and text fragments raises questions about the actual fidelity of methods relying on perturbations, when applied to LLMs.

5.3.2 Helpfulness of Explanations

This work has proven that different interpretability approaches can create sound explanations within the context of different models. It has also been proven that the implementation of an application prototype that implements such methods is possible and a high quality can be achieved. However, and this is reflected in the previously described results, the helpfulness of the generated explanations can be questioned. While, when compared to attention visualization, the rating of SHAP based explanations is particularly good, the general helpfulness rating of the implementation is lacking behind. This is only partly represented by the combined average evaluation rating of 3.4 on a 5-point scale.

The observed results on explanation quality of the prototype lead towards a conclusion of limited actual helpfulness. Overall, the explanations can be helpful in certain use cases. However, they do require knowledge of the workings of the model, at the very least a sound understanding of the impact of a prompt, conversation history and system prompt. The actual helpfulness of the chosen explanation form for average human users and the generated explanations in form of SHAP Values should be questioned, as the additional insights gained from the explanations are limited. They do proof however that the model is generating output based in part on the input, with certain tokens being more important. What remains unseen without using extra approaches like i.e., baseline values, is its reliance on pretraining, as this is not explained through interpretations. This leads to one of the key issues with interpretability of Large Language Models, as extensive pretraining on unlabeled data is one of the key attributes of the model category. Currently available approaches to machine learning interpretability do not provide any solution to this. When utilizing GODEL and PartitionSHAP, a clearer association between model input and output is possible. However, this usage of GODEL might fairly represent a chatbot that uses additional knowledge from i.e., a pdf converted into embeddings or a browser search, but it does not fairly represent the behavior displayed by language generation models prompted for open ended text generation.

The captum library has introduced tooling that supports the usage of baseline values (Miglani et al. 2023); however, this approach is barely applicable in a scalable way. Since this context would require programmatic determination of a reasonable baseline text values/equivalents that can be used as a replacement for i.e., the value “happiness” in the “Can money buy happiness?” example. Furthermore, it is still unclear if generated explanations are helpful to users to gain a better understanding of model outcomes, which this work does not confirm and should be determined through i.e., large-scale user interviews. Certainly it is clear that the adoption of tools such as ChatGPT has not been hindered by the lack of explainability (Mollman 2022). A potential solution to this could be combining approaches of interpretable machine learning with research such as the mapping of neurons in LLMs (Schaeffer et al. 2023). However, from a perspective of model and application creator, interpretability can nonetheless provide real value. Especially when utilizing a model that is more dependent on input, like GODEL or a use case that is highly dependent on input, like sentiment classification. Furthermore, generated explanations generate a clearer, helpful picture of feature importance. This is why many studies focus on such tasks. In contrast to that, the value of explanations for open-ended generation based on open question as i.e. “Does money happiness?” is clearly limited, for non-knowledgeable users and specialists in the field alike.

5.3.3 Runtime and Scalability

SHAP methods perform well in explanation quality compared to most other interpretability methods. However, as with most modern interpretability methods, they suffer from a higher runtime when executed on more input features., especially when targeting large language models with many layers. In the run tests it was evident that an increase in model complexity and layers leads to a relative increase in the computation time of the test approaches. This is inherent to their characteristics, especially with perturbation-based approaches like LIME, KernelSHAP, and even greater with Shapley Values. The runtime comparison done as part of the evaluation further proves this point. Notably, these tests were also run with models that rank among the smaller or smallest ones in the domain of Large Language Models: GPT with 1.5 billion, LlaMa 2 7B and Mistral 7B with seven billion Parameters. As the runtime difference between 1.5 million and seven billion is already noticeable, it must be assumed that the difference between a seven billion parameters model and models like the seventy billion parameter-sized LlaMa 2 is even higher. Depending on the hardware acceleration setup and the method, the runtime might well increase into the range of (double-digit) hours, even with one of the most performant methods, PartitionSHAP.

Furthermore, it was proven that there is a clear connection between interpretability performance and computing power. With adequate hardware, the runtime can be lowered significantly. However, the computing resources needed for this must be considered. Time to create interpreted results below one minute was only achieved with a small to the minimum amount of input tokens possible, combined with the strongest available hardware acceleration in the given environment, which costs above 10€ per hour and consists of 80GB RAM and A100 GPU with 40GB memory. This is a setup fit for model tuning or training but certainly not for consistently running an application. The application prototype, when supported by adequate hardware, can respond with text and explanation within 60 seconds. While this is also quite low, especially compared to pure CPU performance and other methods, it is not adequate for real world applications and does not fulfill a common user expectation for software.

Overall, acceptable performance of interpretability methods, even PartitionSHAP, is only achievable with significant computing power. Computing power is beyond reason for production applications. Through the described results and this assumption, it is evident that the currently available methods of model interpretability lack highly performant and runtime-efficient methods. This is most definitely the case when (close to) just-in-time explanations are desired. Recently, modern, better, and faster approaches like FastSHAP (Jethani et al. 2021) have been introduced, although they are still to be proven as suitable approaches on scale and there is no current implementation on Large Language Models.

6 Conclusion

In this thesis, the exploration and development of an interpretable prototype chat application have been presented. The prototype was designed to combine principles of Explainable AI, modern transformer based Large Language Models and interpretable machine learning. It furthermore presents an advanced user interface resulting in a functional chat application. This involved extensive exploration and assessment of various methodologies as preparation for the conception, design and building process of the presented application prototype. The integration of XAI into applications comparable to industry standards, such as ChatGPT, has been thoroughly investigated, highlighting potential challenges, and assessing the practicality of such integrations. It has been established that the creation of a functional prototype is achievable. This prototype can fulfill common user expectations to a large part, if supported by adequate hardware. However, user experience is still impacted and the interpretable chatbot application is not on par with other common chatbot applications. Additionally, it could not be proven to what extend generated explanations are helpful for users, both knowledgeable and not knowledgeable, especially in the context of auto-regressive text generation models like Mistral 7B. Through the work it was possible to discover several important points of needed improvement towards the applicability of post hoc interpretability methods for transformer models and specifically Large Language Models. These limits include fundamental issues with perturbation-based approaches like KernelSHAP and the performance of currently available post hoc interpretability approaches. The presented work is limited by the grade of evaluations done on the application prototype, especially the size of the testing user group. Furthermore, a more scientifically sound evaluation framework could be developed to formalize testing. However, the lack of formalized testing approaches for interpretability methods and their implementation represents a general lack of current research. Additionally, more intense adaptation and tuning of currently available approaches towards better usability with Large Language Models could have been done but was deemed out of scope for this thesis.

Three main future research directions can be recognized based on the presented results. These focus on enhancing currently existing methods for post hoc interpretability to account for LLM, especially the extensive usage of prompting and their increasing size. Additionally, the complexity of LLMs calls for the development of newer, innovative interpretability methods to create explanations fast and with a higher explanatory value than currently existing methods. Finally, the convergence of XAI with leading-edge research in foundational AI, as seen by anthropic and other research, might provide more valuable insights into model decision than exclusively relying on method of machine learning interpretability.

In conclusion, this thesis has not only demonstrated the feasibility of integrating interpretability into a chat application that aligns with industry standards but has also mapped the current limitations of the field of XAI and available approaches of post hoc interpretability for transformer models. As an initial venture into this field, significant contributions have been made towards identifying key areas for future research and highlighting the crucial challenges that need addressing.

Publication Bibliography

Abdulla, Hussam; Eltahir, Asim Mohammed; Alwahaishi, Saleh; Saghair, Khalifa; Platos, Jan; Snasel, Vaclav (2022): Chatbots Development Using Natural Language Processing: A Review. In : 2022 26th International Conference on Circuits, Systems, Communications and Computers (CSCC). Crete, Greece, 7/19/2022 - 7/22/2022: IEEE, pp. 122–128.

Abid, Abubakar; Abdalla, Ali; Abid, Ali; Khan, Dawood; Alfozan, Abdulrahman; Zou, James (2019): Gradio: Hassle-Free Sharing and Testing of ML Models in the Wild. Available online at http://arxiv.org/pdf/1906.02569.pdf.

Adamopoulou, Eleni; Moussiades, Lefteris (2020): Chatbots: History, technology, and applications. In Machine Learning with Applications 2, p. 100006. DOI: 10.1016/j.mlwa.2020.100006.

Alec Radford; Karthik Narasimhan; Tim Salimans, Ilya Sutskever (2018): Improving Language Understanding by Generative Pre-Training. Available online at https://www.semanticscholar.org/paper/Improving-Language-Understanding-by-Generative-Radford-Narasimhan/cd18800a0fe0b668a1cc19f2ec95b5003d0a5035.

Amatriain, Xavier; Sankar, Ananth; Bing, Jie; Bodigutla, Praveen Kumar; Hazen, Timothy J.; Kazi, Michaeel (2023): Transformer models: an introduction and catalog. Available online at http://arxiv.org/pdf/2302.07730v3.

Ancona, Marco; Ceolini, Enea; Öztireli, Cengiz; Gross, Markus (2019): Gradient-Based Attribution Methods. In Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, Klaus-Robert Müller (Eds.): Explainable AI. Interpreting, explaining and visualizing deep learning, vol. 11700. Cham, Switzerland: Springer (LNCS sublibrary. SL 7, Artificial intelligence, 11700), pp. 169–191.

Antonini, Cédric (2023): Navigating the EU AI Act: How Explainable AI Simplifies Regulatory Compliance. In Positive Thinking Company, 6/7/2023. Available online at https://positivethinking.tech/insights/navigating-the-eu-ai-act-how-explainable-ai-simplifies-regulatory-compliance/, checked on 11/6/2023.

Arras, Leila; Osman, Ahmed; Samek, Wojciech (2022): CLEVR-XAI: A benchmark dataset for the ground truth evaluation of neural network explanations. In Information Fusion 81, pp. 14–40. DOI: 10.1016/j.inffus.2021.11.008.

Arrieta, Alejandro Barredo; Díaz-Rodríguez, Natalia; Del Ser, Javier; Bennetot, Adrien; Tabik, Siham; Barbado, Alberto et al. (2019): Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI. Available online at https://arxiv.org/pdf/1910.10045.pdf.

Baehrens, David; Schroeter, Timon; Harmeling, Stefan; Kawanabe, Motoaki; Hansen, Katja; Mueller, Klaus-Robert (2009): How to Explain Individual Classification Decisions, checked on 7/27/2023.

Beltagy, Iz; Peters, Matthew E.; Cohan, Arman (2020): Longformer: The Long-Document Transformer. Available online at http://arxiv.org/pdf/2004.05150v2.

Bhattacharya, Aditya (2022): Applied machine learning explainability techniques. Best practices for making ML algorithms interpretable in the real-world applications using LIME, SHAP and others. Birmingham: Packt Publishing.

Covert, Ian; Lundberg, Scott; Lee, Su-In (2020): Explaining by Removing: A Unified Framework for Model Explanation. Available online at http://arxiv.org/pdf/2011.14878.pdf.

Deng, Li; Liu, Yang (2018): Deep learning in natural language processing. Singapore: Springer.

Dettmers, Tim; Lewis, Mike; Belkada, Younes; Zettlemoyer, Luke (2022): LLM.int8: 8-bit Matrix Multiplication for Transformers at Scale. Available online at http://arxiv.org/pdf/2208.07339.pdf.

Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton; Toutanova, Kristina (2018): BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Available online at https://arxiv.org/pdf/1810.04805.pdf.

Doshi-Velez, Finale; Kim, Been (2017): Towards A Rigorous Science of Interpretable Machine Learning. Available online at http://arxiv.org/pdf/1702.08608v2.

Edoardo Mosca; Ferenc Szigeti; Stella Tragianni; Daniel Gallagher; Georg Groh (2022): SHAP-Based Explanation Methods: A Review for NLP Interpretability. In Proceedings of the 29th International Conference on Computational Linguistics, pp. 4593–4603. Available online at https://aclanthology.org/2022.coling-1.406/.

Erik Štrumbelj; I. Kononenko (2010): An Efficient Explanation of Individual Classifications using Game Theory. In Journal of Machine Learning Research. Available online at https://www.semanticscholar.org/paper/An-Efficient-Explanation-of-Individual-using-Game-%C5%A0trumbelj-Kononenko/7715bb1070691455d1fcfc6346ff458dbca77b2c?p2df.

Galitsky, Boris (2019): Developing Enterprise Chatbots. Learning Linguistic Structures. Cham, Switzerland: Springer.

Gunning, David; Aha, David W. (2019): DARPA's Explainable Artificial Intelligence Program. In AI Magazine 40 (2), pp. 44–58. DOI: 10.1609/aimag.v40i2.2850.

Hao, Yaru; Dong, Li; Wei, Furu; Xu, Ke (2020): Self-Attention Attribution: Interpreting Information Interactions Inside Transformer. Available online at http://arxiv.org/pdf/2004.11207v2.

Hariri, Walid (2023): Unlocking the Potential of ChatGPT: A Comprehensive Exploration of its Applications, Advantages, Limitations, and Future Directions in Natural Language Processing.

Hedström, Anna; Weber, Leander; Bareeva, Dilyara; Krakowczyk, Daniel; Motzkus, Franz; Samek, Wojciech et al. (2023): Quantus: An Explainable AI Toolkit for Responsible Evaluation of Neural Network Explanations and Beyond. In Journal of Machine Learning Research. Available online at http://arxiv.org/pdf/2202.06861v3.

Hoffmann, Jordan; Borgeaud, Sebastian; Mensch, Arthur; Buchatskaya, Elena; Cai, Trevor; Rutherford, Eliza et al. (2022): Training Compute-Optimal Large Language Models.

Holzinger, Andreas; Saranti, Anna; Molnar, Christoph; Biecek, Przemyslaw; Samek, Wojciech (2022): Explainable AI Methods - A Brief Overview. In Andreas Holzinger, Randy Goebel, Ruth Fong, Taesup Moon, Klaus-Robert Müller, Wojciech Samek (Eds.): xxAI - Beyond Explainable AI: International Workshop, Held in Conjunction with ICML 2020, July 18, 2020, Vienna, Austria, Revised and Extended Papers. Cham: Springer International Publishing (Lecture Notes in Computer Science), pp. 13–38, checked on 7/28/2023.

Jain, Mohit; Kumar, Pratyush; Kota, Ramachandra; Patel, Shwetak N. (2018): Evaluating and Informing the Design of Chatbots. In Ilpo Koskinen, Youn-kyung Lim, Teresa Cerratto-Pargman, Kenny Chow, William Odom (Eds.): Proceedings of the 2018 Designing Interactive Systems Conference. DIS '18: Designing Interactive Systems Conference 2018. Hong Kong China, 09 06 2018 13 06 2018. New York, NY, USA: ACM, pp. 895–906.

Jethani, Neil; Sudarshan, Mukund; Covert, Ian; Lee, Su-In; Ranganath, Rajesh (2021): FastSHAP: Real-Time Shapley Value Estimation. Available online at http://arxiv.org/pdf/2107.07436.pdf.

Jha, Ashish Ranjan (2023): Mastering Pytorch. Build powerful deep learning architectures using advanced pytorch features. [S.l.]: Packt Publishing.

Jiang, Albert Q.; Sablayrolles, Alexandre; Mensch, Arthur; Bamford, Chris; Chaplot, Devendra Singh; Casas, Diego de Las et al. (2023): Mistral 7B. Available online at https://arxiv.org/pdf/2310.06825.pdf.

Kaplan, Jared; McCandlish, Sam; Henighan, Tom; Brown, Tom B.; Chess, Benjamin; Child, Rewon et al. (2020): Scaling Laws for Neural Language Models. Available online at https://arxiv.org/pdf/2001.08361.pdf.

Kochkach, Arwa; Kacem, Saoussen Belhadj; Elkosantini, Sabeur; Lee, Seongkwan M.; Suh, Wonho (2023): On the Different Concepts and Taxonomies of eXplainable Artificial Intelligence. In Akram Bennour, Ahmed Bouridane, Lotfi Chaari (Eds.): Intelligent System an Patter Recognition. Third international conference, vol. 1941. [S.l.]: Springer International Publishing (Communications in Computer and Information Science), pp. 75–85.

Kokhlikyan, Narine; Miglani, Vivek; Martin, Miguel; Wang, Edward; Alsallakh, Bilal; Reynolds, Jonathan et al. (2020): Captum: A unified and generic model interpretability library for PyTorch. Available online at http://arxiv.org/pdf/2009.07896.pdf.

Lee, Raymond S. T. (2024): Natural language processing. A textbook with Python implementation. Singapore: Springer.

Li, Haoran; Liu, Yiran; Zhang, Xingxing; Lu, Wei; Wei, Furu (2023): Tuna: Instruction Tuning using Feedback from Large Language Models. Available online at https://arxiv.org/pdf/2310.13385.pdf.

Linardatos, Pantelis; Papastefanopoulos, Vasilis; Kotsiantis, Sotiris (2020): Explainable AI: A Review of Machine Learning Interpretability Methods. In Entropy (Basel, Switzerland) 23 (1). DOI: 10.3390/e23010018.

Loukides, Michael Kosta (2023): What are ChatGPT and its Friends? Opportunities, costs, and risks for large language models. First edition. Sebastopol, CA: O'Reilly Media, Inc.

Lundberg, Scott M.; Lee, Su-In (2017): A Unified Approach to Interpreting Model Predictions. In Advances in Neural Information Processing Systems 30.

Madsen, Andreas; Reddy, Siva; Chandar, Sarath (2022): Post-hoc Interpretability for Neural NLP: A Survey. In ACM Computing Surveys 55 (8), 155:1-155:42. DOI: 10.1145/3546577.

Masís, Serg (2022): Interpretable Machine Learning with Python. Build explainable, fair, and robust high… -performance models with hands-on, real-world exam. [S.l.]: Packt Publishing Ltd.

Miglani, Vivek; Yang, Aobo; Markosyan, Aram H.; Garcia-Olano, Diego; Kokhlikyan, Narine (2023): Using Captum to Explain Generative Language Models. Available online at http://arxiv.org/pdf/2312.05491.pdf.

Miller, Tim (2017): Explanation in Artificial Intelligence: Insights from the Social Sciences. Available online at http://arxiv.org/pdf/1706.07269v3.

Mollman, Steve (2022): ChatGPT gained 1 million users in under a week. Here’s why the AI chatbot is primed to disrupt search as we know it. Available online at https://finance.yahoo.com/news/chatgpt-gained-1-million-followers-224523258.html, checked on 2/5/2023.

Molnar, Christoph (2022): Interpretable machine learning. A guide for making black box models explainable. Second edition. Munich, Germany: Christoph Molnar. Available online at https://christophm.github.io/interpretable-ml-book/.

Molnar, Christoph (2023): Interpreting Machine Learning Models With SHAP. A Guide With Python Examples And Theory On Shapley Values. Available online at https://leanpub.com/shap, checked on 11/18/2023.

Mosca, Edoardo; Demirtürk, Defne; Mülln, Luca; Raffagnato, Fabio; Groh, Georg (2022): GrammarSHAP: An Efficient Model-Agnostic and Structure-Aware NLP Explainer. In : Proceedings of the First Workshop on Learning with Natural Language Supervision. Dublin, Ireland: Association for Computational Linguistics, pp. 10–16. Available online at https://aclanthology.org/2022.lnls-1.2, checked on 5/16/2023.

Naveed, Humza; Khan, Asad Ullah; Qiu, Shi; Saqib, Muhammad; Anwar, Saeed; Usman, Muhammad et al. (2023): A Comprehensive Overview of Large Language Models.

OpenAI (2023): GPT-4 Technical Report. Available online at https://arxiv.org/pdf/2303.08774.pdf.

Palacio, Sebastián M.; Lucieri, Adriano; Munir, Mohsin; Hees, Jörn; Ahmed, Sheraz; Dengel, A. (2021): XAI Handbook: Towards a Unified Framework for Explainable AI. In 2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW). Available online at https://www.semanticscholar.org/paper/XAI-Handbook%3A-Towards-a-Unified-Framework-for-AI-Palacio-Lucieri/af85fa6837eac0a72f6ce22cd83643748ed7508f.

Peng, Baolin; Galley, Michel; He, Pengcheng; Brockett, Chris; Liden, Lars; Nouri, Elnaz et al. (2022): GODEL: Large-Scale Pre-Training for Goal-Directed Dialog. Available online at http://arxiv.org/pdf/2206.11309.pdf.

Radford, Alec; Wu, Jeff; Child, Rewon; Luan, David; Amodei, Dario; Sutskever, Ilya (2019): Language Models are Unsupervised Multitask Learners. In. Available online at https://api.semanticscholar.org/CorpusID:160025533.

Remmer, Eliott (2022): Explainability Methods for Transformer-based Artificial Neural Networks: a Comparative Analysis. Stockholm, Sweden.

Ribeiro, Marco Tulio; Singh, Sameer; Guestrin, Carlos (2016a): "Why Should I Trust You?": Explaining the Predictions of Any Classifier. Available online at https://arxiv.org/pdf/1602.04938.pdf.

Ribeiro, Marco Tulio; Singh, Sameer; Guestrin, Carlos (2016b): Model-Agnostic Interpretability of Machine Learning. Available online at http://arxiv.org/pdf/1606.05386v1.

Rothman, Denis (2020): Hands-On Explainable AI (XAI) with Python: Interpret, visualize, explain, and integrate reliable AI for fair, secure, and trustworthy AI apps: Packt Publishing Ltd.

Rothman, Denis; Gulli, Antonio (2022): Transformers for natural language processing. Build, train, and fine-tune deep neural network architectures for NLP with Python, PyTorch, TensorFlow, Bert, and GPT-3 / Denis Rothman. Second edition. Birmingham: Packt Publishing.

Sarti, Gabriele; Feldhus, Nils; Sickert, Ludwig; van der Wal, Oskar (2023): Inseq: An Interpretability Toolkit for Sequence Generation Models. In Proceedings of ACL: System Demonstrations (DOI: 10.18653/v1/2023.acl-demo.40.

Schaeffer, Rylan; Khona, Mikail; Ma, Tzuhsuan; Eyzaguirre, Cristobal; Koyejo, Sanmi; Fiete, Ila R. (2023): Self-Supervised Learning of Representations for Space Generates Multi-Modular Grid Cells. In Thirty-seventh Conference on Neural Information Processing Systems. Available online at https://openreview.net/forum?id=8ox2vrQiTF.

Schwalbe, Gesina; Finzel, Bettina (2023): A comprehensive taxonomy for explainable artificial intelligence: a systematic survey of surveys on methods and concepts. In Data Min Knowl Disc 20 (93), p. 1. DOI: 10.1007/s10618-022-00867-8.

Shazeer, Noam (2020): GLU Variants Improve Transformer. Available online at http://arxiv.org/pdf/2002.05202v1.

Su, Jianlin; Lu, Yu; Pan, Shengfeng; Murtadha, Ahmed; Wen, Bo; Liu, Yunfeng (2021): RoFormer: Enhanced Transformer with Rotary Position Embedding. Available online at http://arxiv.org/pdf/2104.09864v4.

Touvron, Hugo; Martin, Louis; Stone, Kevin; Albert, Peter; Almahairi, Amjad; Babaei, Yasmine et al. (2023): Llama 2: Open Foundation and Fine-Tuned Chat Models.

Tunstall, Lewis; van Werra, Leandro; Wolf, Thomas (2022): Natural language processing with Transformers. Building language applications with Hugging Face / Lewis Tunstall, Leandro von

Werra, and Thomas Wolf ; foreword by Aurélien Géron. Revised (color) edition. Sebastopol, CA: O'Reilly.

Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N. et al. (2017): Attention Is All You Need. Available online at https://arxiv.org/pdf/1706.03762.pdf.

Vig, Jesse (2019): Visualizing Attention in Transformer-Based Language Representation Models. Available online at https://arxiv.org/pdf/1904.02679.pdf.

Wang, Zheng; Long, Chen; Xu, Shihao; Gan, Bingzheng; Shi, Wei; Cao, Zhao; Chua, Tat-seng (2023): LGM3A '23: 1st Workshop on Large Generative Models Meet Multimodal Applications. In ACM Multimedia. Available online at https://www.semanticscholar.org/paper/LGM3A-'23%3A-1st-Workshop-on-Large-Generative-Models-Wang-Long/15c84a980eb2905c6dffb9f9abbea089074b9db8.

Wei, Jason; Tay, Yi; Bommasani, Rishi; Raffel, Colin; Zoph, Barret; Borgeaud, Sebastian et al. (2022): Emergent Abilities of Large Language Models. Available online at https://arxiv.org/pdf/2206.07682.pdf, checked on 1/22/2024.

Woodie, Alex (2021): One Model to Rule Them All: Transformer Networks Usher in AI 2.0, Forrester Says. Available online at https://www.datanami.com/2021/02/16/one-model-to-rule-them-all-transformer-networks-usher-in-ai-2-0-forrester-says/.

Yeh, Catherine; Chen, Yida; Wu, Aoyu; Chen, Cynthia; Viégas, Fernanda; Wattenberg, Martin (2023): AttentionViz: A Global View of Transformer Attention. Available online at https://arxiv.org/pdf/2305.03210.pdf.

Yenduri, Gokul; M, Ramalingam; G, Chemmalar Selvi; Y, Supriya; Srivastava, Gautam; Maddikunta, Praveen Kumar Reddy et al. (2023): Generative Pre-trained Transformer: A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions. Available online at https://arxiv.org/pdf/2305.10435.pdf.

Yildirim, Savas; Asgari-Chenaghlu, Meysam (2021): Mastering Transformers. Build state-of-the-art models from scratch with advanced natural language processing techniques. Birmingham: Packt Publishing. Available online at https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=2985752.

Zhang, Biao; Sennrich, Rico (2019): Root Mean Square Layer Normalization. In Advances in Neural Information Processing Systems 32.

Zhang, Shengyu; Dong, Linfeng; Li, Xiaoya; Zhang, Sen; Sun, Xiaofei; Wang, Shuhe et al. (2023): Instruction Tuning for Large Language Models: A Survey. Available online at http://arxiv.org/pdf/2308.10792v4.

Zhao, Haiyan; Chen, Hanjie; Yang, Fan; Liu, Ninghao; Deng, Huiqi; Cai, Hengyi et al. (2023a): Explainability for Large Language Models: A Survey. Available online at http://arxiv.org/pdf/2309.01029.pdf.

Zhao, Wayne Xin; Zhou, Kun; Li, Junyi; Tang, Tianyi; Wang, Xiaolei; Hou, Yupeng et al. (2023b): A Survey of Large Language Models. Available online at http://arxiv.org/pdf/2303.18223v12.

Zini, Julia El; Awad, Mariette (2023): On the Explainability of Natural Language Processing Deep Models. In ACM Computing Surveys 55 (5), pp. 1–31. DOI: 10.1145/3529755.

Glossary

Word	Description/Definition
EXPLAINABLE AI	Explainable AI (XAI) refers to the notion of making AI systems human-understandable and explain their operations, decisions, and underlying mechanisms, aiming to increase transparency and trustworthiness.
(POST HOC) INTERPRETABILITY METHOD	An interpretability method in is a technique used to create explanations that make decision of machine learning algorithms understandable.
EXPLANATION	An explanation in the context of XAI is a description or rationale provided to clarify how a model arrived at a particular decision or output.
TRANSFORMER	The transformer architecture is a neural network-based architecture relying on self-attention. It's the leading neural network architecture.
AUTO-REGRESSIVE TRANSFORMER	A type of transformer that generates sequences by predicting each element of the sequence based on the previously generated elements.
BI-DIRECTIONAL TRANSFORMER	A bi-directional transformer is a transformers network architecture that reads input data in both forward and backward directions.
SEQUENCE TO SEQUENCE TRANSFORMER	A transformer architecture excelling at the creation of output based on sequence input such as translation tasks
PERMUTATION	Permutation refers to the rearrangement of elements in a dataset or sequence. A permutation is one such rearrangement.
PERTURBATION-BASED	Perturbation-based methods involve using permutations of input feature to evaluate the impact on a models output and decision making.
GRADIENT-BASED	Gradient-based interpretability approaches use the gradients (derivatives) of model outputs to explain model decisions.
(COMPETITIVE) GAME THEORY	Competitive game theory is a branch of mathematics and economics that studies strategic interactions among rational decision-makers.

Annex

Showcase Link and Repositories

Thesis Webapp

Showcase Link: https://lennardzuendorf-thesis.hf.space/

Username: htw

Password: berlin@123

This Link leads to a hosted instance of the application prototype. The given password and username are needed for access. This environment runs on a CPU, but a GPU acceleration can be activated again for a limited time. The performance of CPU and GPU versions are not comparable.

Repository Link: https://github.com/LennardZuendorf/thesis-webapp

This repository contains all the code used to implement the application prototype. The application structure is depicted in the application; folders are named accordingly. The code is extensively commented to explain functionalities and actions and give credits to original authors (marked as "CREDIT: "). The README.md provides a short guide on how the application can be run locally. Additionally, the application can be hosted on any adequately equipped hardware through the provided Dockerfile and the available docker image that can be found at the linked repository.

If not commented otherwise, all visible work/code in this repository can be considered the authors. Credits are given on a function level with an indicator if code was copied, copied, and changed ("adopted") or inspired by somebody else's work. The only exception to this is the custom component that is saved in the components folder. It is based on a template component provided by the Gradio framework that is mostly not the author's work. Several minor changes are marked individually.

Thesis Files

Link: https://github.com/LennardZuendorf/thesis-files

This repository collects all other files used for this work. Most notably, these are several Jupyter notebooks used for prototyping, testing, and implementation. An overview and a brief description of each is given here. Each notebook is thoroughly commented on, including credit and advice on how to run code. Code between notebooks is partially shared and copied across all notebooks. Additionally, the repository contains the graphs and data that were created when evaluating LLM interpretability.

The included notebooks are as follows:

- Exploration: shap – Notebook used to test the shap library and shap implementations for model interpretability. Evaluated with GPT-2, GODEL, Mistral-7B Instruct and LlaMa 7B Chat.
- Exploration: captum - Notebook used for testing the captum library and different implementations for model interpretability. Assessed with GPT-2, GODEL, Mistral-7B Instruct, and LlaMa 7B Chat.
- Exploration: BERTViz - Notebook used to test the BERTViz library and different attention visualization approaches. Evaluated with GPT-2, GODEL, Mistral-7B Instruct and LlaMa 7B Chat.
- Exploration: inseq – Notebook used to test the inseq libraries for usage with GODEL and interpretation of GODEL using GradientSHAP and Integrated Gradients.
- Evaluation: LLM Interpretability – Notebook used to evaluate and compare different post-hoc interpretability approaches with different models. Results have been reflected on in the respective part of this thesis. Evaluation run with various model and interpretability approach implementation combinations.

If not commented otherwise, all visible work/code in this repository can be considered the authors. Credits are given on a function level (marked as “CREDIT:”)

Package Forks

Link: https://github.com/LennardZuendorf/thesis-shap
Link: https://github.com/LennardZuendorf/thesis-captum

For testing and the purpose of fixing implementation problems, the shap and captum repositories have been forked. Changes are mostly limited to small fixed and additional logging. The contributions can be tracked through the git commit history.

The work in these repositories cannot be considered the authors, if not commented otherwise. The specific contributions can be tracked with GitHub.

Interpretability Method Evaluation Results

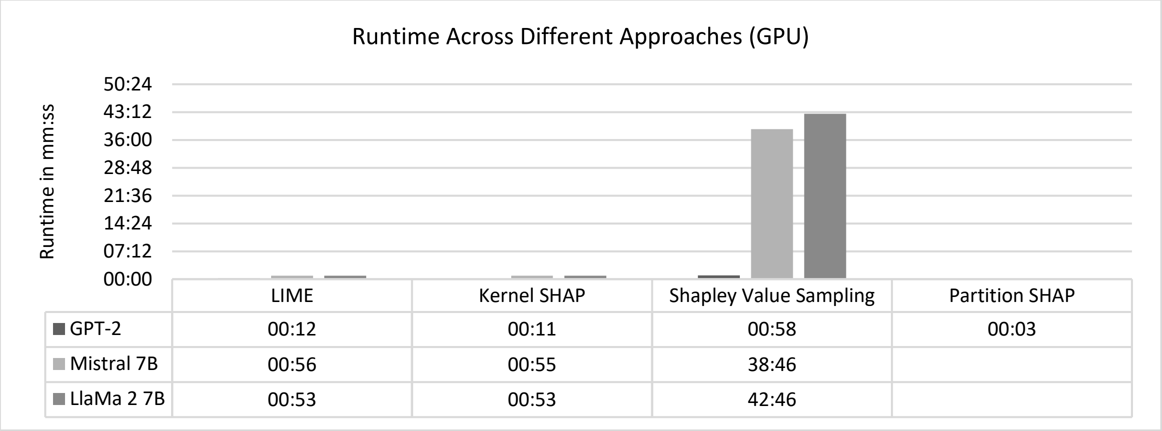
Data

Combined runtime data of all tested interpretability methods and models for both CPU and GPU.

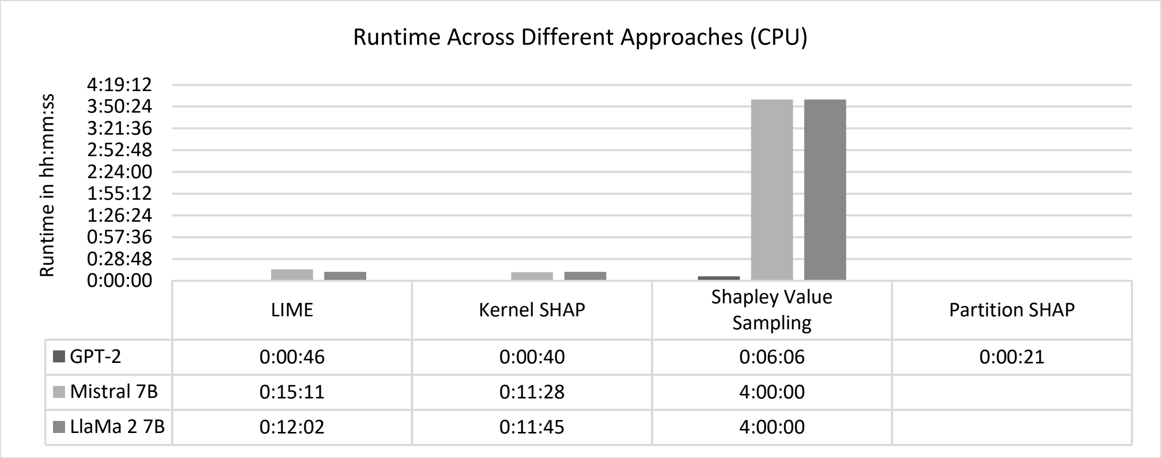
model	method	runtime per hardware	
		GPU runtime	CPU runtime
GPT-2	**LIME**	0:00:12	0:00:46
GPT-2	**Kernel SHAP**	0:00:11	0:00:40
GPT-2	**Shapley Value Sampling**	0:00:58	0:06:06
GPT-3	**Partition SHAP**	0:00:03	0:00:21
Mistral 7B	**LIME**	0:00:56	0:15:11
Mistral 7B	**Kernel SHAP**	0:00:55	0:11:28
Mistral 7B	**Shapley Value Sampling**	0:38:46	4:00:00
LlaMa 2 7B	**LIME**	0:12:02	0:00:53
LlaMa 2 7B	**Kernel SHAP**	0:11:45	0:00:53
LlaMa 2 7B	**Shapley Value Sampling**	4:00:00	0:42:46

Graphics

Runtime of interpretability methods across different approaches and models on CPU and GPU.



	LIME	Kernel SHAP	Shapley Value Sampling	Partition SHAP
■ GPT-2	00:12	00:11	00:58	00:03
■ Mistral 7B	00:56	00:55	38:46	
■ LlaMa 2 7B	00:53	00:53	42:46	



	LIME	Kernel SHAP	Shapley Value Sampling	Partition SHAP
■ GPT-2	0:00:46	0:00:40	0:06:06	0:00:21
■ Mistral 7B	0:15:11	0:11:28	4:00:00	
■ LlaMa 2 7B	0:12:02	0:11:45	4:00:00	

Prototype Evaluation Results

Data

Combined Prototype scoring in application- and human-grounded evaluation for both interpretability approaches and delta in rating.

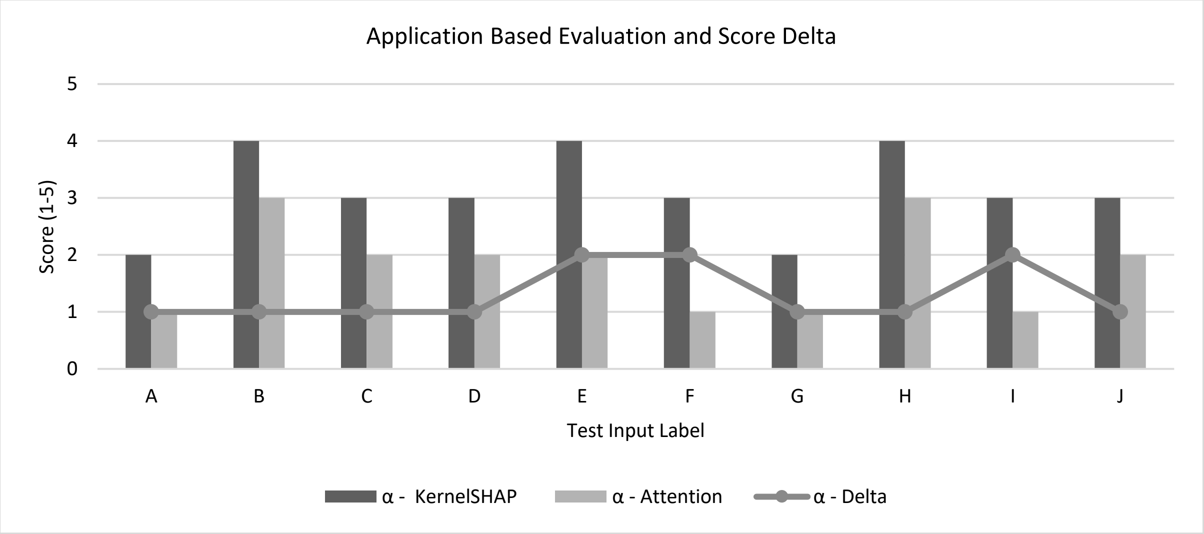
Input Label	**Application-Grounded**			**Human Grounded**			**Combined Delta**
	KernelSHAP	**Attention Visualization**	**Delta**	**KernelSHAP**	**Attention Visualization**	**Delta**	
A	2	1	+1,0	2	1	+1,0	+2,0
B	4	3	+1,0	5	3	+2,0	+3,0
C	3	2	+1,0	4	2	+2,0	+3,0
D	4	2	+2,0	4	1	+3,0	+5,0
E	3	2	+1,0	4	1	+3,0	+4,0
F	2	1	+1,0	2	1	+1,0	+2,0
G	5	3	+2,0	3	3	0,0	+2,0
H	3	3	0,0	4	2	+2,0	+2,0
I	4	1	+3,0	3	3	0,0	+3,0
J	4	1	+3,0	3	2	+1,0	+4,0
Avg	3,4	1,9	+1,5	3,4	1,9	+1,5	+3,0

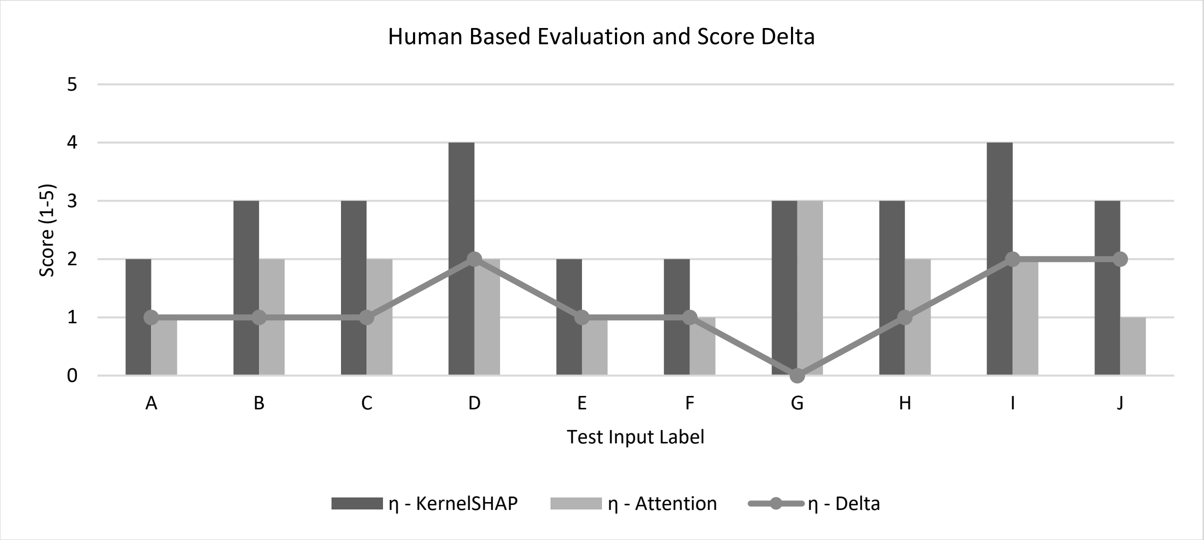
Table of Test Input Data and Model Responses

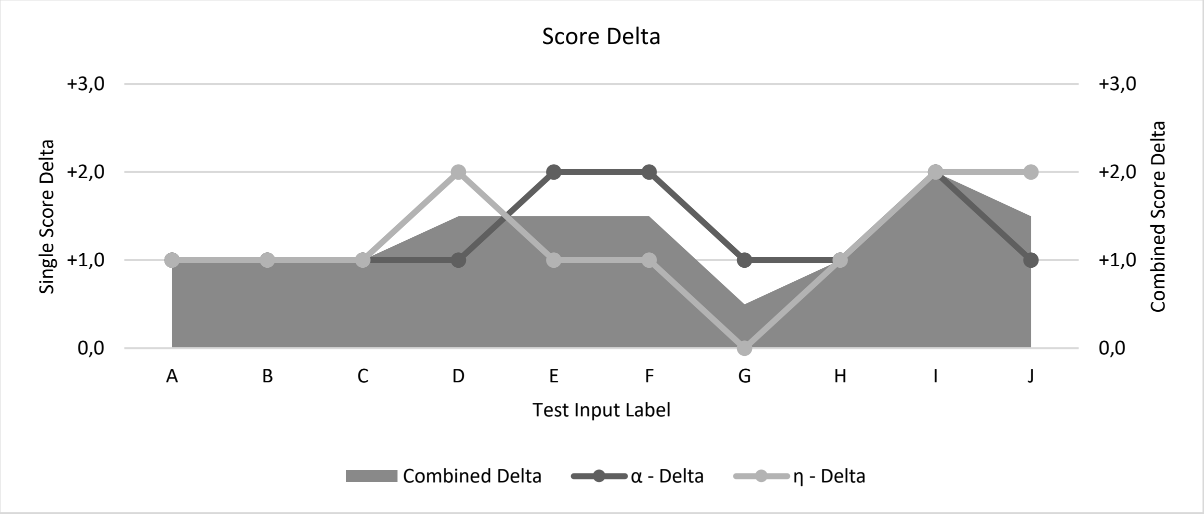
Label	**User Message**	**System Prompt**	**Answer (shortened)**
A	Does money buy happiness?	You are a helpful, respectful, and honest assistant. Always answer as helpfully as possible, while being safe.	Money can provide certain comforts and freedoms that can contribute to overall well - being and happiness, such as financial security […]
B	Does money buy happiness?	Respond from the perspective of a billionaire enjoying life in Dubai.	From my personal experience living as a billion aire in Dub ai, I ' d say that while money certainly provides many comforts […]
C	Who was the first person to walk on the Moon?	You are a helpful, respectful, and honest assistant. Always answer as helpfully as possible, while being safe.	Neil Arm strong was the first man to walk on the moon. He took his famous " One small step for man, one giant leap for man kind " […]
D	When did living beings first land on the Moon?	You are a helpful, respectful, and honest assistant. Always answer as helpfully as possible, while being safe.	I ' d be happy to help answer your question! However, it seems there may be some confusion in your query. The term " life beings " […]
E	Why is the sky blue?	You are a helpful, respectful, and honest assistant. Always answer as helpfully as possible, while being safe.	The color of the sky appears blue due to a phenomenon called Ray le igh scattering. As sunlight reaches Earth ' s atmosphere, […]
F	Follow Up Question: And why is the sky black in the night?	You are a helpful, respectful, and honest assistant. Always answer as helpfully as possible, while being safe.	I apolog ize for any confusion earlier. Here ' s an expanded explanation : The color of the sky varies depending on atm ospher […]
G	Who wrote 'Romeo and Juliet'?	You are a helpful, respectful, and honest assistant. Always answer as helpfully as possible, while being safe.	The play " R ome o and Jul iet " was written by the famous English poet and play wright William Shakespeare around […]
H	Follow up question: What is 'Romeo and Juliet' about?	You are a helpful, respectful, and honest assistant. Always answer as helpfully as possible, while being safe.	I apolog ize for any confusion earlier. Your message did not contain […]
I	What is the meaning of life?	Respond as if you were a Buddhist monk.	In Budd h ism, we believe that the meaning of life is to find inner peace and en light en ment through following the Eight fold Path.[]
J	What is the meaning of life?	Respond as if you were a nihilist.	As a nihilist, I would argue that life inherently has no meaning. The universe and all its events are random and devoid of any ultimate purpose or significance […]

Graphics

Graphics displaying application-, human-grounded helpfulness scoring and combined score delta.


Application Based Evaluation and Score Delta

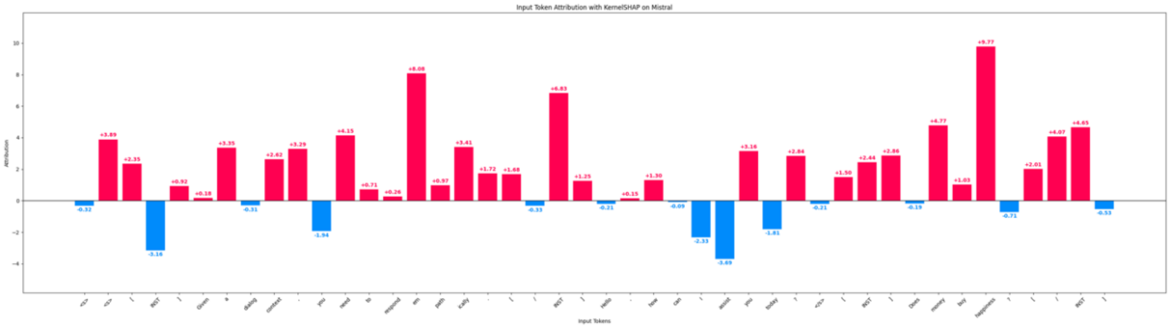

Human Based Evaluation and Score Delta


Score Delta

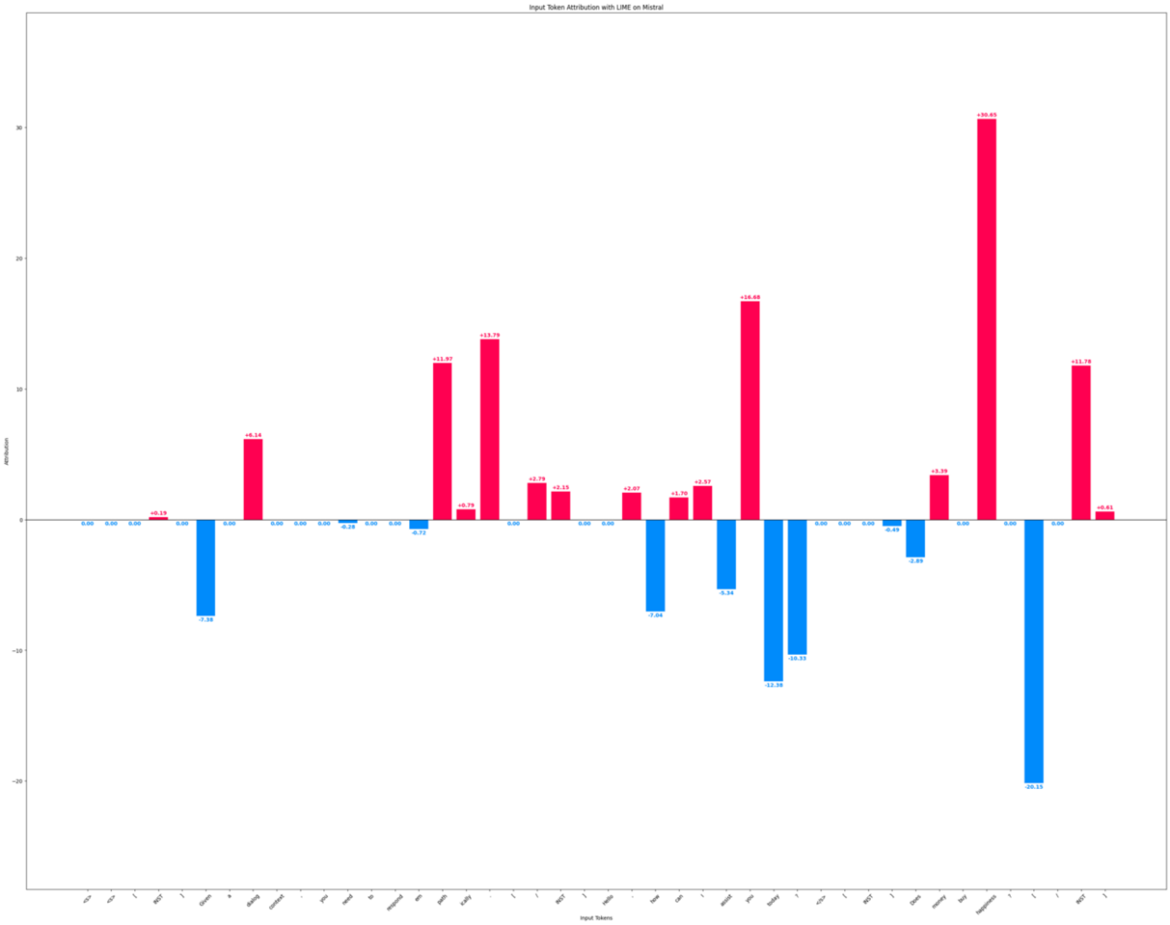
Selected Explanations for Mistral 7B

Sequence Attribution Graphs

Mistral 7B sequence attribution graphic creating using KernelSHAP.

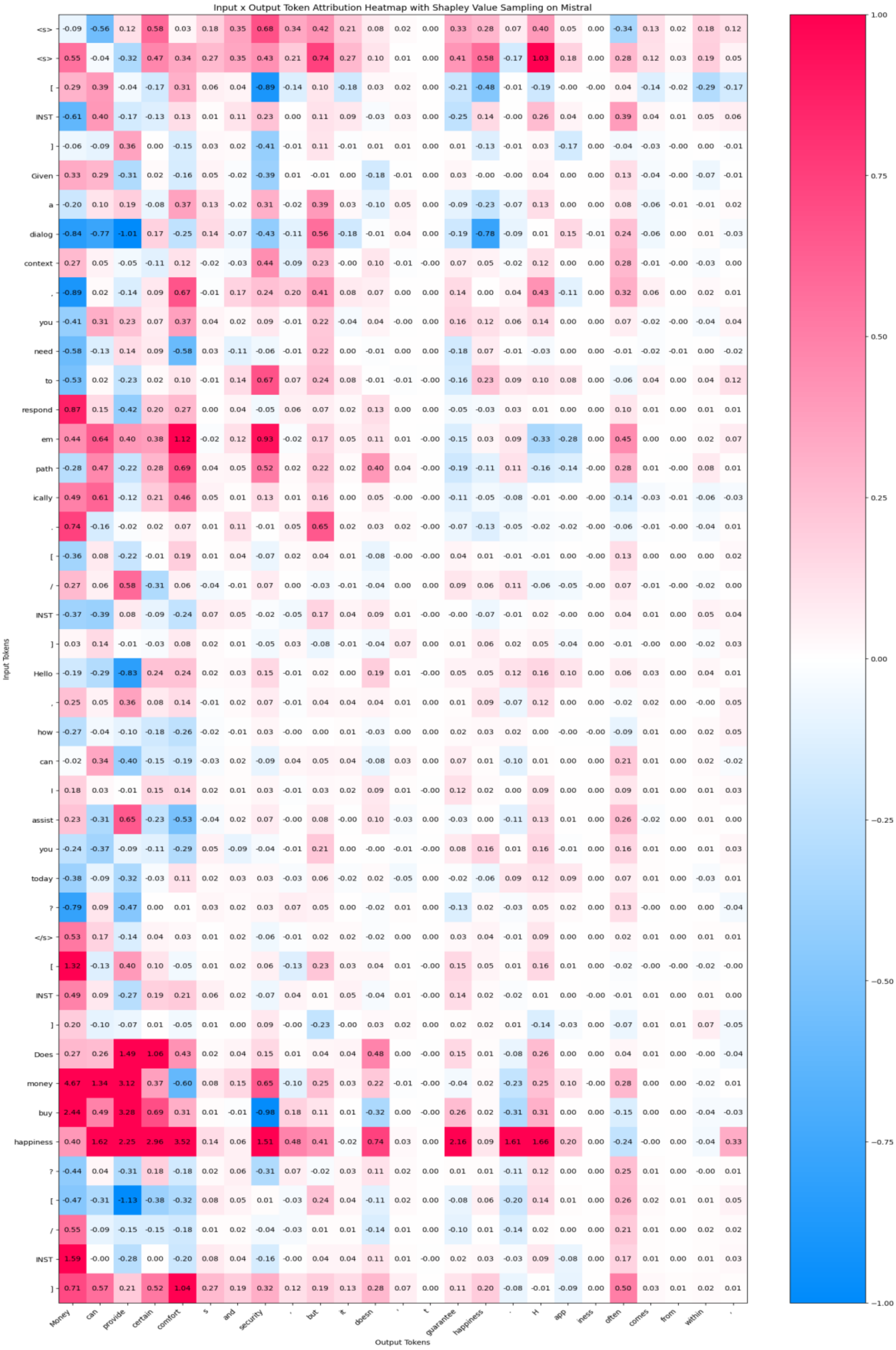


Mistral 7B sequence attribution graphic creating using LIME.



Token Wise Attribution Graphic

Mistral 7B token to token attribution graphic creating using Shapley Value Sampling.



	Money	can	provide	certain	comfort	s	and	security	,	but	it	doesn	'	t	guarantee	happiness	.	H	app	iness	often	comes	from	within	.
<s>	-0.09	-0.56	0.12	0.58	0.03	0.18	0.35	0.68	0.34	0.42	0.21	0.08	0.02	0.00	0.33	0.28	0.07	0.40	0.05	0.00	-0.34	0.13	0.02	0.18	0.12
<s>	0.55	-0.04	-0.32	0.47	0.34	0.27	0.35	0.43	0.21	0.74	0.27	0.10	0.01	0.00	0.41	0.58	-0.17	1.03	0.18	0.00	0.28	0.12	0.03	0.19	0.05
[	0.29	0.39	-0.04	-0.17	0.31	0.06	0.04	-0.89	-0.14	0.10	-0.18	0.03	0.02	0.00	-0.21	-0.48	-0.01	-0.19	-0.00	-0.00	0.04	-0.14	-0.02	-0.29	-0.17
INST	-0.61	0.40	-0.17	-0.13	0.13	0.01	0.11	0.23	0.00	0.11	0.09	-0.03	0.03	0.00	-0.25	0.14	-0.00	0.26	0.04	0.00	0.39	0.04	0.01	0.05	0.06
]	-0.06	-0.09	0.36	0.00	-0.15	0.03	0.02	-0.41	-0.01	0.11	-0.01	0.01	0.01	0.00	0.01	-0.13	-0.01	0.03	-0.17	0.00	-0.04	-0.03	-0.00	0.00	-0.01
Given	0.33	0.29	-0.31	0.02	-0.16	0.05	-0.02	-0.39	0.01	-0.01	0.00	-0.18	-0.01	0.00	0.03	-0.00	-0.00	0.04	0.00	0.00	0.13	-0.04	-0.00	-0.07	-0.01
a	-0.20	0.10	0.19	-0.08	0.37	0.13	-0.02	0.31	-0.02	0.39	0.03	-0.10	0.05	0.00	-0.09	-0.23	-0.07	0.13	0.00	0.00	0.08	-0.06	-0.01	-0.01	0.02
dialog	-0.84	-0.77	-1.01	0.17	-0.25	0.14	-0.07	-0.43	-0.11	0.56	-0.18	-0.01	0.04	0.00	-0.19	-0.78	-0.09	0.01	0.15	-0.01	0.24	-0.06	0.00	0.01	-0.03
context	0.27	0.05	-0.05	-0.11	0.12	-0.02	-0.03	0.44	-0.09	0.23	-0.00	0.10	-0.01	-0.00	0.07	0.05	-0.02	0.12	0.00	0.00	0.28	-0.01	-0.00	-0.03	0.00
,	-0.89	0.02	-0.14	0.09	0.67	-0.01	0.17	0.24	0.20	0.41	0.08	0.07	0.00	0.00	0.14	0.00	0.04	0.43	-0.11	0.00	0.32	0.06	0.00	0.02	0.01
you	-0.41	0.31	0.23	0.07	0.37	0.04	0.02	0.09	-0.01	0.22	-0.04	0.04	-0.00	0.00	0.16	0.12	0.06	0.14	0.00	0.00	0.07	-0.02	-0.00	-0.04	0.04
need	-0.58	-0.13	0.14	0.09	-0.58	0.03	-0.11	-0.06	-0.01	0.22	0.00	-0.01	0.00	0.00	-0.18	0.07	-0.01	-0.03	0.00	0.00	-0.01	-0.02	-0.01	0.00	-0.02
to	-0.53	0.02	-0.23	0.02	0.10	-0.01	0.14	0.67	0.07	0.24	0.08	-0.01	-0.01	-0.00	-0.16	0.23	0.09	0.10	0.08	0.00	-0.06	0.04	0.00	0.04	0.12
respond	0.87	0.15	-0.42	0.20	0.27	0.00	0.04	-0.05	0.06	0.07	0.02	0.13	0.00	0.00	-0.05	-0.03	0.03	0.01	0.00	0.00	0.10	0.01	0.00	0.01	0.01
em	0.44	0.64	0.40	0.38	1.12	-0.02	0.12	0.93	-0.02	0.17	0.05	0.11	0.01	-0.00	-0.15	0.03	0.09	-0.33	-0.28	0.00	0.45	0.00	0.00	0.02	0.07
path	-0.28	0.47	-0.22	0.28	0.69	0.04	0.05	0.52	0.02	0.22	0.02	0.40	0.04	-0.00	-0.19	-0.11	0.11	-0.16	-0.14	-0.00	0.28	0.01	-0.00	0.08	0.01
ically	0.49	0.61	-0.12	0.21	0.46	0.05	0.01	0.13	0.01	0.16	0.00	0.05	-0.00	-0.00	-0.11	-0.05	-0.08	-0.01	-0.00	-0.00	-0.14	-0.03	-0.01	-0.06	-0.03
.	0.74	-0.16	-0.02	0.02	0.07	0.01	0.11	-0.01	0.05	0.65	0.02	0.03	0.02	-0.00	-0.07	-0.13	-0.05	-0.02	-0.02	-0.00	-0.06	-0.01	-0.00	-0.04	0.01
[	-0.36	0.08	-0.22	-0.01	0.19	0.01	0.04	-0.07	0.02	0.04	0.01	-0.08	-0.00	-0.00	0.04	0.01	-0.01	-0.01	0.00	-0.00	0.13	0.00	0.00	0.00	0.02
/	0.27	0.06	0.58	-0.31	0.06	-0.04	-0.01	0.07	0.00	-0.03	-0.01	-0.04	0.00	0.00	0.09	0.06	0.11	-0.06	-0.05	-0.00	0.07	-0.01	-0.00	-0.02	0.00
INST	-0.37	-0.39	0.08	-0.09	-0.24	0.07	0.05	-0.02	-0.05	0.17	0.04	0.09	0.01	-0.00	-0.00	-0.07	-0.01	0.02	-0.00	0.00	0.04	0.01	0.00	0.05	0.04
]	0.03	0.14	-0.01	-0.03	0.08	0.02	0.01	-0.05	0.03	-0.08	-0.01	-0.04	0.07	0.00	0.01	0.06	0.02	0.05	-0.04	0.00	-0.01	-0.00	0.00	-0.02	0.03
Hello	-0.19	-0.29	-0.83	0.24	0.24	0.02	0.03	0.15	-0.01	0.02	0.00	0.19	0.01	-0.00	0.05	0.05	0.12	0.16	0.10	0.00	0.06	0.03	0.00	0.04	0.01
,	0.25	0.05	0.36	0.08	0.14	-0.01	0.02	0.07	-0.01	0.04	0.04	0.04	0.01	0.00	0.01	0.09	-0.07	0.12	0.00	0.00	-0.02	0.02	0.00	-0.00	0.05
how	-0.27	-0.04	-0.10	-0.18	-0.26	-0.02	-0.01	0.03	-0.00	0.00	0.01	-0.03	0.00	0.00	0.02	0.03	0.02	0.00	-0.00	-0.00	-0.09	0.01	0.00	0.02	0.05
can	-0.02	0.34	-0.40	-0.15	-0.19	-0.03	0.02	-0.09	0.04	0.05	0.04	-0.08	0.03	0.00	0.07	0.01	-0.10	0.01	0.00	0.00	0.21	0.01	0.00	0.02	-0.02
I	0.18	0.03	-0.01	0.15	0.14	0.02	0.01	0.03	-0.01	0.03	0.02	0.09	0.01	-0.00	0.12	0.02	0.00	0.09	0.00	0.00	0.09	0.01	0.00	0.01	0.03
assist	0.23	-0.31	0.65	-0.23	-0.53	-0.04	0.02	0.07	-0.00	0.08	-0.00	0.10	-0.03	0.00	-0.03	0.00	-0.11	0.13	0.01	0.00	0.26	-0.02	0.00	0.01	0.00
you	-0.24	-0.37	-0.09	-0.11	-0.29	0.05	-0.09	-0.04	-0.01	0.21	0.00	-0.00	-0.01	-0.00	0.08	0.16	0.01	0.16	-0.01	0.00	0.16	0.01	0.00	0.01	0.03
today	-0.38	-0.09	-0.32	-0.03	0.11	0.02	0.03	0.03	-0.03	0.06	-0.02	0.02	-0.05	0.00	-0.02	-0.06	0.09	0.12	0.09	0.00	0.07	0.01	0.00	-0.03	0.01
?	-0.79	0.09	-0.47	0.00	0.01	0.03	0.02	0.03	0.07	0.05	0.00	-0.02	0.01	0.00	-0.13	0.02	-0.03	0.05	0.02	0.00	0.13	-0.00	0.00	0.00	-0.04
</s>	0.53	0.17	-0.14	0.04	0.03	0.01	0.02	-0.06	-0.01	0.02	0.02	-0.02	0.00	0.00	0.03	0.04	-0.01	0.09	0.00	0.00	0.02	0.01	0.00	0.01	0.01
[	1.32	-0.13	0.40	0.10	-0.05	0.01	0.02	0.06	-0.13	0.23	0.03	0.04	0.01	-0.00	0.15	0.05	0.01	0.16	0.01	0.00	-0.02	-0.00	-0.00	-0.02	-0.00
INST	0.49	0.09	-0.27	0.19	0.21	0.06	0.02	-0.07	0.04	0.01	0.05	-0.04	0.01	-0.00	0.14	0.02	-0.02	0.01	0.00	-0.00	-0.01	0.01	0.00	0.01	0.00
]	0.20	-0.10	-0.07	0.01	-0.05	0.01	0.00	0.09	-0.00	-0.23	-0.00	0.03	0.02	0.00	0.02	0.02	0.01	-0.14	-0.03	0.00	-0.07	0.01	0.01	0.07	-0.05
Does	0.27	0.26	1.49	1.06	0.43	0.02	0.04	0.15	0.01	0.04	0.04	0.48	0.00	-0.00	0.15	0.01	-0.08	0.26	0.00	0.00	0.04	0.01	0.00	-0.00	-0.04
money	4.67	1.34	3.12	0.37	-0.60	0.08	0.15	0.65	-0.10	0.25	0.03	0.22	-0.01	-0.00	-0.04	0.02	-0.23	0.25	0.10	-0.00	0.28	0.00	0.00	-0.02	0.01
buy	2.44	0.49	3.28	0.69	0.31	0.01	-0.01	-0.98	0.18	0.11	0.01	-0.32	0.00	-0.00	0.26	0.02	-0.31	0.31	0.00	0.00	-0.15	0.00	0.00	-0.04	-0.03
happiness	0.40	1.62	2.25	2.96	3.52	0.14	0.06	1.51	0.48	0.41	-0.02	0.74	0.03	0.00	2.16	0.09	1.61	1.66	0.20	0.00	-0.24	-0.00	0.00	-0.04	0.33
?	-0.44	0.04	-0.31	0.18	-0.18	0.02	0.06	-0.31	0.07	-0.02	0.03	0.11	0.02	0.00	0.01	0.01	-0.11	0.12	0.00	0.00	0.25	0.01	0.00	-0.00	0.01
[	-0.47	-0.31	-1.13	-0.38	-0.32	0.08	0.05	0.01	-0.03	0.24	0.04	-0.11	0.02	0.00	-0.08	0.06	-0.20	0.14	0.01	0.00	0.26	0.02	0.01	0.01	0.05
/	0.55	-0.09	-0.15	-0.15	-0.18	0.01	0.02	-0.04	-0.03	0.01	0.01	-0.14	0.01	0.00	-0.10	0.01	-0.14	0.02	0.00	0.00	0.21	0.01	0.00	0.02	0.02
INST	1.59	-0.00	-0.28	0.00	-0.20	0.08	0.04	-0.16	-0.00	0.04	0.04	0.11	0.01	-0.00	0.02	0.03	-0.03	0.09	-0.08	0.00	0.17	0.01	0.00	0.01	0.03
]	0.71	0.57	0.21	0.52	1.04	0.27	0.19	0.32	0.12	0.19	0.13	0.28	0.07	0.00	0.11	0.20	-0.08	-0.01	-0.09	0.00	0.50	0.03	0.01	0.02	0.01

Overview of Used AI Tools

AI TOOL/ASSISTANT	USAGE	LINK
CHATGPT PLUS	rephrasing, broad questions and research, programming help	https://chat.openai.com/
GITHUB COPILOT CHAT	programming help	https://github.com/features/copilot
GRAMMARLY	spelling, grammar, form, rephrasing, plagiarism detection	https://www.grammarly.com/ai
ELICIT	research	https://elicit.com/
CONSENSUS.AI	research	https://consensus.app/

Declaration of Autonomy

I declare that I have written this bachelor’s thesis independently and without outside help and have not used any literature other than that specified. All passages taken verbatim from other authors, as well as those closely based on the ideas of other authors, are specially marked.

This thesis has not been submitted to any other examination authority in the same or a similar form and has not been published.

(Ich versichere hiermit, dass ich die vorliegende Bachelorarbeit selbstständig und ohne fremde Hilfe angefertigt und keine andere als die angegebene Literatur benutzt habe. Alle von anderen Autoren wörtlich übernommenen Stellen wie auch die sich an die Gedankengänge anderer Autoren eng anlehnenden Ausführungen meiner Arbeit sind besonders gekennzeichnet.

Diese Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.)

Berlin, 14.02.2024

X

Lennard Zündorf