

Large Language Model-based Knowledge Creation Verified by Knowledge Graphs and News Articles

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Felix Wilberg, BSc

Registration Number 12229938

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dr. Emanuel Sallinger

Second advisor: Univ.Prof. Dr. Katja Hose

Vienna, September 8, 2025

Felix Wilberg

Emanuel Sallinger



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Verifizierung von LLM-generiertem Wissen mit Knowledge Graphs und Nachrichtenartikeln

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Felix Wilberg, BSc

Matrikelnummer 12229938

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dr. Emanuel Sallinger

Zweitbetreuung: Univ.Prof. Dr. Katja Hose

Wien, 8. September 2025

Felix Wilberg

Emanuel Sallinger



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Felix Wilberg, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, habe ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT-Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 8. September 2025

Felix Wilberg



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

Hallucination is a prevalent phenomenon that manifests in various contexts involving the utilization of Large Language Models (LLMs). When engaging with LLMs, users must remain vigilant against the illusion of accuracy that these models present. This thesis examines the implications of hallucination within LLMs and outlines an approach designed to detect and verify claims within the responses generated by LLMs. To address the issue of hallucination, we propose a claim detection and verification framework for LLMs that encompasses a comprehensive, multi-step methodology. This methodology involves retrieving information through Google Search, extracting data from Wikidata, and ultimately presenting an assessment alongside a justification to the user. We adhere to the Design Science Framework (DSF) to develop a framework that leverages state-of-the-art LLM technology, including structured output, prompt chaining, and role prompting, to identify and verify existing hallucinations. A specifically-tailored gold-standard dataset was meticulously curated, encompassing questions and corresponding correct responses about the executive management positions of all corporations listed in the Fortune 500. We conduct a rigorous evaluation employing our gold data, demonstrating high accuracy and a macro F1 score of 0.8611 for the framework, proving its applicability in real-world environments.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

Halluzinationen sind ein weit verbreitetes Phänomen im Kontext von Large Language Models (LLMs). In den LLM-generierten Antworten steckt oft eine vorgetäuschte Sicherheit der Aussagen, die getroffen wurden. Diese Arbeit untersucht die Auswirkungen von Halluzinationen innerhalb von LLMs und skizziert einen Ansatz zur Erkennung und Überprüfung von Aussagen in den von LLMs generierten Antworten. Um dieses Problem zu bewältigen, schlagen wir ein Framework mit einer mehrstufigen Methodik für die Erkennung und Überprüfung von Behauptungen für LLMs vor. Dieses umfasst das Beschaffen von Informationen über die Google-Suche, das Extrahieren von Daten aus Wikidata und schließlich die Darstellung einer Beurteilung zusammen mit einer Begründung für den Benutzer. Wir arbeiten mit dem Design Science Framework (DSF), um ein Framework zu entwickeln, das modernste LLM-Technologie wie Structured Output, Prompt Chaining und Role Prompting nutzt, um bestehende Halluzinationen zu identifizieren und zu verifizieren. Es wurde ein speziell zugeschnittener Goldstandard-Datensatz zusammengestellt, der Fragen und entsprechend korrekte Antworten zu den Führungspositionen aller in den Fortune 500 gelisteten Unternehmen umfasst. Wir führen eine genaue Evaluierung unter Verwendung unserer Golddaten durch. Wir weisen eine hohe Genauigkeit der Pipeline mittels eines Makro-F1-Scores von 0,8611 nach, der auch die Anwendbarkeit im produktiven Einsatz bestätigt.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Abstract	vii
Kurzfassung	ix
Contents	xi
1 Introduction	1
1.1 Problem Statement and Research Questions	2
1.2 Main Contributions	3
1.3 Methodology	4
1.4 Structure of this Work	6
2 Related Work	7
2.1 Large Language Models and Hallucination	7
2.2 Natural Language Processing Framework for Automated Fact-Checking	12
3 The Claim Detection and Verification Framework	15
3.1 Pipeline Design	15
3.2 LLM Guidance	16
3.3 Claim Detection	19
3.4 Claim Verification	21
3.5 Stance Prediction	26
3.6 Justification Production	26
4 Implementation	27
4.1 Architecture and Technology	27
4.2 Claim Detection	32
4.3 Claim Verification	32
4.4 Deployment and Visualization	39
5 Evaluation	41
5.1 Experimental Setup	41
5.2 Dataset	43
5.3 Quantitative Evaluation	45
	xi

5.4	Results	49
5.5	Discussion	50
6	Conclusion and Future Work	55
6.1	Discussion of Research Questions	55
6.2	Limitations	56
6.3	Future Work	58
6.4	Conclusion	58
7	Appendix	61
7.1	Static SPARQL Queries	61
7.2	Used Prompts in the Framework	63
	Overview of Generative AI Tools Used	71
	List of Figures	73
	List of Tables	75
	Acronyms	77
	Bibliography	79

CHAPTER 1

Introduction

LLMs have become an essential tool in our daily lives, impacting various fields and transcending industry boundaries. Since the release of OpenAI's first public AI chatbot *ChatGPT* [1] based on their Generative Pretrained Transformer (GPT)-3.5 LLM architecture, the AI world has changed drastically. The use case possibilities of its integration are enormous, such as GitHub Copilot, an AI coding assistant built on the architecture of GPT-3.5 [2].

Subsequently, a competitive endeavor emerged to develop the most advanced model. Prominent technology companies, including Meta [3], Google [4], and Alibaba [5], engaged in this pursuit by designing their proprietary models. These models are frequently updated with iterations that offer enhanced parameters, expanded context windows, and increased functionality. As a result, they are attracting significant attention, and a variety of new applications are being realized with their implementation.

However, LLMs also have considerable limitations and risks. Potential biases within responses can lead to prejudiced statements, or a lack of emotional intelligence may result in a misunderstanding of sarcasm or humor [6]. Another primary concern with LLMs is that the content they generate may not always be accurate, leading to a reduction in factual reliability. LLMs occasionally generate responses that sound convincing but are actually incorrect or irrational [1]. Because these models are trained on various datasets, including data scraped from online sources, the presence of inaccurate or misleading information can inadvertently be integrated into the models.

Consequently, users should exercise caution when relying solely on the responses provided by LLMs. Within the Question-Answer functionality of those chatbots, people must be cautious about assuming that these models will consistently deliver accurate answers. Models often exhibit unfounded confidence and misleading certainty, which can lead users to accept the generated responses as correct without realizing it. Augenstein, Baldwin, Cha, *et al.* has demonstrated that LLMs exhibit a documented factuality

issue [7], characterized by statements that are incongruent with the underlying facts, a phenomenon commonly referred to as hallucination. Furthermore, recent models continue to exhibit, or even intensify, these hallucination issues [8]. Numerous researchers have developed benchmarks to evaluate the factuality of LLMs [9], [10], resulting in results that are less than satisfactory.

However, tools like ChatGPT are gaining popularity for routine tasks such as information retrieval and direct question answering. They are increasingly used as primary search engines, positioning themselves as direct competitors of established search engines such as Google [11]. This is particularly true in instances where the information sought by users is semantically complex and cannot be addressed through a straightforward search query within a traditional search engine framework. Additionally, ensuring factual accuracy remains a challenge in these situations.

1.1 Problem Statement and Research Questions

As previously demonstrated in the preceding section, the extent to which users can rely on the responses of the LLM is limited. Numerous factors contribute to the appearance of hallucinations. These can manifest at various stages within LLMs: during data collection, during the training phase of the LLM, or during the model's use when generating a response. We go into the various causes in more detail in Chapter 2. Furthermore, hallucinations are anchored in the architecture of the language models and cannot be completely ruled out. So far, no way has been found to train or use an LLM without hallucinations. This is why the LLM providers only try to minimize hallucinations without eliminating them. So, the issue will also persist in the near future.

Some models have integrated features designed to reduce instances of hallucinations, such as integrated web search functionality. These features identify queries for which the model lacks sufficient information to provide reliable answers. By extracting current and accurate information through web crawling and incorporating it into the response, the accuracy of outputs is enhanced. However, hallucinations can still occur [12].

Therefore, we leave the hallucination issue aside and concentrate on the effects of hallucinations, which are plausible-sounding claims made by the LLM that can be true or false. In this work, our objective is to provide an approach that efficiently detects hallucinations and makes them visible to users, thereby preventing them from believing false claims made by the LLM.

To structure our research, we formulate three research questions (RQ) that accompany this work.

RQ1: What is an appropriate means to effectively validate LLM-created knowledge using knowledge graphs and news sources?

RQ2: What is an appropriate method or metric to measure the success of validation?

RQ3: To what extent can newly created knowledge created by LLMs be effectively validated through news sources and knowledge graphs?

To ensure that our research questions are rigorously addressed, we utilize several methods. For RQ1, we use literature, software documentation, and software development best practices. RQ2 is based on the work of Venable, Pries-Heje, and Baskerville and Pries-Heje, Baskerville, and Venable, where they show strategies to evaluate an artifact within the DSF [13], [14]. We also compare evaluation methods in research areas close to our work. For the last research question, RQ3, we employ the chosen evaluation method to thoroughly assess the demonstrated approach.

1.2 Main Contributions

To address the previously mentioned issues, we propose a claim detection and verification framework for LLMs integrated in a chat user interface. In this work, every generated response from the LLM is checked against current news articles, web sources, and knowledge graphs.

The framework is structured to systematically analyze responses from the LLM to identify claims that comprise factual components in a data pipeline approach. Initially, the framework identifies all claims that contain factual information. Subsequently, in the second phase, it aims to verify each identified claim. This objective is achieved through a comprehensive multistep procedure that includes information retrieval via the Google search engine, data extraction from the Wikidata Knowledge Graph (KG), and ultimately presenting an assessment along with a justification to the user. We use multiple prompting techniques to achieve accurate claim detection and verification. Detecting claims is performed in a single prompting step; the verification phase involves multiple prompts and additional software and data processing phases.

The evaluation utilizes a precisely curated gold-standard dataset that includes 1500 questions about Fortune 500 executive managers, along with their corresponding correct answers. Each question is presented to the LLM, whereupon the generated responses are systematically evaluated.

We make the following contributions.

- We introduce a **holistic framework for claim detection and claim verification**. It serves as a full chat application with a modular and robust pipeline, leveraging prompt chaining and a language-based verification framework
- We curate a **high-quality dataset** comprising 500 Fortune-500 companies and 1,500 questions targeting C-level leadership roles.
- We perform a **rigorous evaluation** using our custom gold standard dataset, demonstrating the accuracy of the pipeline and its applicability in the real world.

- **Extensive documentation** is provided to facilitate replication and further enhance the framework.
- We offer a comprehensive foundation in the detection and mitigation of hallucinations and suggest paths for **further investigation** to enhance understanding in this area.

1.3 Methodology

To address research objectives, this study adopts the DSF originally proposed by Hevner, March, Park, *et al.* in 2004 [15]. This framework was further refined by Hevner, who introduced a more detailed three-cycle view [16]. The original DSF comprises three interrelated components: the environment, information systems research, and the knowledge base. These are operationalized through three iterative cycles as shown in Figure 1.1: the Relevance Cycle, the Rigor Cycle, and the Design Cycle. The primary aim of the DSF is to facilitate the rigorous development of solutions to well-defined research problems. Within the DSF, researchers are expected to construct a purposeful artifact. In the context of this study, the artifact developed is a software solution. In the following subsections, we explain the different components of this framework.

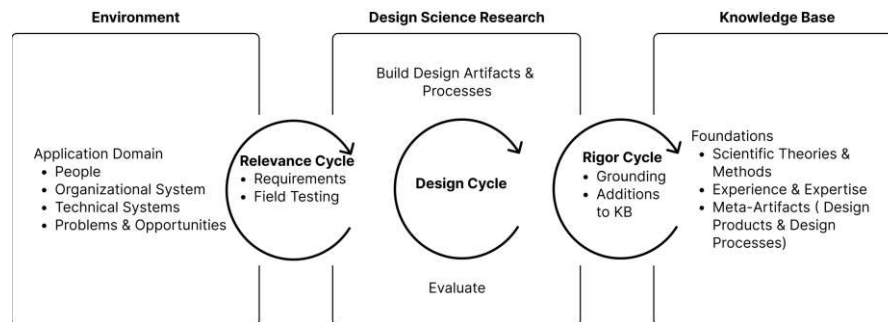


Figure 1.1: The Three Cycle View of Design Science Research by Hevner [16]

Environment The Environment describes the problem space, including people with their characteristics, such as roles or capabilities, organizations, which include processes, culture, and strategies, and the technologies utilized within the organization, including architecture and applications. From then on, business needs emerge from problems, and from these, goals or opportunities arise. The framework ensures that it addresses only the challenges relevant to the environment within the DSF.

Knowledge Base The Knowledge Base serves as the foundational pillar for all of our scientific investigations. It encompasses a wide range of foundation research, including theories, frameworks, instruments, constructs, models, methods, and instantiations. Additionally, it encompasses various methodologies, including data analysis approaches, formal methods, measures, and evaluation criteria. All of these foundations and methods

stem from prior research, and similar topics and reference disciplines offer valuable resources for future studies in different fields. These foundational theories and frameworks are then applied as relevant knowledge within the Design Science (DS) Research domain. Thus, rigor can be achieved by meticulously adhering to established methodologies. A review of related work within the literature identifies relevant sources to build our knowledge base.

Design Science Research The DS Research is carried out when a relevant business need is identified. The framework considers both behavioral science and design science as complementary parts. Hevner *et al.* mention that both disciplines are inseparable; however, we will focus on the design science part. Design science involves the creation of artifacts and their subsequent evaluation and assessment. The insights gained from this evaluation inform the further development and refinement of the artifact. If weaknesses are identified, they can be explored in future research.

Relevance Cycle The Relevance Cycle establishes a connection between the environment and the DS research. This cycle identifies the requirements for the research, posing both problems and opportunities to be addressed, as well as the acceptance criteria for evaluating the artifact to ensure it delivers value to the environment. The findings must be returned to the environment for evaluation. We will use this cycle to verify whether we can achieve satisfactory results within the claim detection and verification pipeline. We make sure that the software artifact meets the actual requirements of the environment, thereby confirming its relevance.

Rigor Cycle The Rigor Cycle connects the knowledge base with the DS research. Within this cycle, the knowledge base provides prior research to support the development of innovations and further scientific contributions to the research community. As noted by Hevner and Iivari, rigor in this research comes from applying appropriate methodologies and foundational principles to develop the artifact. However, they also point out that identifying a foundational theory can often be challenging and that sometimes a combination of different foundational theories may be necessary to ground innovative research.

Design Cycle The Design Cycle iterates between the development and the evaluation of the artifact. This phase requires the most effort, as it encompasses both creation and assessment. By considering environmental requirements and drawing on established foundations and methodologies, we can ensure relevance and scientific rigor. According to Hevner, it is advisable to carry out “multiple iterations of the design cycle in design science research before contributions are output into the relevance cycle and the rigor cycle” [16].

We intend to adopt this approach by iterating multiple times between development and evaluation to create a robust artifact. Our plan involves developing the pipeline and

performing an analytical assessment simultaneously. Additionally, we will integrate the requirements of the relevance cycle, the Natural Language Processing (NLP) framework, and other relevant foundations identified during the literature review into our development and evaluation processes. We outline the methodology, the approach, and the corresponding evaluation in Figure 1.2.

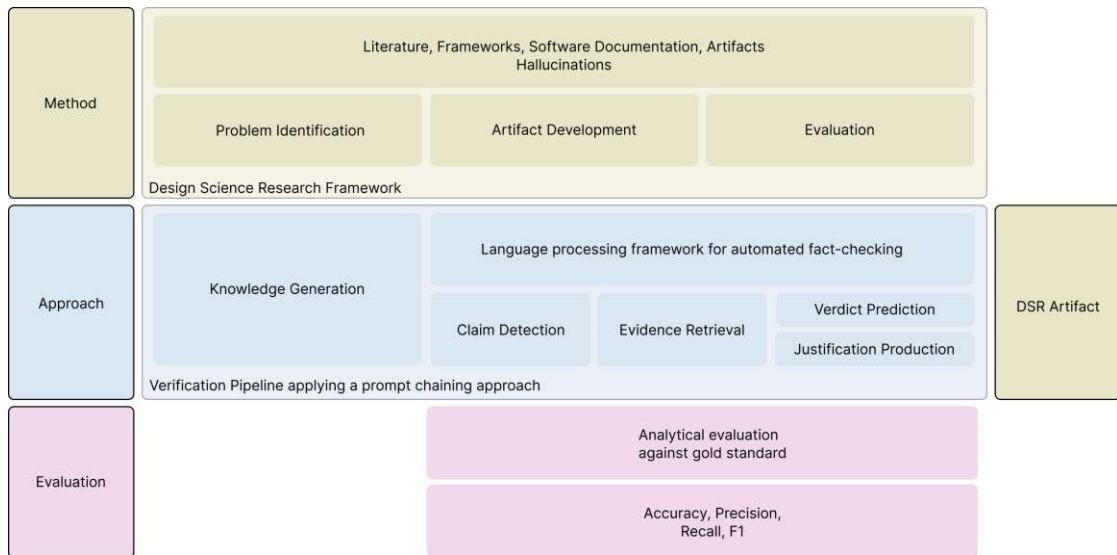


Figure 1.2: Methodology, Approach, and Evaluation of This Work

1.4 Structure of this Work

We structured this work chronologically. Chapter 2 defines our knowledge base and shows current state-of-the-art research, explains how LLMs work, and what different types of hallucination exist. The chapter further shows the language processing framework utilized to detect and verify existing hallucinations. In Chapter 3, we describe the approach chosen by elaborating on how we used a language processing framework to build a claim detection and verification pipeline. The architecture and implementation are described in Chapter 4, where we delve into the technological details. Chapter 5 details our approach chosen for the evaluation and discusses the results we achieve with our framework. Finally, we conclude this work in Chapter 6, providing an overview of the limitations and future work.

Related Work

In this thesis, we follow the design science framework in creating an artifact — a chat interface with integrated claim detection and verification — that can be used for hallucination detection and mitigation. Therefore, we cover the current state-of-the-art in relevant topics and build a foundation. In Section 2.1, we cover the general functionality of LLMs and explain why hallucinations occur or in which way they appear. In Section 2.2, we talk about general fact verification approaches, including existing frameworks. Those frameworks include the steps of claim detection, claim verification, stance prediction, and justification production. Finally, we discover approaches within the C-level domain and why this domain is suitable for fact-checking.

2.1 Large Language Models and Hallucination

LLMs denote a collection of general-purpose artificial intelligence models grounded in transformer architectures. They undergo extensive training on large-scale data to support a wide range of natural language processing tasks. They are also referred to as Foundation Models due to their broad applicability across numerous downstream tasks. Prominent examples include GPT-4 [18], Gemini [4], or LLaMA [3]. The use cases range from code generation [2] to question answering [19] to deep reasoning [20], thus covering a wide range of possibilities.

How do LLMs work? The current LLM models run on a transformer architecture that depends on a mechanism called *attention*. It was first introduced in the article by Vaswani *et al.* and showed an innovative model for language translation tasks [21]. BERT [22] was one of the first so-called Pretrained Language Models (PLMs) that was developed on this transformer architecture. Scaling the model and data size leads to an improved performance in solving complex work [23]. Those larger PLMs are then called LLMs.

The transformer model estimates the likelihood of sequences of words within natural language. Unlike other approaches developed before, the fact that a transformer can process all words in a sequence simultaneously, rather than one word at a time, is attributed to its highly parallelizable architecture with self-attention mechanisms. Transformers learn which words are most relevant to each other, regardless of their distance from each other in the text. This enables the model to infer or generate subsequent or missing tokens within a given context [24].

Training of LLMs occurs in two stages, pretraining and fine-tuning. During pretraining, the model is trained on a vast amount of data, including websites, books, and articles. During training, the model learns to predict the next word within a given sequence. This process enables the model to develop an understanding of language patterns, factual knowledge about the world, reasoning abilities, and other skills. The model adjusts billions of parameters to minimize prediction errors in this enormous dataset. Within the fine-tuning stage, the model is often optimized for specific datasets or tasks. It learns to respond to specific tasks that are helpful to user requests and to act with safety features that align with human values [25].

However, the model lacks a proper understanding, in the human sense, of the output it generates. Instead, it is performing pattern matching based on statistical relationships learned from its training data. This process can produce remarkably coherent and helpful responses across a wide range of topics. However, there are challenges in working with LLMs. They match patterns based on their trained data; they are not knowledge machines. It does not look up facts; instead, it generates the most statistically likely continuation based on the patterns it learned. And here lies an architectural issue.

What are Hallucinations? Hallucination as a term has been adopted in the context of Natural Language Generation (NLG), which describes an issue where the model generates “unfaithful or nonsensical text”[26]. In the scenario of LLMs, it arises when models generate text with an artificial certainty or misleading confidence that includes facts that are false, misleading, or entirely fabricated [27]. That leads to several issues, such as the production use of LLMs.

Two different types of hallucination have been classified in NLG: Intrinsic and extrinsic hallucination [28]. Intrinsic hallucination deals with the issue that the generated output does not correspond to the given source text and is simply false [29]. Extrinsic hallucination is referred to when the made claim can be neither supported nor denied. In the context of text summarization, it could be the issue that a claim is made up and there is no connecting fact in the source document, such as inventing a new source [30].

However, within the context of LLMs, the meaning of the term hallucination shifts mainly to extrinsic hallucinations because there is no longer only one source document; moreover, an LLM constantly uses world knowledge where deviations from training data might occur more frequently [26]. Other researchers differentiate in a more nuanced way. Zhang *et al.* distinguish between input-conflicting errors, context-conflicting hallucinations, and fact-conflicting errors [31]. Huang *et al.* separate hallucination into factuality hallucination

and faithfulness hallucination. Input-conflicting hallucination refers to erroneous behavior where LLMs produces text that is not in line with the user input, context-conflicting hallucination describes conflicts with the generated content preceding, and the LLM loses sight of the context. When the LLM generates content that conflicts with verified facts and information from the real world, it is called a fact-conflicting hallucination [31]. Huang *et al.* distinguish even more in factual contradiction (against the world knowledge) and factual fabrication (proposing new facts that do not exist). In addition, they differentiate between instruction inconsistency, context inconsistency, and logical inconsistency [32]. We agree on their separation, and in this work, we mainly focus on the factual contradiction hallucination.

Why are Hallucinations hard to eliminate? Hallucinations are difficult to eliminate based on the functionality of how LLMs work. Different sources of hallucination errors result from a complex interplay of factors throughout their development. Those errors can be classified into data, training, and inference issues [32].

Data-related Hallucination This type of hallucination primarily originates from three sources: misinformation and biases in training data, knowledge boundaries, and low-quality alignment data.

The ever-increasing amount of data for pretraining makes it challenging to maintain consistent data quality, which leads to misinformation in the training data [33]. LLMs often ingest this flawed online content, which includes fake news, and then reproduce it as imitative falsehoods [32]. The easy way to create textual data with LLMs does not support this issue, instead making it even harder. Biases, such as the association of certain professions with a particular gender [34] or a nationality [35], often persist due to the widespread biased perspectives in online texts [36].

Another issue is the knowledge boundary. Despite their vast knowledge, LLMs cannot retain everything, especially scarce and rarely mentioned facts [37], [38]. They also lack access to live updates or copyrighted materials, making them prone to hallucinations when asked about up-to-date [39] or restricted or specialized topics. This illustrates a clear boundary in temporal data that is frequently updated, highlighting a knowledge gap in topics that are not publicly accessible [40]. Furthermore, inferior alignment data, where instructional fine-tuning exceeds the model’s knowledge scope, can unintentionally increase hallucinations by pushing the model to generate plausible-sounding but inaccurate responses [41].

Training-related Hallucination Different phases of training are to blame for various hallucination errors. The pretraining phase employs left-to-right token prediction, which sometimes struggles with long-range dependencies and context coherence, particularly due to the limitations of soft attention [42]. This can lead to subtle factual errors. A significant factor is exposure bias, a discrepancy between how models are trained with

a correct context and how they generate text one token at a time, often leading to error compounding [43], [44].

During Supervised Fine-Tuning (SFT), models may overfit to instructional data and be forced to answer questions beyond their trained capabilities, rather than expressing uncertainty [41], [45]. This leads to fabrication when those models are faced with queries that go beyond their knowledge in their responses. Similarly, reinforcement learning with human feedback can lead to hallucinations due to a tendency to agree with user opinions, even when they contradict internal model knowledge. This is called sycophancy, and the models are even aware of the misstatement [46].

Inference-related Hallucination Hallucinations can also occur during text generation itself. Stochastic decoding strategies, which add randomness for creativity, often trade off accuracy [28]. Increasing the sampling temperature increases the chance of selecting rare, potentially incorrect tokens. Models may also exhibit overconfidence, prioritizing fluency over factual correctness, especially in long-form responses where earlier errors influence later content [47]. Finally, reasoning failures, such as the *reversal curse*, which means that models fail to deduce the fact that if A is B, then also B is A, reveal weaknesses in logical inference, even when the relevant facts are known [48].

2.1.1 Hallucination Detection

Hallucinations are a systemic issue embedded in the full life cycle of LLM development, as we have shown before. Addressing them requires more than post hoc corrections. It requires better data curation, architectural improvements, robust training techniques, and better inference mechanisms. This is why techniques like Retrieval Augmented Generation (RAG) and fact-checking systems are being developed. They try to ground the model's pattern-matching abilities with access to verified, real-time information sources. Existing ways to detect and mitigate hallucination errors can be classified according to the nature of the hallucination being addressed. Factuality-based detection aims to identify inaccuracies by evaluating the factual correctness of the model's output against verified external knowledge. Faithfulness-based detection evaluates the degree to which the model's responses remain consistent and grounded in the input context provided [32]. In this work, we focus on factuality, which is why we only show current strategies of this type.

Factuality-based Hallucination Detection This type of hallucination detection involves analyzing the extent to which LLM-generated information is consistent with established real-world knowledge. Most strategies focus either on fact checking, which deals with the process of verifying the generated output against external, well-established facts, or on uncertainty estimations, which refers to the process of identifying factual inconsistencies in model outputs by analyzing internal indicators of the model's confidence [32]. The usual steps consist of first detecting a factual claim and second verifying the

claim against other established truthful sources. We will investigate those steps further in Section 2.2.

Fact-checking was a well-studied field of research even before the arrival of LLMs. We differentiate between fact-checking approaches within the context of NLP and LLMs. It can be used through external sources, but internal verification is also plausible as LLMs have a tremendous amount of knowledge in their parameters.

Chern *et al.* proposed a framework to detect factual inaccuracies utilizing a suite of **external** tools designed for evidence collection. That enables catching errors in the category of math problems, code generation, or knowledge-based Question Answering (QA) [49]. Huo *et al.* explores verifying LLM-generated answers by querying a corpus with both the question and the response, using expanded queries to ensure relevant evidence and address the topic drift [50]. Min *et al.* presented FACTSCORE. This metric evaluates long-form text by checking which proportion of atomic facts are supported by reliable sources, highlighting its effectiveness through human evaluation of model-generated biographies [51]. Another approach by Wei *et al.* utilized Google search to either support or refute atomic facts of generated long-form texts [52]. Ni *et al.* introduced AFaCTA, a framework that supports the annotation of factual claims with the help of LLMs fine-tuned on three distinct reasoning paths, which are subsequently aggregated. Another approach by Wang *et al.* proposes Factcheck-Bench, a holistic framework that classifies facts, opinions, and other statements on different levels of granularity, including those that do not relate to meaningful claims [54]. Finally, Song *et al.* created VeriScore, which combines claim classification, decontextualization, and verification in one prompt by only extracting verifiable claims [55].

However, the **internal** LLM’s knowledge can also be employed to fact-check the generated answers. An interesting approach was suggested by Dhuliawala *et al.* where a model fact-checks itself by first drafting a response, then creating questions that verify the answer, then answering those questions in individual steps, and finally delivering an ultimate response [56].

2.1.2 Hallucination Mitigation

In this subsection, we provide a brief overview of general hallucination mitigation approaches, including those implemented before training, after training, and during the use of LLMs. There are also techniques for addressing training- and inference-related hallucination; however, we mainly focus on data-related hallucination mitigation approaches.

Data Filtering describes a mitigation method that typically happens before training. This involves filtering and cleaning the training data hallucination sources. That includes a judicious choice of excellent pre-training data derived from credible origins. By doing so, we can maintain data accuracy and simultaneously reduce the impact of social biases

[32]. The recognition of this was already evident during the early development of GPT models [57].

Model Editing In this methodology, a pre-trained model acquires supplementary knowledge, thereby enhancing its performance and capabilities. In a locate-then-edit approach, first, false model parameters are located, and afterwards are updated to improve the model behavior [58]. Meta-learning constitutes an alternative approach to model editing, whereby an auxiliary model is trained to forecast improved weight data for the original model [59].

Retrieval Augmented Generation RAG is a technique that follows an information retrieval pipeline, where relevant information is first gathered [60] and then the response is generated, incorporating information from the fetched sources. It is therefore possible to bridge the knowledge gap described beforehand and improve the performance of the LLM [61]. The classification of retrieval processes into categories such as one-time retrieval, iterative retrieval, and post-hoc retrieval [32] is based on the frequency and nature of the retrieval operations conducted. While one-time retrieval focuses on performing information retrieval a single time before the generation phase, an iterative approach is intended for information retrieval in contexts that require complex reasoning [62] or during the creation of extended responses [63]. Conversely, the post-hoc retrieval strategy is dedicated to refining text that has already been generated, enhancing accuracy and resolving inconsistencies with the available evidence [64].

2.2 Natural Language Processing Framework for Automated Fact-Checking

In this section, we aim to elucidate comprehensive details within the realm of fact-checking. This section is organized according to the NLP framework for fact-checking outlined by Guo *et al.*[65]. Initially, we explore claim detection and verification, encompassing the retrieval of evidence from news articles and knowledge graphs. Subsequently, we examine the subjects of stance prediction and justification production.

2.2.1 Claim Detection

The term “claim detection” is common in various research areas, specifying different concepts. We focus on claim detection solely within the domain of fact-checking. Claim detection is typically the initial step in the fact-checking process, focusing on the detection of check-worthy claims [66]. In their study on detecting check-worthy factual claims in presidential debates, Hassan *et al.* defined check-worthy sentences as those facts that the general public is concerned about having the information about whether it is true or false [67]. The general results of claim detection could be a Boolean type or an importance-sorted ranking of statements, while prioritized statements ranked higher [68],

[69]. Various models for claim detection exist, each employing different approaches [22], [70], which can also be used in combination [68].

2.2.2 Claim Verification

The process of claim verification involves several key components, namely evidence retrieval, stance prediction, and justification production. To facilitate more precise readability, the elements of stance prediction and justification production will be discussed in subsequent subsections.

Evidence Retrieval The next step in fact-checking involves retrieving evidence once a checkable claim has been identified. This can be done in several ways. We want to focus on evidence retrieval based on news articles and knowledge graphs. This is not only relevant for building a stance but also essential for building a thorough justification that makes a user trust the fact-check [65].

Evidence Retrieval based on News Articles The initial challenge lies in finding the most relevant news articles, followed by the challenge of extracting the correct passage within those articles. Samarinas *et al.* have addressed the issue of evidence retrieval and developed a model for automatic fact checking based on BERT [22]. It is capable of retrieving the most relevant passage, selecting the correct sentence, and performing the final entailment classification [71]. Another approach is contributed by Ferreira *et al.* who retrieved, summarized, and labeled news articles to predict the stance of a given claim [72]. For evaluation purposes, multiple data sets are available, including those from Augenstein *et al.* [73] or Thorne *et al.* [66].

Evidence Retrieval based on Knowledge Graphs A KG is a structured representation of data that models entities along with their relationships and attributes in the form of a graph [74]. Unlike other tabular databases, graph databases enhance the interconnection of different data points, providing a more detailed view of the context or significance of information.

The goal of this section is to acquire evidence to support a claim within a knowledge graph. While RAG employs knowledge graphs in the form of embeddings to improve responses generated by LLMs, this context focuses on formulating actual queries to obtain evidence for a claim. The construction of query templates grounded in natural language questions is integral to this area of research [75]. Abujabal *et al.* proposed a system designed to automatically generate templates for more complex questions [76]. Concurrently, ongoing research exists aimed at leveraging LLMs to formulate database queries, wherein the LLM functions as an interface to the database [77], [78]. However, the challenge of evaluating the relevance of the actual claim remains a significant concern.

Liu *et al.* proposed an approach in which they train a judge model based on triples from a knowledge graph to evaluate the factuality of LLMs [79]. However, there is limited

research on automatic query generation based solely on LLMs; in cases where it exists, it yields unsatisfactory results [80].

2.2.3 Reranking

This component is not invariably incorporated into fact-checking methodologies, as an alternative approach could employ an unranked list of evidentiary pieces. However, several opportunities exist to evaluate factors that contribute to a final ranking score signaling its relevance to the given claim. BERTScore uses a pre-trained BERT [22] model and its generated contextual embeddings to represent tokens in candidate and reference sentences. Those tokens capture the specific meaning of a token within its sentence, which allows for a measure of similarity [81]. Zhang *et al.* leverages a transformer encoder to rerank documents. Through its self-attention mechanism, this encoder models hierarchical and relational dynamics between documents, ultimately predicting final relevance scores [82]. An alternative methodology involves prioritizing evidence based on recency, thereby attributing greater significance to more recent evidence. [83]

2.2.4 Stance Prediction

The objective of stance or verdict prediction is to assess the veracity of a specified claim utilizing multiple evidential sources, which are procured in relation to the claim under consideration. Ultimately, this task constitutes a classification problem, wherein the outcome may manifest as either a binary classification, such as true or false [84], or as a more nuanced classification involving multiple categories that reflect varying levels of veracity [73]. A more indicative way would be to say that a claim is supported or refuted, rather than using `true` and `false`. This reflects that the decision was made based on more than just evidence. The algorithms upon which stance prediction is predicated generally do not achieve a 100% accuracy rate, thereby indicating the appropriateness of employing language that is less assertive than that found in `true` and `false` [85].

2.2.5 Justification Production

Once a fact-checking process is done, it is crucial to explain the rationale behind the decision regarding a stance and, therefore, convince readers about the analysis of the evidence [86]. Automatic fact-checking frameworks often appear as a black box with internal processes that are not explained to users; therefore, it is suggested that provided explanations address this issue [87]. There are several ways to give a justification to users. One approach involves marking the most relevant segment within a piece of evidence based on a scoring function that assigns weights [88]. Gad-Elrab *et al.* implemented a system where the stance is calculated in a logic-based system, and the justification is derived from that calculation [89]. Lastly, a justification can be a summary of different explanations from various fact-checking segments [90]. A combination could also be possible.

The Claim Detection and Verification Framework

In this chapter, we present our approach for claim detection and verification, following an NLP framework [65] that utilizes current state-of-the-art LLM technologies. It addresses RQ1 by demonstrating a method for effectively validating LLM-created knowledge using knowledge graphs and news articles within a multi-agent framework. Figure 3.1 provides an overview of the complex structure, which is explained in further detail in the following sections.

3.1 Pipeline Design

The pipeline begins with a general chatbot interface, as seen in products such as ChatGPT [1], Claude [91], or LeChat [92]. This interface serves as the entry point for user interaction with the chatbot, featuring integrated claim verification. We have chosen a straightforward interface with an area to phrase a message (Figure 3.2) and an area that displays the chat history. Other chat interfaces include model choice or file selection; we choose to keep it as minimal as possible, ensuring that we provide only the relevant features for this research. We utilize the general-purpose OpenAI GPT4.1 [93] model to process the user request. The architecture uses two separate endpoints for claim detection and claim verification. Once an answer is generated, all factual claims are extracted by the claim detection endpoint. All detected claims are then processed in the claim verification section. Here, the claims are verified against news sources and a knowledge graph. Eventually, the results of the claim verification request are sent back, including a position and justification.

In the following subsections, we describe the various steps involved in detecting and verifying claims, including evidence retrieval from news sources and the knowledge graph, stance prediction, and justification production.

3. THE CLAIM DETECTION AND VERIFICATION FRAMEWORK

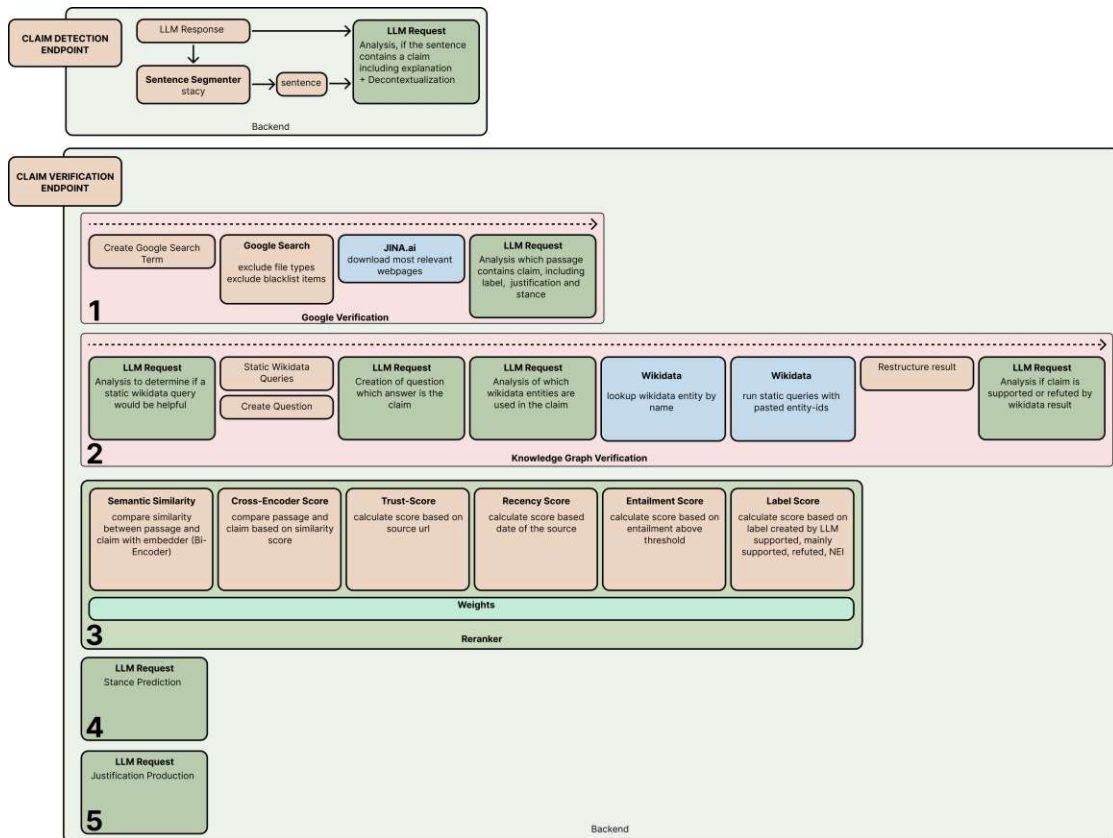


Figure 3.1: A Complete Overview of the Developed Framework

We also want to highlight that the main scientific contribution lies in the claim detection and verification framework; however, we have also implemented a frontend to present the results more clearly and explain the use case of the developed pipeline.

3.2 LLM Guidance

Before delving deeper into the various steps of claim detection and verification, we first outline the approach we took during development and the LLM tooling we employed. We combine an NLP framework for automated fact-checking [65] with state-of-the-art LLM technology. We used Prompt Chaining to break tasks into smaller subtasks and controlled generation to receive structured generated output from the LLM. Within the several requests, we experimented with Zero-Shot and Few-Shot Prompting. Those techniques improved the general results of this framework.

AI Chatbot with Claim Detection and Verification

This chatbot detects claims in the AI's responses, highlights them, and provides sources for verification.

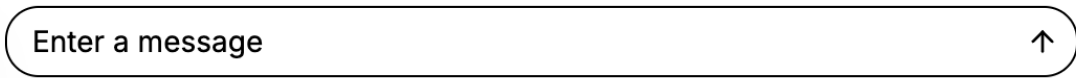


Figure 3.2: A Screenshot of the Chat Interface

3.2.1 LLM-Requests

Within this framework, several requests with different prompts are sent to the LLM. In this subsection, we provide a brief overview of these. All prompts are outlined within the appendix.

- **Claim Detection:** Detects, extracts and decontextualizes claims (Figure 7.1)
- **Claim Verification:**
 - Evidence Retrieval: Check website content for supporting or refuting passages (Figure 7.2)
 - Evidence Retrieval: Decision if Wikidata would be helpful (Figure 7.3)
 - KG Retrieval: Generate a question based on the claim (Figure 7.4)
 - KG Retrieval: Extract entities from claim (Figure 7.5)
 - KG Retrieval: Check if KG statement supports / refutes / not enough evidence (Figure 7.6)
 - Stance Prediction: Decision of attitude towards evidence (Figure 7.7)
 - Justification Production: Produce a justification for the user (Figure 7.8)

3.2.2 Prompting Paradigms and Response Control

In this subsection, we briefly summarize and explain the prompting strategies that we used and how we controlled the response generation process.

Zero Shot Prompting The ability of LLMs to execute tasks without requiring illustrative examples is attributed to its extensive training on immense quantities of data, coupled with its optimization for instruction adherence. This phenomenon, known as zero-shot prompting, involves supplying the model with explicit instructions in the absence of any corresponding training examples. The model utilizes its pre-trained knowledge to analyze and address the task exclusively based on the given prompt [94].

Role Prompting The assignment of roles, such as a food critic or technician, to an AI model is referred to as role prompting [95]. With introductions such as “You are ...” at the beginning of the prompt, it is possible to adapt the tone, style, or accuracy of the generated response. Kong *et al.* also have shown that role prompting improves performance in specific tasks, such as reasoning [96].

Prompt Chaining This technique involves splitting tasks into smaller subtasks, each of which is covered by a separate request. Once a subtask is completed, the generated response is then fed as input to the subsequent request. Because the prompts are tied together, it is referred to as prompt chaining. It is used for very complex tasks where the LLM cannot handle all details in a single prompt. This increases transparency, and debugging is also simplified. It also improves the control over the output. By following smaller steps, it is possible to fine-tune the responses before the generated output is passed on to the following prompt.

Chain Of Thought The Chain-of-Thought (CoT) prompting technique refers to the paradigm of providing the LLM a pattern of ordered reasoning steps within the prompt. It enables the model to break down a complex task into several simplified subtasks. Then each subtask is solved first before generating the final answer to the complex task [97]. This can be achieved by integrating it with a few-shot prompt or by providing ordered stages with subtasks that need to be completed.

Structured Output This feature enables developers to receive a structured response from a natural and unstructured request. OpenAI [98] released this feature in August 2024, initially as JavaScript Object Notation (JSON) mode, and has since further improved it. Other LLM providers such as Google offer a similar feature called controlled generation [99]. This feature is limited and only available for newer models from different providers. By providing a JSON schema, the model output conforms to the provided schema. The earlier developed JSON-mode was not strict in conformity; it only ensured that the output is a valid JSON-output, but it could not ensure that the content within the provided schema is valid. However, the current structured output feature delivers a consistent output with the strict mode enabled [98].

Pokrass implemented a methodology to enhance the reliability of model outputs, adhering to JSON Schema specifications. A deterministic approach was employed, utilizing constrained decoding to guarantee 100% reliability. Constrained decoding restricts

models to generate only schema-compliant tokens, rather than any arbitrary token within their vocabulary. This method involves transforming an JSON Schema into a context-free grammar, which defines valid token sequences. Continuously refreshing the pool of permissible tokens during each phase of output generation significantly reduces the chances of models generating incorrect outputs. Although preprocessing the schema results in initial latency, this configuration enables efficient sampling with minimal continued latency impact [98].

The provided 100% reliability of the structured output feature is a core enabler of the developed pipeline. By trusting that an LLM output follows our provided JSON schema, we can integrate such LLM calls into regular data pipelines with logic-based decision-making processes. The framework can further process the structured output data.

In the structured output of OpenAI, different parameters can be set. Developers can create a schema tailored to their specific needs. However, the parameters must be set in the correct order. Xie found out that the reasoning order is essential and can drastically change the result. In the example of a structured output of the parameters `{answer, reasoning}`, better results can be achieved if `reasoning` is the first parameter, `answer` is the second parameter. The issue lies in the functionality of how output tokens are generated. The model is unable to foresee subsequent output text while producing preceding text. In the provided example, therefore, it does not make sense first to generate an answer and then reason about it [100]–[102]. In Listing 3.1, an example is presented where we observed the outcomes of incorrectly ordered parameters in the structured output.

We reference a structured output schema in Listing 4.2.

3.3 Claim Detection

In this section, we explain the claim definition we follow and how the principles of claim detection and the process of building the prompt are determined.

What is a claim? In research, different definitions of a claim are present [103]. As reported by Lippi *et al.*, three distinct categories of claims exist: *epistemic* claims as such about facts and beliefs, *practical* ones which concern actions, possible alternatives, and their consequences, and *moral* claims which involve values, ethical considerations, or personal preferences [104]. Epistemic claims can also be referred to as factual claims, which is what we are interested in. Hassan *et al.* uses the concept of public interest to further differentiate between an unimportant factual sentence that is not check-worthy and sentences that are relevant to the general public. Unimportant factual sentences include factual claims but are not relevant to the general public or lack newsworthiness, such as “Last week we had dinner at grandma’s home.”. Check-worthy factual sentences involve factual claims that are of public interest, such as “Lip-Bu Tan is the CEO of Intel” [67]. However, some facts that might be interesting to the general public cannot

be verified, and vice versa. Consequently, Ni *et al.* proposed a novel definition for factual claims, which aims to minimize subjective assessments of check-worthiness while enhancing the emphasis on objective verifiability: “A factual claim is a statement that explicitly presents some verifiable facts.” [53]. Konstantinovskiy *et al.* also concentrated less on subjective check-worthiness and importance, and more on actually checkable claims: a claim is “an assertion about the world that can be checked”. [105] We combine both approaches and focus on verifiable and actually checkable facts within the generated responses.

3.3.1 Main Principles

We take advantage of several principles in the area of LLMs and claim extraction. Sentences are typically best understood within the context of surrounding sentences or even a whole paragraph. Therefore, we analyze each sentence along with its context. However, to further process the sentence and claim, we need to understand the claim without context. We have implemented a decontextualization procedure in which all pronouns are replaced by the corresponding entities or referents to which they refer. Other approaches depending on atomicity have also been researched [106]. They show that a sentence can be broken down into several subsentences, each holding exactly one claim. However, we specifically did not consider following this approach, as the research refuted that it improves the claim extraction process [107].

Claim Detection Prompt The claim detection process involves a single prompt that combines multiple tasks into a single one. By splitting claim detection and claim verification into several tasks, we apply chain prompting to concentrate on the detection part first. We empirically developed the prompt and tested several prompting approaches. In this prompt, we start with a simple role prompt introduction by assigning the model a fact-checker persona. The objective is to configure the model to function as a highly precise fact-checker. Subsequently, employing a zero-shot methodology, we provide a definition of a factual claim, along with the requisite criteria that must be fulfilled. Building upon the proposed method by AFaCTA [53], we developed a CoT [97] claim detection prompt that delineates the following instructions in three distinct reasoning steps: (1) evaluate the sentences, (2) pinpoint the specific segment containing the claim, and (3) reason about the verifiability. In the final section, we give classification and decontextualization instructions. We tried to improve the prompt by integrating an approach of adding “Let’s think step by step.” originally introduced by Kojima *et al.* [108], later integrated into a CoT process by Zhang *et al.* [109] but could not improve it further, instead got worse results.

Structured Output We use structured output in this request in the correct order to ensure that a JSON object is returned as a response. The choice of parameters has been made based on the needs of the pipeline and the frontend that should reveal the detected prompts.

3.4 Claim Verification

In this section, we outline our approach to verifying claims. That includes both evidence retrieval via a search engine and the use of a knowledge graph. We adopted a strategy of combining both sources of knowledge to enhance the retrieval outcome. First, we approach Google search for several sources, and in the next step, we query the knowledge graph with static queries. Finally, we merge all the collected information and rank it based on several factors. Figure 3.3 provides an abstract overview of the processes involved in verifying the claim. Each detected claim triggers a new pipeline. As a result, a list of several sources coming from Google search and a knowledge graph is then passed on to the next step, stance prediction.

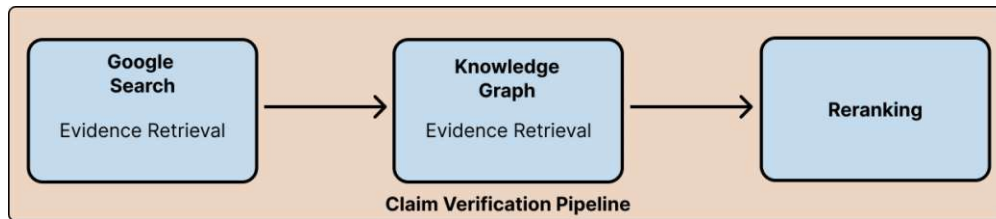


Figure 3.3: Procedure of the Claim Verification

3.4.1 Evidence Retrieval with Google Search

We utilize Google Search to receive external sources to refute or support a detected claim. After creating a suitable search term, every source is downloaded and sent via a prompt to an LLM to analyze the website’s content and extract the most crucial parts.

Creation of the Google Search Term In this step, we generate a query from the given claim to search for relevant websites using a search engine later. We do not use the entire claim; instead, we follow [110] and [111], using a shortened query that eliminates unnecessary words. We keep verbs, adjectives, and nouns, including nouns that refer to entities such as names or organizations. We skip all other words that belong to the remaining word categories. From the claim “The President and Chief Executive Officer of Amazon is Andy Jassy” results the query `Amazon Andy Jassy President Chief Executive Officer`.

Query Google Search Subsequently, we employed the formulated query to retrieve the ten most relevant search results. We excluded specific pages that could not provide any value during the evaluation, such as those of social media platforms. We use Google Search in our framework because it has the largest market share. Based on that fact, we assume that Google provides the best search results.

Downloading Sources The framework autonomously acquires the content of each website. The use of Jina.ai facilitates the extraction of web content and its subsequent

conversion to LLM-compliant markdown files through their specialized LLM [112]. Unlike alternative web scraping methodologies, there are no imposed limitations on website accessibility.

Search Engine Passage Prompt In the next step, the markdown file with the website content is sent to an LLM to verify the given claim based on the website content. In this prompt, we first use Role Prompting to set the scene and tone of the response. We provide detailed instructions to find supporting or refuting passages, then justify why a passage is supporting or refuting the claim, and then give a label of supported or refuted. If there is no passage on the website, the prompt instructs the LLM to leave the passage empty, give a detailed justification why no claim could be found, and give it the label `not_enough_evidence`. We also utilize structured output to convey the generated information to the next step. First, we order the parameters as follows: (`label`, `link`, `passage`, `justification`); an example is shown in Listing 3.1. The example demonstrates that the LLM initially refutes the claim; however, in its justification, it ultimately concludes that it actually supports the statement after a considerable amount of back-and-forth. Fixing the order of the parameters and changing it to (`link`, `passage`, `justification`, `label`) further improved the results. We explain the phenomenon in Section 3.2.2.

3.1

Listing 3.1: A Bad Example of Sorting the Structured Output Parameters

```
1 {
2   "claim": "Ann Dennison joined Cigna in 2024",
3   "sources": [
4     {
5       "label": "refuted",
6       "link": "https://www.insurancebusinessmag.com/us/news/life-insurance/cigna-
group-announces-leadership-changes-528538.aspx",
7       "domain": "insurancebusinessmag.com",
8       "found_via": "google search",
9       "date": "2025-03-14T00:00:00",
10      "passage": "Ann Dennison has been named executive vice president and CFO of
Cigna Group. In this capacity, Dennison will have responsibility for all
enterprise financial operations and functions. She will also join the company\
u2019s enterprise leadership team and will report to Cordani. Prior to joining
Cigna in early 2024 as deputy CFO, Dennison was executive vice president and CFO
at Nasdaq.",
11      "justification": "The passage states that Ann Dennison joined Cigna in
early 2024, not 2024, which is the year mentioned in the claim. Therefore, the
claim that she joined in 2024 is incorrect based on the information provided, as
she joined in early 2024, which is still within the year 2024, but the claim
implies a specific date or that she joined in the year 2024 as a whole, which is
supported. However, the claim's phrasing suggests she joined at some point in 2024,
which is consistent with 'early 2024'. Since the claim is about the year, and she
joined in early 2024, the claim is supported by the content. But if the claim
implies a specific date or that she joined after 2024 started, then it would be
refuted. Given the information, the most accurate label is 'supported' as she
joined in early 2024, which is within the year 2024.",
12      "embedding_score": 0.7999521493911743,
13      "cross_encoder_score": 0.9999219179153442,
14      "entailment_score": 0,
```

```

15         "trust_score": 0.5,
16         "date_score": 0.7,
17         "final_score": 0.514983502626419
18     }
19 ]
20 }
```

Finally, we combine all results into an evidence list.

3.4.2 Evidence Retrieval with Knowledge Graphs

Evidence retrieval on the knowledge graph requires more intermediate steps than the search engine retrieval approach. It is not trivial to automatically query an KG and further integrate the results with the information already gathered from the search engine. Retrieving information from a KG needs the creation of a correct semantic and syntactical query. [79]

We break down this challenge into several subtasks and provide a brief overview of the steps involved in the process.

1. We prompt LLM to give a response on whether one of the static queries created would help to verify the provided claim.
2. A prompt to generate a question whose answer is the exact claim.
3. A prompt to identify all relevant entities of the prompt, such as names or organizations. Those are needed to construct the SPARQL query.
4. Find out the Wikidata IDs of the received entities.
5. Replace the placeholders in static queries with Wikidata IDs.
6. Query Wikidata.
7. Prompt the LLM to verbalize the structured data and reason if the results support or refute the claim.

Would a KG be helpful for verifying the claim? Not all claims can be answered with a knowledge graph. During development, we conducted tests to automatically create suitable queries with LLM technology that could verify the claim. Several prompting techniques were unable to yield any suitable outcomes, as other research has also reached the same conclusion. Wikidata, for example, requires corresponding IDs for subjects, predicates, and objects. We had issues with hallucination of predicates or wrong IDs of entities. The language model generated non-existent IDs and entities.

Therefore, we changed our approach from dynamically creating queries with LLMs to statically prepared queries. We limited the approach to four static queries that we provide with placeholders. An LLM decides how to fill the placeholders with the

correct entities. Other issues with the knowledge graph exist with inconsistent data modeling. For example, to describe a parent-child-company relationship, it is possible to use the predicate `parent organization` and `child organization` predicate, or the `owned by` and `owner of` predicate. To cover such a relationship, it is essential to be aware of these different possibilities. We use the following four queries to discover facts about high-ranking managers:

- List all employers of a person, including the dates of start and end of employment.
- List all CEOs of a company, including the dates of start and end.
- List all companies where a person held the position of the CEO.
- List potential parent companies and company owners to discover holding structures.

We used a simple prompt with role and zero-shot prompts to determine which query could be helpful in the given claim. With a structured output, we could proceed with the chosen specific queries.

Correct Entities and Query Generation Subsequently, we requested the LLM to formulate a question for which the provided claim would serve as the answer. That helps identify the necessary entities for the next step. Role prompting and structured output enabled this task.

In the next step, we continue by providing the generated question and the claim to LLM. We use role prompting to set the tone for an expert in SPARQL query generation. However, we explicitly describe that the LLM should return only entities, such as companies and persons, as they would appear in Wikidata. We provide two examples of the few-shot prompting approach to enhance the response. With the correct IDs provided by Wikidata of the returned entities, it is now possible to query Wikidata by replacing the correct IDs with the placeholders of the static queries.

The information received from Wikidata is then fed into an LLM to convert the structured data into natural language. This is needed to combine the search results from the search engine and information from the KG. Moreover, the LLM is instructed, based on the structured data from the knowledge graph, whether it supports or refutes the claim or if there is not enough evidence at all.

3.4.3 Reranking

In the preceding phases, the framework systematically collected data that either corroborate or refute the claim. In the subsequent stage, the framework systematically arranges the retrieved information based on its relevance. This prioritization is instrumental for the tasks of stance detection and justification generation. Furthermore, the source considered most relevant is regarded as the authoritative source for the user in the interface. We

developed a Reranking algorithm that comprises several calculated scores, which are then combined into a final ranking score. All values can be between 0 and 1, where one is rated as very relevant and zero is not relevant. We use the following calculations to get as close as possible to the full truth:

- **Semantic Similarity Score:** The semantic similarity between a claim and a passage determines the degree to which two sentences convey an equivalent meaning. In our case, this is done with word embeddings. The closer the embeddings are, the higher the score.
- **Cross Encoder Score:** That score is similar to the Semantic Similarity; however, the cross-encoder is an PLM that is trained on a large amount of text. It can understand the relationships and the dependence between the claim and the passage.
- **Trust Score:** In this score, we calculate how trustworthy the source of information is. Of course, this can only be done in a limited way. We consider the following parameters. We believe that, in the context of C-level executives, the company's own website provides the most accurate information on which manager positions are currently occupied by whom. Consequently, sources originating from the same second-level domain are accorded a higher ranking. Additionally, we rank higher on pages that also have specific keywords integrated in the URL. That could be `ir`, or `investor-relations`, as we know that many companies have a space dedicated to that topic on their website.
- **Recency Score:** Recent sources of information are considered to provide greater value, particularly in the domain of large-scale corporate management, where circumstances may evolve over time. Consequently, up-to-date data is considered more pertinent than outdated references. Priority is given to current information rather than outdated information.
- **Label Score:** In this specific evaluation, the framework factors in the stance attributed to the evidence. A definitive classification, such as `supported` or `refuted`, is awarded a higher score compared to an ambiguous label, such as `not_enough_evidence`
- **Entailment Score:** This is a classification approach. Entailment means that if one statement is true, then the other statement is also true. One statement is a consequence of the other. In this case, a cross-encoder determines whether the passage entails the claim with confidence above a specified threshold. If the value is above the threshold, we count the passage as an entailment and otherwise as a contradiction.

Finally, we apply different weights to combine the mentioned scores into a final score. We empirically derive the weights from the development process and assign them the values shown in Table 3.1.

Table 3.1: The Used Weights For Different Scores in the Reranking Process.

Semantic Similarity	Cross-Encoder	Trust	Label	Recency	Entailment
0.1	0.15	0.15	0.1	0.3	0.2

We provide the claim and all evidence, drawn from both the knowledge graph and the search engine, presented as a single list.

3.5 Stance Prediction

Once the framework has reranked the evidence accordingly, we proceed by producing a stance that considers all the evidence produced. We use a strategy based on an LLM request. Within this request, we provide the claim and all evidence, drawn from both the knowledge graph and search engine, presented as a single list. The prompt utilizes a role prompt, setting the tone as a stance prediction assistant. We provide the LLM with the opportunity to assign four different final labels: `supported`, `refuted`, `conflicting_evidence`, and `not_enough_evidence`. Moreover, we instruct the LLM to adhere to the following decision logic: (1) If the evidence only supports the claim, label it `supported`. (2) If the evidence only refutes the claim, label it `refuted`. (3) If the evidence includes supporting and refuting elements, label the claim as `conflicting_evidence`. (4) If the evidence mainly supports or refutes the claim, take care of the date of the evidence and label it accordingly. Newer evidence should be prioritized. (5) If the evidence is both supporting and refuting, but the supporting evidence is more recent or stronger, label it `supported`; and vice versa. (6) If there is no clear supporting or refuting evidence, or the evidence is too weak or ambiguous, label it as `not enough evidence`.

Finally, we instruct the LLM to explain the choice made and the label itself in a structured output. We outline the prompt in Figure 7.7

3.6 Justification Production

In the final step of the framework, a justification for the chosen stance is generated. In this case, we also utilize an LLM to support the chosen stance. We therefore apply an approach that summarizes the sources and stances predicted and reranked, building on the idea of Atanasova *et al.*[90]. The justification helps the user understand why a particular label is assigned to the claim. We provide the evidence list, the stance, the stance explanation, and the claim to the LLM and set the role of a justification production assistant. We instruct the LLM only to use the provided evidence and to produce a short and concise statement. Figure 7.8 shows the full prompt.

Implementation

In this chapter, we discuss the implementation of the whole claim detection and verification pipeline. We structure this chapter by following our development approach. In the first section, we define our architecture, including all technical details. We describe our development approach by following the DSF. We clarify the reasons for the chosen LLMs and KG in the final prototype. We also clarify how we created the dataset to evaluate the framework. In the second section, we dive deeper into the claim detection process. The final section outlines the procedure for claim verification.

4.1 Architecture and Technology

A prevailing trend in software development is the decoupling of front- and backend architectures. In this paradigm, communication between the two layers primarily relies on data exchange in JSON format. We adopted this principle for the separation of concerns. The frontend is responsible for building and providing the user interface. At the same time, the backend handles computation and communication with various Application Programming Interfaces (APIs) such as the LLM or the KG. This communication is implemented with the well-established architectural style of RESTful API [113]. The backend operates a web server that the frontend can access via two API endpoints. We utilize two main programming languages: Python for the backend and TypeScript for the frontend. The following subsections provide a detailed explanation of both components.

4.1.1 Frontend

The frontend is the visual component that users interact with. We selected NextJS as our frontend framework¹. Next.js is a framework built on React that enables the creation of comprehensive full-stack web applications. It relies on Node.js as its runtime

¹<https://nextjs.org/>

environment and uses its package management system Node Package Manager (npm) to install, manage, and update its dependencies. The framework supports the addition of various modules that enhance the application’s functionality. Next.js enables developers to build applications quickly and efficiently. We chose this framework because of its widespread adoption, robust developer community, and our familiarity with it. Figure 4.1 displays the chat interface with detected and verified claims.

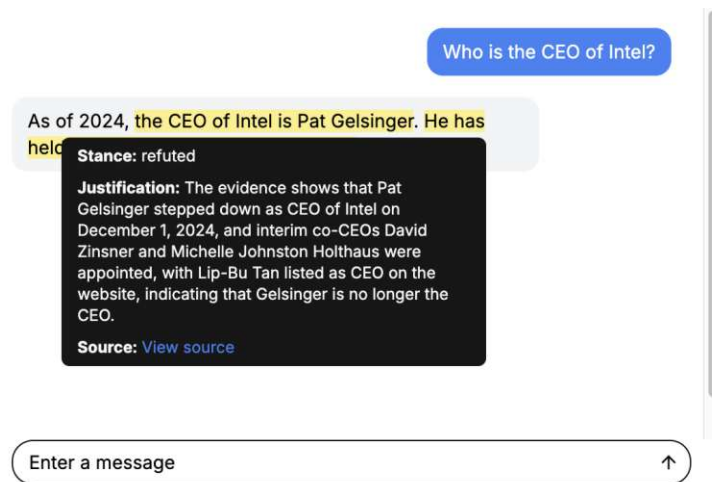


Figure 4.1: A Screenshot of the Chat Interface with Detected and Verified Claims

4.1.2 Backend

The backend is the backbone of our application, handling all the logic of the developed pipeline. Python is the language of choice for this part of the application. We leverage FastAPI² and uvicorn³ for the web server functionality. FastAPI is a modern, high-performance Python web framework designed for building APIs, leveraging asynchronous capabilities for scalability and speed. Uvicorn runs FastAPI applications, handling client requests and managing asynchronous event loops. We structure the code components into different files and classes for better code maintainability. Figure 4.2 shows the backend directory tree. `main.py` is the central start script of the application, encompassing the web server, and can be started with the simple command `python main.py`. All files within the `src` directory include the components for the backend functionality and the different parts of the pipeline. The `evaluation` directory holds the scripts for the evaluation. `pipeline.py` defines the endpoints and starts the correct functions from the defined `claim_detection.py` and `claim_verification.py` classes.

²<https://fastapi.tiangolo.com/>

³<https://www.uvicorn.org/>

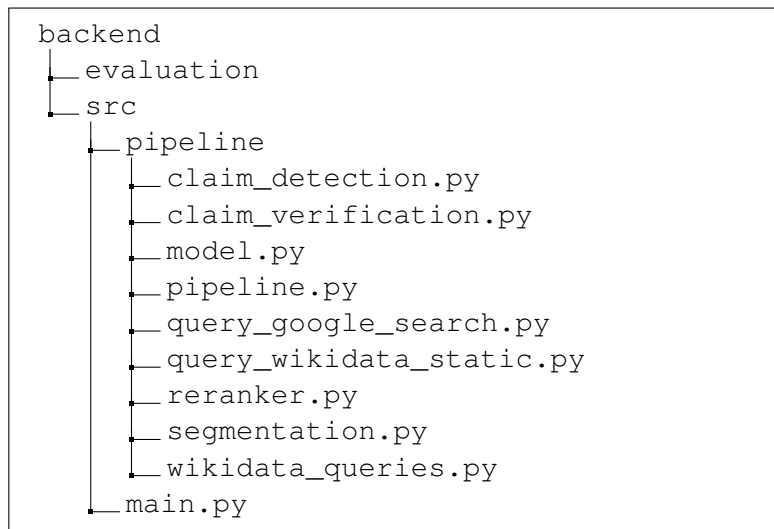


Figure 4.2: Directory Tree of the Backend

4.1.3 Development Approach

Hevner describes the design cycle of the DSF as a constant iteration between development, testing the artifact, and further enhancement based on feedback obtained from the testing process [16]. We adhere to this approach and continuously improve the artifact iteratively. Our development journey began with setting up the development environment and making foundational decisions regarding the technology we would use. For software version management, we chose Git⁴, a free and open-source Source Code Management (SCM) tool. During our development phase, we use GitHub as our SCM tool of choice. The implementation can be found here: <https://github.com/dbai-tuw/claim-detect-verify-framework>.

Given our background in software development, our goal was to encapsulate the logic from the visual frontend component. In the initial cycles, we focused on building a functional backend with a web server that provides the essential structure for both claim detection and claim verification endpoints. We utilized Postman⁵ to support the development process, designing, managing, and testing the APIs.

At the outset, we also experimented with a Node.js web server; however, we ultimately opted for a Python backend since it allows for more convenient handling of data, training, and management of machine learning models. As we are building a pipeline that processes data with numerous intermediate steps, the data continues to evolve. For ongoing development, we prioritize modifying only one part of the pipeline to observe the actual impact on the outcome. Additionally, frequent logging of the data enhances the developer experience during debugging.

⁴<https://git-scm.com/>

⁵<https://www.postman.com/>

4.1.4 Choice of LLM

During the development process, we utilized several LLM models, including OpenAI’s o4-mini, GPT-4.1, Google Gemini 2.5 Flash, and Anthropic’s Claude Sonnet 4. Currently, LLMs offer various media types; however, the purely textual input and output were entirely adequate for our prototype. Consequently, we opted to forgo LLMs with broader functionalities. As time progresses, more models from different manufacturers are emerging, each increasingly specialized for specific use cases.

We had specific requirements for the models to ensure that our artifact operates optimally. Each claim requiring verification necessitates multiple sequential LLM calls, where the output of each call serves as the input for the subsequent one, making the processing speed of the LLMs a critical requirement. Additionally, the cost of LLM requests must be manageable, as development and testing processes consume a substantial number of tokens. However, it is the evaluation phase that incurs the highest token usage. As is customary for independently conducted theses, this work is carried out without external financial support, making cost efficiency an important consideration. Smaller models frequently prove sufficient for various applications; however, models endowed with greater reasoning capabilities are not inherently negative. But it is necessary to recognize that such advanced reasoning models often incur higher costs, reflecting their enhanced functionalities. The final requirement pertains to the functionality of structured output, which is crucial for the system’s integrity. Adherence to the prescribed structure is imperative, as any deviation from it could undermine the entire pipeline’s cohesion and result in a significant number of errors.

After conducting several experiments, we decided to continue using OpenAI models exclusively. OpenAI was among the first companies to develop competitive models, leading to a strong and established market position. That is also reflected in the maturity and variety of the models. Each LLM provider requires differently structured prompts, as each has established its own API design and structure. This means that switching between different providers often requires more than just a few lines of code. In contrast, transitioning from one OpenAI model to another is straightforward, often requiring only a change in the model names. OpenAI offers two distinct API endpoints for communication. The `v1/chat/completions` endpoint and the `v1/responses` endpoint. Both offer everything we need for executing our pipeline. We aim to build this pipeline as future-proof as possible in this rapidly evolving LLM environment. Therefore, we chose to use the `v1/responses` endpoint as it offers more functionality for tool usage, state management, and other features; it is a superset of the `v1/chat/completions` API.

In Table 4.1, we present the main differences among the OpenAI models. In our pipeline, we consistently use the `gpt-4.1-mini` model, as it strikes the best balance between speed, intelligence, and token pricing. Models from OpenAI typically come in three versions: complete, mini, and nano. In terms of intelligence versus latency, the `gpt-4.1-mini` model offers significantly lower latency compared to the `gpt-4.1`, while also providing considerably more intelligence than the `gpt-4.1-nano` [93]. In

the frontend chat interface, we utilize the `gpt-4.1` model to enhance the system’s intelligence. This model is responsible for responding to user inquiries, and its claims are subsequently detected and verified.

For all OpenAI models, we consistently use a temperature of 0.0 for the most reproducibility.

Table 4.1: LLM Model Comparison

	gpt-4.1-nano	gpt-4.1-mini	gpt-4.1	o4-mini	o3
Company	OpenAI	OpenAI	OpenAI	OpenAI	OpenAI
Speed	5/5	4/5	3/5	3/5	1/5
Intelligence	2/5	3/5	4/5	4/5	5/5
Context Window	1,047,576	1,047,576	1,047,576	Reasoning	Reasoning
Input Price	\$0.10	\$0.40	\$2.00	\$1.10	\$2.00
Output Price	\$0.40	\$1.60	\$8.00	\$4.40	\$8.00
Max Output T.	32,768	32,768	32,768	100,000	100,000
Knowl. Cutoff	Jun 2024	Jun 2024	Jun 2024	Jun 2024	Jun 2024

4.1.5 Choice of Knowledge Graph

The KG in our pipeline is responsible for providing proof to verify the detected claims. In the development phase, we experimented with two different KGs: DBpedia and Wikidata. We are aware that other public KG exist, such as Yago, NELL, or OpenCyc; however, the size and therefore the information they provide is restricted.

Wikidata⁶ is an open and publicly available knowledge graph owned by the Wikimedia Foundation [114]. This database is collaboratively edited, and every item is uniquely identified by its QID. Several options exist to access and query the data stored in Wikidata, such as an interactive query builder or an API for developers. The general query language is SPARQL Protocol and RDF Query Language (SPARQL).

DBpedia is a project co-developed by various universities to enable users to query data published in Wikipedia articles semantically. The primary query language used is SPARQL. Similar to Wikidata, DBpedia structures its data in a format called Resource Description Framework (RDF). We tested the APIs from both DBpedia and Wikidata. Querying Wikidata can be challenging, as subjects can only be accessed using their QIDs. However, after reverse-engineering and discovering an entity search feature in Wikidata’s API, we were able to achieve better results with Wikidata compared to DBpedia. After testing various queries, we achieved improved data coverage regarding companies, high-level managers, and their relationships in Wikidata. Consequently, we decided to continue using Wikidata as our primary knowledge graph.

⁶<https://www.wikidata.org/>

4.2 Claim Detection

The Claim Detection is the first API in the pipeline of our framework. This endpoint accepts the whole text message that was returned by the LLM. First, the message is split into sentences. We use `spacy`⁷ for that task. Spacy loads an NLP pipeline to process the message. In this case, we use the `en_core_web_sm` pipeline that is specialized in the English language and prioritizes efficiency over accuracy. In the next section, each record, including the entire context of the message, is passed on to the OpenAI model for processing. In return, we receive the parameters in a structured output, as shown in Listing 4.1.

Listing 4.1: Structured Output of the LLM Call for Claim Detection

```
1 {
2   "sentence": "<original sentence>",
3   "reason": "<reason for classification>",
4   "original_claim": "<original claim>",
5   "claim": "<identified claim>",
6   "value": <boolean>,
7 }
```

We structure the output of the claim detection LLM call in the following way:

First, the original sentence is returned. Second, the reason for the classification is provided, allowing the LLM to consider its response before indicating whether a claim is present and identifying which claim it is. We make a clear distinction between the sentence and the `original_claim`, as the latter may only be a part of the sentence. The `claim` refers to the detected claim with pronouns exchanged, ensuring that each sentence and claim can be understood independently, without relying on surrounding context. Finally, the `value` returns a boolean indicating whether a claim has been found.

In the final response, we also return the token usage. This is not relevant for the frontend, but use the endpoint for the evaluation script as well. With that information, we can determine the total evaluation cost and average cost of detecting a claim.

4.3 Claim Verification

In this section, we outline the implementation of the claim verification API endpoint. This endpoint covers the processes of evidence retrieval from both the search engine and the knowledge graph, stance prediction, and justification production. We have combined all of these components into a single endpoint because they do not provide significant value when operated individually. In the claim detection endpoint, we have already decontextualized the claims; therefore, in this section, the claim itself can be seamlessly transferred to the backend. As discussed in Chapter 3, we will follow the data flow to explain these processes.

⁷<https://spacy.io/>

4.3.1 Evidence Retrieval

We use two separate knowledge bases to retrieve evidence for the given claim. First, we utilize a search engine; then, we query a knowledge graph.

Google Search Evidence Retrieval Google has the largest search engine market share. Google provides a Custom Search JSON API that enables querying the Google search engine and receiving a structured JSON Object. We therefore chose Google as our preferred search engine.

After receiving a claim via the claim verification API endpoint, a search query term is first created. The query is a subset of the claim. We once again utilize `spacy` and its `en_core_web_sm` pipeline for the query generation. The NLP pipeline extracts named entities and relevant content words from the claim, removes redundancies and stopwords, and compiles them into a concise, unique search query string. By calling Google's custom search JSON API with the query q , we receive a list of evidence for the q . We iterate through the list, extract the domain names, and remove all evidence from blacklisted domains. Figure 4.3 shows all blacklisted domain strings. Those domains are marked as such because the scraper was unable to extract any content other than the headings. Moreover, social media websites often appear prominently in the top results, displacing other, more relevant news websites for us.

```
"facebook.com", "twitter.com", "linkedin.com", "at.linkedin.com", "ch.linkedin.com",
"de.linkedin.com", "reddit.com", "youtube.com", "x.com", "instagram.com", "medium.com"
```

Figure 4.3: The List of All Blacklisted Domains for the Google Search

Next, we also sort out result items that are in the format of `.docx`, `.doc`, `.pdf`, or `.xlsx`, as the scraper is unable to extract sufficient content from these file types. We then compile all domains and complete Uniform Resource Locators (URLs) into a list for further processing. As a result for the query q we obtain a list of evidence $E_q = \{e_1, e_2, \dots, e_n\}$ where $n \leq 10$.

Following this, we continue with our pipeline by downloading each evidence source $e \in E$ using Jina. Jina works very simply. By calling their API endpoint in the following format: `"https://r.jina.ai/" + url` where `url` is the URL of evidence e , we receive the entire web page as a markdown file that is suitable for processing by a language model. We then prompt the language model with the claim c , the website content of e_n , and the URL of e_n . The goal is to extract a passage that either supports or refutes the claim, provide a justification for why this passage does so, assign a clear label, and include the publication date of the content. We show the JSON Schema in Listing 4.2.

Listing 4.2: JSON Schema for Verifying the Claim based on Website Content

```
1 {
2   "format": {
```

```

3     "type": "json_schema",
4     "name": "claim-schema",
5     "schema": {
6         "type": "object",
7         "properties": {
8             "passage": {
9                 "type": "string",
10                "description": "The passage that verifies or refutes the claim",
11            },
12            "justification": {
13                "type": "string",
14                "description": "The justification for the label",
15            },
16            "label": {
17                "type": "string",
18                "description": "The label of the claim",
19                "enum": ["supported", "refuted", "not_enough_evidence"],
20            },
21            "date": {
22                "type": ["string", "null"],
23                "description": "The date when the content was published in YYYY-MM-
24                DD format. If not applicable, set to null.",
25            },
26            "required": ["passage", "justification", "label", "date"],
27            "additionalProperties": False,
28        },
29    },
30 }

```

For each evidence, we add the gathered information such as URL, `found_via`, `date`, `domain`, `passage`, `justification`, and `label` into a JSON object

$$e_i = (\text{label}_i, \text{url}_i, \text{found_via}_i, \text{date}_i, \text{domain}_i, \text{passage}_i, \text{justification}_i)$$

and build a list of those. Because the process of downloading and passage extraction takes its time, we process each evidence e in a separate thread.

Knowledge Graph Evidence Retrieval As elucidated in Subsection 3.4.2, the process of obtaining knowledge graphs presents greater complexity than obtaining evidence from search engines. Given that we have selected Wikidata as our definitive source for the knowledge graph, it is imperative to use its QIDs for entity identification. Following the initiation prompt to the LLM to generate a precise question that aligns with the claim, and the subsequent request to the LLM to provide all Wikidata entities relating to individuals and companies, entities can be located and their QIDs extracted. The query of Wikidata API is accomplished with the use of a name parameter and a limit on the number of results. For example, an illustrative query for Lisa Su, AMD CEO, is as follows <https://www.wikidata.org/w/api.php?action=wbsearchentities&search=Lisa+Su&language=en&limit=10>. The output constitutes a JSON object encompassing all potential entities referenced by the name Lisa Su; the most pertinent entity consistently occupies the first position. Hence, from the initial entity, we retain the parameters `id`, the `label`, which represents the name of the entity, and `description`, where applicable.

Using the identifiers obtained for individuals and companies, we can substitute the placeholder within the static Wikidata SPARQL queries. These static queries are imported from an external Python file, as demonstrated in Listing 7.1 in the Appendix. Distinctions are made between the placeholders `company` and `person`, with each query containing one placeholder of each type. The SPARQL Wrapper class⁸ is used to execute the query service, allowing SPARQL queries to be transmitted to the defined Wikidata endpoint. Upon receiving results from Wikidata, a restructuring of the results is performed to render them in a more readable format.

Additionally, a target entity is specified, which can be an individual or a corporation. The Listing 4.3 presents the reformatted results, along with the results of two additional queries. If there are no results for any of the static queries, they are omitted from the restructured results, as their inclusion is deemed unnecessary.

Listing 4.3: Restructured Wikidata Results

```

1 {
2   "get_employers_for_person": {
3     "goal_entity": "Jeffrey W. Martin",
4     "results": [
5       {
6         "employerId": "Q7449804",
7         "employerName": "Sempra"
8       }
9     ]
10  },
11  "get_ceos_for_company": {
12    "goal_entity": "Sempra",
13    "results": [
14      {
15        "ceoId": "Q63383058",
16        "ceoName": "Jeffrey Martin",
17        "startDate": "2018-01-01T00:00:00Z"
18      }
19    ]
20  }
21 }

```

In the subsequent step, the LLM is prompted to articulate the results, which are then evaluated to determine their support or refutation of the claim. Should the results lack sufficient meaningfulness, they are classified as `not_enough_evidence`. In the final stage of the evidence retrieval process, the results obtained from Google Search and Wikidata are consolidated into a unified list. Listing 4.4 presents this unified compilation, integrating the various pieces of evidence. Although it is evident that the format of the JSON objects remains consistent, there exists information exclusive to the Google Retrieval. Any data not provided by the Wikidata Retrieval process is assigned to `null`.

Listing 4.4: Pieces of Evidence From Both Wikidata and Google Search

```

1 {
2   "claim": "Jeffrey W. Martin is the Chairman and CEO of Sempra as of June 2024.",

```

⁸<https://sparqlwrapper.readthedocs.io/en/stable/index.html>

```

3     "sources": [
4         {
5             "label": "supported",
6             "link": "https://www.sempra.com/newsroom/press-releases/sempra-appoints-
jennifer-m-kirk-board-directors",
7             "domain": "sempra.com",
8             "found_via": "google search",
9             "date": "2024-06-20T00:00:00",
10            "passage": "Jeffrey W. Martin, Sempra chairman and CEO.",
11            "justification": "The passage explicitly states that Jeffrey W. Martin is
the chairman and CEO of Sempra, confirming his roles. The date of the announcement
is June 20, 2024, which aligns with the claim date of June 2024."
12        },
13        {
14            "label": "supported",
15            "link": null,
16            "domain": null,
17            "found_via": "wikidata",
18            "date": null,
19            "passage": "Jeffrey W. Martin has been the CEO of Sempra since January 1,
2018. There is no information indicating he has left the position as of June 2024.",
20            "justification": "The data shows Jeffrey W. Martin has been the CEO since
2018 and does not indicate an end date, suggesting he is still in the role as of
June 2024."
21        },
22        {
23            "label": "not_enough_evidence",
24            "link": null,
25            "domain": null,
26            "found_via": "wikidata",
27            "date": null,
28            "passage": "The structured data indicates that Jeffrey W. Martin is
associated with Sempra, but it does not specify his current role or title as of
June 2024.",
29            "justification": "The data confirms his employment at Sempra but does not
specify his position or whether he is the Chairman and CEO as of June 2024."
30        }
31    ]
32 }

```

4.3.2 Evidence Reranking

During the evidence reranking process, we developed a proprietary Python class designed explicitly for reranking. Various metrics are computed and subsequently combined to derive a final score. Each score undergoes normalization to ensure alignment with the scoring range, which spans from 0 to 1.

Semantic Similarity Score: We use the sentence transformer `all-mpnet-base-v2` from Huggingface⁹ to create embeddings. This model, designed for general-purpose sentence tasks, has been pre-trained using a dataset consisting of over 1 billion examples. It provides the best performance compared to other general-purpose sentence models¹⁰.

⁹<https://huggingface.co/sentence-transformers/all-mpnet-base-v2>

¹⁰https://www.sbert.net/docs/sentence_transformer/pretrained_models.html

After conversion of the claim and the passage into tensor representations, the cosine similarity is calculated, which also serves as the resulting score.

Cross Encoder Score For this score, we use the AutoTokenizer with the pretrained cross-encoder model `ms-marco-TinyBERT-L-2-v2` [22]. Unlike the semantic similarity score, which typically processes input elements individually, this approach requires the concurrent input of both the claim and the passage into the model, thereby requiring explicit computational execution. The sigmoid function is employed to yield a range of values ranging from zero to one.

Entailment Score For this classification task, we utilize the `bart-large-mnli` from Facebook. It provides a pre-trained `bart-large` model [115] on the MultiNLI dataset [116]. BART is a transformer-based sequence-to-sequence model that features a bidirectional encoder, similar to BERT, and an autoregressive decoder, similar to GPT [115]. We use the `pipeline` function from Huggingface to process the claim-passage pair. If the pair receives a `entailment` label, we use the `entailment-score`; otherwise, in case of the label `contradiction`, we return the score subtracted from 1.

Label Score The evaluation is carried out using a methodical approach. A presence of either label `supported` or label `refuted` warrants a score of 1.0. In contrast, the presence of label `not_enough_evidence` for a given piece of evidence results in a score of 0.0, indicating its limited utility in the assessment process.

Trust Score The trust score is based on certain principles that are explained in Chapter 3. We assume a base score of 0.5. First, we extract the company within the claim, utilizing the `spaCy en_core_web_sm` model once again to filter out any organizations from the claim, which is then referred to as the target entity. Next, we extract the second-level domain with string matching. We compare the target entity with the second-level domain using both fuzzy matching and direct matching, utilizing the `rapidfuzz` package¹¹. Finally, we check if the URL contains any investor-related keywords, which boosts the score slightly higher. A fuzzy match gets a score of 0.8, and a direct match gets a 0.9.

Recency Score The calculation of the score is relatively straightforward. The temporal difference between the current date and the date of publication of the evidence is classified as follows: (1) If the difference is less than 30 days, the evidence is assigned a score of 1.0. (2) If the difference is between 30 and 180 days, the evidence is assigned a score of 0.7. (3) If the difference is between 180 and 365 days, a score of 0.5 is assigned. (4) If the temporal difference is between 1 and 2 years, a score of 0.3 is assigned. (5) If the evidence exceeds 2 years in age, it receives a score of 0.1. In cases where the publication date is unavailable, a score of 0.5 is assigned to avoid penalizing or rewarding the lack of a publication date.

¹¹<https://rapidfuzz.github.io/RapidFuzz/>

Final Score The final score is calculated by multiplying the weights presented in Table 3.1. We sort the pieces of evidence E_c in descending order of the final score.

4.3.3 Stance Prediction

The stance prediction constitutes the penultimate phase of the claim verification pipeline; nevertheless, it remains a critical component. The preceding phases have established a robust foundation that enables the stance prediction to function effectively. Here, the framework decides whether a claim is supported or refuted. We prompt the LLM to choose a label based on all the sorted evidence pieces. The evidence array E_c contains the following information from each evidence $e_i = (label_i, passage_i, domain_i, justification_i, date_i)$. The critical part of this LLM call is the order of the parameters in the structured output. The LLM first explains its choice and then only the suitable label. It can be one of the following labels: `supported`, `refuted`, `conflicting_evidence`, and `not_enough_evidence`. The LLM should decide on the following decision logic:

- **supported:** If there is only supporting evidence.
- **refuted:** If there is only refuting evidence.
- **conflicting_evidence:** If the evidence includes supporting and refuting evidence.
- **supported:** If the evidence is both supporting and refuting, but the supporting evidence is more recent or stronger.
- **refuted:** If the evidence is both supporting and refuting, but the refuting evidence is more recent or stronger.
- **not_enough_evidence:** If there is no clear supporting or refuting evidence, or if the evidence is too weak or ambiguous.

The complete prompt is shown in the appendix in Figure 7.7.

4.3.4 Justification Production

Ultimately, the framework offers an explanation for the user. The justification provides a coherent rationale for the user to understand the decision of support or refusal. We once again utilize LLM for this task. We present all evidence pieces, the predicted stance, and the claim that requires verification. Within the prompt, we specify that the model should elucidate why the stance is suitable, generating a sentence of one to two in length. We provide the evidence to furnish the model with a foundational base for argumentation while emphasizing that the model should solely rely on the provided evidence, abstaining from introducing additional information or conjecture.

In the structured output, the model is required to provide the justification text solely. Furthermore, the model is requested to identify the correct person who should be

referenced in the claim. This information is used exclusively for evaluation purposes and is not disclosed to the user in the front-end interface.

4.4 Deployment and Visualization

The pipeline presented is responsible for the whole functionality in the background, detecting and verifying the claim. The frontend on the other side contains little to no logic. In the frontend, we present a very simple User Interface (UI) which is common to LLM Chat interfaces. The interface was developed to make the artifact development more tangible, reasonable, and comprehensible. However, the actual development of the evaluation is irrelevant.

As already mentioned above, we use Next.js as our frontend framework. We built the components using shadcn/ui¹². It supports the frontend development by providing designed components to develop our component library. Next.js enables us to create both client and server components. Moreover, we use Next.js route handlers to manage the connection to the backend.

In the frontend, we use OpenAI's model gpt-4.1 for communication with the user. Ultimately, this is the model whose responses are validated. Once the model responds to the user's message, the frontend first sends the response to the claim detection endpoint of the backend. Once the endpoint responds with the detected claims, the latter are sent to the claim verification endpoint. Whenever Next.js receives the response from this endpoint, it is then forwarded to the visual part of the frontend.

Since the whole pipeline runs for quite a while, we implemented partial responses to make the user feel the wait a bit less. Those references are text snippets, such as "I'm thinking..." or "Let me check that...". [65]

Deployment Within this thesis, we develop an artifact that serves a research purpose. The deployment to a cloud environment is typically the last step in the development process. We are limited in monetary resources and time in this project. Therefore, we did not deploy this artifact to any environment. However, the code can be accessed in our repository, and the artifact can be experienced locally.

¹²<https://ui.shadcn.com/>



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Evaluation

In this chapter, we discuss the evaluation of our developed artifact. The DSF does not give any suggestion on how to evaluate the artifact. Therefore, we first discuss our experimental setup by examining various other approaches and their evaluation strategies. We explain the process by which we curated our dataset to establish a suitable ground truth for comprehensive evaluation. We describe our evaluation methodology and discuss the results achieved extensively. Ultimately, we outline the limitations we identified within our framework.

5.1 Experimental Setup

In this section, we highlight the various evaluation metrics employed in academic research on claim detection and verification, as well as hallucination detection.

5.1.1 Evaluation Metrics

In this subsection, we aim to address the RQ2 research question “What is an appropriate method or metric to measure the success of validation?”. We examine various alternative hallucination detection methodologies and scrutinize their validation metrics. Furthermore, we examine NLP frameworks within the domains of claim detection and claim validation, evaluating the assessment metrics applied to them.

In general, there are different methodologies to evaluate a response generated by an LLM. Chen *et al.* provide a limited overview of evaluation metrics [117] applied for hallucination detection. Although we do not aim to detect hallucinations, the metrics used can also be partially applied to claim verification. However, we focus only on machine metrics, as we lack the resources to manually evaluate all claims. Chen *et al.* mentions, for example, the F1 score, Recall, ROUGE [118], BLEU [119], or BERT [81] as possible metrics. Assessing data against a ground truth allows for the evaluation of correctness through metrics such

as accuracy or F1-score. BLEU, Recall, and ROUGE evaluate the degree of similarity between the responses produced and the reference answers. Lastly, similarity metrics such as BERT evaluate the resemblance between the LLM answer and ground-truth [117].

An examination of existing evaluations in the domains of fact checking and question answering has revealed that some methodologies employ accuracy as a metric. In contrast, others utilize the F1 score as an evaluative metric. Chen, Fisch, Weston, *et al.* developed an approach to answer open-domain questions using Wikipedia as a source. They evaluated the framework on several QA datasets and used exact string matching to measure the F1 score and accuracy [120].

The AVeriTeC dataset, used for verifying claims based on Internet sources by Schlichtkrull *et al.*, is evaluated with QA against the open web. As a metric, they used an adaptation of the Hungarian Meteor score, which they call the Averitec Score and the F1 score [121].

FEVER is a massive dataset for “Fact Extraction and VERification” [66] used accuracy and F1 score to measure the success of classification in three classes, namely supported, refuted, and not enough info [66].

AFaCTa is a framework for assistance of claim annotation based on LLMs [53]. Ni *et al.* accuracy and agreement, as they compared human-annotated claims against the annotation of the framework.

Factcheck-Bench is a framework for evaluation and annotation to assess the factuality of LLMs [54]. Wang, Gangi Reddy, Mujahid, *et al.* use Accuracy, Precision, Recall, and F1 to evaluate the framework against a groundtruth.

Wei *et al.* developed a Factuality Evaluator (SAFE) for long responses generated by LLMs based on Google Search and a multi-step reasoning process. They used an adapted F1 score especially for long-form text evaluation [52]. The improved metric is called F1@K to incorporate “human-preferred length” [52].

5.1.2 Chosen Evaluation Metrics

In the vast majority of cases, Accuracy, F1, Precision, and Recall are used in frameworks with similar research topics if a ground truth is available. There are only a few deviations, such as the F1@K score. We therefore follow the majority and use accuracy, precision, recall, and the F1 score for our evaluation metrics.

We calculate accuracy with a formula

$$Accuracy = \frac{\text{correct classification}}{\text{total classifications}} = \frac{TP + TN}{TP + TN + FP + FN} \quad (5.1)$$

while correct classification includes all correctly classified stances and total classifications include all classifications with a classified stance other than `not_enough_evidence` and `conflicting_evidence`. We refer to the latter classes as abstention.

We calculate Recall with the formula:

$$Recall = \frac{\text{correctly classified actual positives}}{\text{all actual positives}} = \frac{TP}{TP + FN} \quad (5.2)$$

We calculate this metric for both classes of supported and refuted. We exclude

We calculate Recall with the formula:

$$Precision = \frac{\text{correctly classified actual positives}}{\text{all classified as positive}} = \frac{TP}{TP + FP} \quad (5.3)$$

The F1-score is the harmonic mean and is calculated as follows:

$$F1\text{-Score} = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2TP}{TP + FP + FN} \quad (5.4)$$

We further calculate the macro F1-Score, which is the average of all calculated F1-Scores for the different classes

$$\text{Macro F1-Score} = \frac{\sum_{i=1}^n F1\text{-Score}_i}{n} \quad (5.5)$$

5.2 Dataset

As we have already outlined in Section 2, hallucinations have various sources. They can be incorporated into the data that an LLM is trained with; it may also be related to the training phase. Lastly, hallucination can occur in the application of the model itself, during the generation of text. As checking every possible hallucination is impossible with an automatic testing dataset, we focus on a subset of hallucination sources and knowledge domains. We need a dataset that does not evaluate the model itself, instead aiming to assess the framework developed. Every model has an information cutoff date, such as June 2024 for the OpenAI GPT-4.1. This restricts the model to temporal data and data that are updated frequently.

We will use this source of hallucination to evaluate our framework on such temporal data. A notable subset of domains is the area of C-level executive managers. Such high-ranking manager positions are changed frequently, which makes this domain ideal for testing our framework. Our framework utilizes Google search and Wikidata to verify the claims detected in the generated LLM responses. The transitions of high-ranking managers, when they occur, are typically announced with a press release, and articles from prominent press organizations often accompany these events. Additionally, public interest dictates that these changes be reflected in Wikidata as well. These factors make this knowledge domain ideal for evaluating our framework.

An appropriate dataset would comprise knowledge-based QA questions related to C-level executives. We explored various alternative datasets within the domain of claim

detection and validation, namely FEVER [66] and AVeriTeC [121], which encompass general knowledge across diverse domains. Other datasets focus on specific domains, such as COVID-19 [122]; however, an up-to-date dataset about senior C-level executives was not forthcoming. Consequently, we proceeded to develop our own data set that includes current management roles.

The Fortune 500 list serves as a valid framework for evaluation. This list, updated annually, includes the 500 largest corporations in the United States, ranked based on total annual revenue [123]. It is hypothesized that media coverage is more likely for companies of such magnitude in the event of management changes. Consequently, the 2024 edition of this list has been selected as the framework for our dataset. Although the 2025 edition has been released at the time of this writing, it does not impact the evaluation, as this index functions solely to establish the dataset’s framework. The companies themselves publish the relevant data.

5.2.1 Dataset Construction

In this subsection, we outline the methodology used in the development of our dataset, which includes the processes of data acquisition, data cleaning, and data preparation. The dataset comprehensively consists of all Fortune 500 companies, detailing both their executive management positions and the personnel occupying these roles. The dataset’s information was collected through web scraping on 19 March 2025, utilizing JavaScript as the primary tool. The entire company list was manually compiled, and a json file was created with attributes of Rank, Company Name, Ticker, Website, and Headquarters, with the focus predominantly placed on the company name. After achieving a comprehensive JSON format for all companies, we began data extraction for each corporation using a systematic approach implemented through two JavaScript scripts.

1. We utilized the same Google Search JSON API as used in the Google Search Information Retrieval step of the framework described in subsection 4.3.1. For every organization, we searched the web for a combination of the following keywords: {company_name} leadership team.
2. This API provides a large amount of information; however, we solely retain the URL of the first ten entries. Furthermore, a series of timeouts and errors occurred due to the excessive frequency of requests. In response, a delay mechanism was implemented, which integrates a random pause time ranging from one to five seconds between consecutive requests to the API.
3. We proceeded by downloading the content of the first URL with the help of Jina ReaderLanguage Model (LM) [112], which converts the website content to LLM-friendly markdown data.

4. Upon saving the content of the markdown website content, we employed a methodology LLM to identify and extract all managerial positions and their respective incumbents. The exact prompt used is presented in the appendix in Figure 7.9. Our process leverages the model `gpt-4o-mini-2024-07-18` developed by OpenAI. The structured output functionality, as detailed in Section 3.2.2, enabled us to obtain a `json` object that comprises an array list of all identifiable individuals and positions from the webpage. Additionally, we incorporated a command within the prompt that generates a Boolean value in the output, indicating whether an adequate number of positions were extracted from the website content.

The Boolean value is intentionally placed at the end of the structured output, as the LLM ought to prioritize extracting the necessary data from the website content before evaluating the sufficiency of this data. Should the data extracted be inadequate, the procedure advances to the subsequent URL, executing the process as previously described.

5. Subsequently, a manual plausibility assessment was carried out to ensure the establishment of a comprehensive data set. Although a significant portion of the data was of high quality, adjustments were necessary for the staff data of certain companies. Specifically, corrections were required for 49 of 500 companies. The primary issue stemmed from companies whose names included an `&` symbol, which interfered with accurate querying, and circumstances in which the scraper could not retrieve information due to restricted access, such as age verification or geoblocking. The complication with the `&` symbol arises because it has a special function in our system, serving specifically as a delimiter for separating different URL parameters when attached to a search query URL while calling the Google Search JSON API. This error was identified midway through the correction process, after adjustments had been made to approximately half of the affected companies; as such, the process was not repeated.
6. In the final step, we went automatically through all positions within a company and listed duplicates. We remove these duplicates, but we also ensure that we still have at least $n \geq 3$ personnel positions per company in our dataset. Duplicates exist because some companies list, for example, their director and vice president positions without further areas of responsibility. It therefore happens that we have different manager positions with the same name in the data record. This is not necessarily a bad thing, but we ask our evaluation questions in a specific pattern, and duplicating positions without further areas of responsibility is suboptimal in this context.

5.3 Quantitative Evaluation

We divided the evaluation into three different parts that build on each other. The first part takes place during the development stage. According to Hevner, a constant

evaluation takes place during development within the DSF. Therefore, we first describe our constant evaluation approach. Second, we did a limited evaluation on different OpenAI models, comparing their performance within our framework. Finally, we conducted a comprehensive assessment of the entire dataset to obtain the final results.

5.3.1 Evaluation during Development

We systematically evaluated our framework using diverse test datasets. The pipeline was developed incrementally, commencing with the general detection of claims. Subsequently, we focused on verification methodologies associated with Wikidata and Google Search JSON API. Ultimately, we incorporated reranking, stance prediction, and justification production. Each component of the system was developed to function independently, facilitating comprehensive testing of the individual segments. In the phase dedicated to claim verification, a variety of testing claims about managers within major corporations were utilized, such as those found in Satya Nadella is the Chairman and Chief Executive Officer (CEO) of Microsoft. and Mark Zuckerberg is the CEO of Facebook.. Since each component was engineered within its distinct Python class, we were able to export the results of each step as a JSON file and subsequently import them for the next phase, thereby enabling continuous testing.

5.3.2 Evaluation

In our conclusive evaluation, we seek to assess our developed framework utilizing an extensive database to ensure adequate informational value. We have previously outlined the methodology for using our own curated dataset and described its creation process. Before executing the final evaluation, a preliminary assessment was performed using a condensed dataset. Each invocation of an LLM entails the consumption of resources, including financial costs, computational capacity, and temporal investment. We initially describe our evaluation methodology and subsequently demonstrate the methods employed to condense the dataset for the preliminary evaluation.

Imitation of Question Answering Typically, an individual engages with the framework by posing an inquiry concerning the executive manager. Subsequently, the detection and verification processes are initiated. Ultimately, the individual receives a response from the framework LLM, which includes highlighted claims and an assessment regarding the validity of these claims. Consequently, our evaluation process was developed following the same methodology. Initially, we utilize our gold-standard dataset to formulate a query for which the answer is already known. We then proceed by generating questions based on this dataset, employing a consistent question template for all generated inquiries, as illustrated in Figure 5.1.

To formulate a question, we utilize three positions, each with its corresponding manager occupying a role. The primary selection is invariably the CEO of the company, justified by the universal presence of this executive across all corporations and the pivotal

Who is the {position of manager} of the company {company name}?

Figure 5.1: Question Template

significance of the role within the organizational hierarchy. Additionally, two other positions and their respective managers are selected through a randomization process, with particular attention to preventing duplicates. OpenAI’s `gpt-4.1-2025-04-14` model was employed to generate the answers to our formulated questions. A Python script was developed to create three questions for each of the 500 companies included in our dataset.

Claim Detection Upon generating responses for each question, we subsequently engage both disparate endpoints within our framework. Initially, each response is transmitted to the claim detection endpoint as indicated in API. Following engagement with API, each resultant response is segmented into discrete sentences with the attendant classification of each as containing a claim or not. Additionally, the process yields both the input and output tokens associated with each assessed sentence, alongside the duration required to perform the claim detection procedure per response.

Claim Verification We recognized the pattern within the responses of the LLM that the first sentence of a response always contains the name of the executive we are looking for. In the second sentence, it usually follows how long one has been part of the company or filling the position. Therefore, we only verify the first claim in a generated response. In addition, in that case, we measure not only the stance and all other parameters which are returned by the claim verification API, but we also measure how long it took the API to run through the whole process and how many tokens are used.

The way in which the question is asked to the LLM enables us to measure the success of the framework with string matching methods. We differentiate the following cases here for the class supported:

- The framework provided the stance supported and the true label is supported: TRUE POSITIVE
- The framework provided the stance supported and the true label is refuted: FALSE POSITIVE
- The framework provided the stance refuted and the true label is refuted: TRUE NEGATIVE
- The framework provided the stance refuted and the true label is supported: FALSE NEGATIVE

For the `texttrefuted` class, we swap the perspective but keep the same logic.

Preliminary Evaluation During development, we used several models, always aiming to utilize those that are either free or cost very little. Once we achieved substantial results, we also tested more powerful and intelligent models. However, for the final evaluation, we want to use a model that offers speed, an acceptable price, and the best possible intelligence. Therefore, we conducted a preliminary evaluation of ten companies, addressing three questions each. Table 5.1 shows the results of that analysis, in which we compared the three models GPT-4.1-mini, GPT-4.1-nano, and GPT-4o-mini. GPT-4.1 was not an option as Token Per Minute (TPM) are limited to 30,000¹. Unfortunately, that is not enough to complete the evaluation within a reasonable timeframe. It would also have exceeded the budget, which is why we did not consider this option.

We can see that GPT-4.1-mini has the highest accuracy and also performs best in terms of the F1 score within the *refuted* class; however, in terms of the F1 score of the *supported* class, the model is equal to the GPT-4o-mini model. If we extrapolate the price of these 30 questions to the 1500 questions, we end up with around 50\$ when using the GPT-4.1-mini model. GPT-4.1-nano was not convincing, as both its accuracy and its F1 scores are worse compared to those of the other models.

We chose GPT-4.1-mini for the full evaluation.

Table 5.1: LLM Preliminary Evaluation

	gpt-4.1-nano	gpt-4.1-mini	gpt-4o-mini
Accuracy	0.6551	0.8214	0.7826
F1-supported	0.6666	0.8	0.8
F1-refuted	0.6428	0.8387	0.7619
Ground Truth (s/r)	11 / 19	11 / 19	11 / 19
Predicted Stance (s/r/c/n)	14 / 9 / 6 / 1	14 / 14 / 1 / 1	14 / 9 / 6 / 1
Input Tokens	2,321,128	2,352,548	2,559,921
Input \$ p. 1M	0.1\$	0.4\$	0.15\$
Input Total Cost	0.23\$	0,94\$	0.38 \$
Output Token	37,274	36,495	37,295
Output \$ p 1M	0.4\$	1.6\$	0.6\$
Output Total Cost	0.01\$	0,06\$	0.02\$

5.3.3 Ablation Study

We utilize two knowledge sources, Google Search and Wikidata. Within this ablation study, we aim to visualize the impact of using Wikidata. We compare two runs, each involving the same 100 companies and addressing three questions each. In the first run, we run the complete pipeline, including retrieving knowledge from Google Search JSON API and Wikidata. In the second run, we do not use Wikidata, but only Google Search JSON API. We ensure to use the same Google Results for the best comparison. We present the results of this study in Table 5.2.

¹<https://platform.openai.com/docs/models/gpt-4.1>

Table 5.2: Ablation Study Comparing the Framework with Wikidata and Google Search Against Google Search Only

Ablation	class	Precision	Recall	F1-Score	Accuracy
Google Search + Wikidata	supported	0.8855	0.8909	0.8882	0.8635
	refuted	0.8286	0.8208	0.8246	
Only Google Search	supported	0.8951	0.8788	0.8869	0.8635
	refuted	0.8165	0.8396	0.8279	

5.4 Results

The conclusive evaluation was conducted from August 28 to 30 and required approximately 26 hours of computational time. In total, 1,500 questions concerning all 500 corporations within the Fortune 500 index were posed to the LLM. Verification was limited to claims articulated within the initial sentence of the generated response, resulting in the analysis of 1,489 claims. Of these claims, the framework was able to predict a stance, either supported or refuted, for 1,320 out of the 1,489 claims. Nine claims were assigned the label `conflicting_evidence`, and for 160 claims, the label `not_enough_evidence` could be attributed. Of the 1,320 claims with a support or refute stance, 1,140 predictions were accurate, yielding an accuracy rate of 86.3636%. In class `supported`, an F1-Score of 0.8798 was achieved, while class `refuted` attained an F1-Score of 0.8424. The results for precision, recall, and F1 score for both supported and refuted classes are presented in Table 5.3. We also identified 349 errors, including all wrongly classified claims, as well as abstentions with either `conflicting_evidence` or classified as `not_enough_evidence`.

Table 5.3: Precision, Recall and F1-Score

Class	Precision	Recall	F1-Score
supported	0.8846	0.8752	0.8798
refuted	0.8365	0.8483	0.8424
macro	0.8605	0.8617	0.8611

The mean duration required to identify all claims within a generated LLM response was 15.68 seconds. However, it is essential to note that the responses used in the evaluation typically consist of two to five sentences. The average time taken for claim verification per individual claim is 55.28 seconds. During the evaluation, a significant amount of LLM tokens were processed. During the detection phase, a total of 1,116,702 input tokens and 906,782 output tokens were consumed. The claim verification process required the use of 162,882,020 input tokens and yielded 1,905,947 output tokens.

5.5 Discussion

LLMs have entered the world at full speed, and more and more applications have integrated the usage of LLMs. Moreover, the use of an LLM chatbot, such as ChatGPT[1], is increasing steadily, helping people in their daily lives and during professional tasks at work. Unfortunately, the emergence of hallucinations has not decreased with the further development of the models, but rather increased [8]. As models become more reliable and users build confidence in them by providing accurate information in familiar domains, the risk posed by hallucinations may increase [18]. Therefore, the objective of this work is to provide an approach for detecting and verifying claims within LLM responses, thus offering a solution for detecting and treating hallucinations. In the evaluation, we demonstrated the extent to which the framework supports users by providing veracity predictions. With an accuracy rate of 86.36%, the framework correctly classifies the claim in a significant number of cases.

Having such a high accuracy rate might suggest that one can fully trust the developed framework. This is a valid assumption; however, we also need to consider that the framework did 349 errors within 1489 claims, which means that the framework is not reliable in all cases. However, this can also be challenged, as the stance classes of `not_enough_evidence` and `conflicting_evidence` are also included in this category. With those stances, the framework admits that no clear stance can be found here, either due to a lack of evidence or conflicting sources.

The F1-scores for both classes `supported` and `refuted` were computed to ensure that our framework does not default to a particular class. It is observed that both classes exhibit relatively similar values, with the `supported` class having a score of 0.8798 and the `refuted` class having a score of 0.8424. This indicates that the framework does not exhibit unilateral decision-making or bias towards a specific class. It is important to clarify that varied error costs are associated with fact-checking. A false positive, or Type-1 error, occurs when the framework incorrectly identifies a statement as `supported` when it should, in fact, label it as `refuted`, thereby accepting false information. This type of error poses a greater risk than a Type-2 error, or false negative, where the framework indicates a `refuted` statement while the ground truth is `supported`. Although this is also undesirable, it results in different implications, as the user would likely re-evaluate the claim and conduct further research.

Compared to similar approaches, such as Averitec [121], which is a dataset for real-world claim verification with evidence from the open web, our framework produces excellent results.

This study also recognizes certain limitations inherent in the research conducted. Notably, the time lag between the curation of the dataset, which occurred in March 2025, and the evaluation period has led to instances where the framework accurately predicts the appropriate stance, yet the dataset reflects a divergent ground truth. An illustrative example of such a scenario is presented in the following question:

Who is the Senior Vice President and Chief Financial Officer of the company Alphabet?

The generated response from the LLM starts with the sentence

The Senior Vice President and Chief Financial Officer of Alphabet Inc. is Ruth Porat.

At the time of dataset creation, this would have been accurate. However, the framework predicted the stance `refuted` as it identified evidence suggesting the contrary. The framework provides the subsequent explanation:

The evidence shows that Ruth Porat served as Senior Vice President and Chief Financial Officer of Alphabet Inc. from May 2015 until July 2024. However, more recent evidence from mid-2024 indicates that she transitioned to the role of President and Chief Investment Officer, and Anat Ashkenazi took over as CFO and Senior Vice President starting July 31, 2024. Therefore, Ruth Porat is not currently the Senior Vice President and CFO of Alphabet Inc., refuting the claim.

In this instance, the framework exhibits superior quality compared to the data utilized for its evaluation.

We also discovered that management changes that occurred before the knowledge cut-off date were also partially inaccurate. Therefore, this phenomenon is not attributed to an artificial hallucination originating from temporal changes after the knowledge cut-off date of a model, but rather arises solely from inaccuracies present in the original training data. An illustrative example is the following. We asked this particular question because many companies also list CEO positions of subsidiaries on their own parent company site, such as Walmart Inc. listing the CEO position of Sam's Club, which is a division of Walmart. We have posed the following question to the LLM:

Who is the President and CEO, Sam's Club of the company Walmart?

We received the following response.

The President and CEO of Sam's Club, a division of Walmart Inc., is Kathryn McLay. She has held this position since November 2019.

She has been the CEO of Sam's Club until September 2023. Since then, she has been promoted to lead Walmart International.

In this work, we use the method of DSF [15], [16]. It comprises three cycles with distinct objectives. The relevance cycle aims to verify the relevance of the developed artifact. We

argue that relevance is given to the requirement that users of LLMs and products that integrate them want to trust the responses by the LLM. Hallucinations prevent people from fully trusting the latter.

Within the rigor cycle, we utilize established foundations, such as scientific literature, software documentation, and frameworks, to ground our research. Primarily, we utilize an NLP framework for automated fact checking [65] that separates the core issue in several phases of claim detection and claim verification. We utilized those phases to break down our core problem into smaller pieces and develop a framework for automatic fact-checking based on current state-of-the-art LLM technology.

Within the design cycle, we built the artifact with constant evaluation. We integrated a final evaluation recommended by Pries-Heje *et al.* [14], which followed an ex post/artificial evaluation approach based on a dataset covering a large number of current executives within the 2024 Fortune 500 index.

Usage of Wikidata Wikidata effectively supported the verification of specific claims with verified information. However, this applies only to very few claims in our dataset. Unfortunately, we were unable to achieve significantly better results using Wikidata in comparison to only using the Google Search JSON API. There are various reasons for this. The first reason is based on the usage of static queries, which, in fact, limits the information retrieval from Wikidata. We were sometimes unable to extract the required data using the four static queries. We hypothesized that Wikidata can be particularly convincing in temporal data; however, we were able to disprove this because information is often not fully mapped in Wikidata.

Costs To sum up the financial expenditures associated with this research, the predominant costs were incurred from utilizing the LLM during both the development and evaluation phases. During development, significant expenses arose from the trial-and-error processes involved in prompt engineering and testing of the pipeline procedures. The cost of running the final pipeline is approximately 0.03\$ to 0.05\$, depending on the length of the response and the number of claims detected. Moreover, the expenses not only included LLM costs but were also allocated to Jina, the readerLM [112], which is responsible for converting website content into LLM-optimized markdown format. Initially, a strategy of using tokens in a free tier was used during development; however, a transition to a paid tier was necessary during evaluation, as the process would have exceeded the free token allocation. Jina does not offer a pay-per-use plan; instead, token packages can be purchased. We made an expenditure of 20\$ for 1,000,000,000 tokens. The complete evaluation required approximately 100,000,000 tokens, so we consumed about 10% of the purchased budget. Lastly, costs have been incurred by using the Google Search JSON API. In general, 100 requests per day are free of charge, which was enough during development. In the course of the evaluation, we used a paid subscription that is 5\$ for 1000 requests, resulting in costs of approximately \$6 to use the API.

In conclusion, we propose a set of recommendations. The current methodology addresses hallucinations post-occurrence. Alternative strategies exist, such as RAG, or the integration of web search capabilities within the ChatGPT interface. These measures proactively minimize hallucinations by enabling the model to recognize the need for updated data in specific inquiries. We advocate for the utilization of a claim detection and verification framework in conjunction with other hallucination mitigation strategies to optimize the user experience.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Conclusion and Future Work

In this chapter, we intend to draw conclusive insights from our research. Initially, we address the research questions comprehensively, followed by a comprehensive conclusion. Subsequently, we outline the limitations that merit further exploration and, ultimately, present a perspective on topics that could be explored in future research.

6.1 Discussion of Research Questions

In Chapter 1.1, we defined three research questions that have defined the direction of our research. Here, we now answer those questions, starting with the first research question:

RQ1: What is an appropriate means to effectively validate LLM-created knowledge using knowledge graphs and news sources?

Based on our iterative development and evaluation methodology within the DSF, we can answer this research question with the artifact produced, which is shown in Section 3. We developed this framework with relevance given by a stated factuality issue [7] and based on a broad knowledge base anchored on software documentation, prior research on LLMs, and is grounded in the NLP framework given by Guo *et al.* [65]. Through the utilization of contemporary advanced methodologies, including prompt chaining, structured outputs, and role prompting, we can integrate responses from LLM into standard software procedures, thereby facilitating the development of a framework for claim detection and verification. The structured output, as illustrated in Subsection 3.2.2, significantly enhances the value of this framework by providing structured data in response to inherently unstructured inquiries with complete reliability. The absence of this feature would significantly hinder the implementation of the framework. We followed the DSF and constantly evaluated the artifact during development, concluding with an expert / artificial evaluation [14] that demonstrated the validity of our created framework.

Regarding the second question:

RQ2: What is an appropriate method or metric to measure the success of validation?

We compared several approaches within the same or related domains and provided an outline on how other frameworks are evaluated in Section 5.1.1. A significant degree of consensus was observed regarding methods and metrics, with a focus on utilizing the F1 score, precision, recall, and accuracy. However, the application of these metrics requires the existence of a predefined ground truth, which the evaluated approaches uniformly depend on. Our study adopts these metrics, dependent on the existence of our created ground truth. We have a curated dataset at our disposal, and a ground truth has been established to facilitate the evaluation of our framework.

Tackling the third research question:

RQ3: To what extent can newly created knowledge created by LLMs be effectively validated through news sources and knowledge graphs?

We can achieve remarkable results in evaluating the framework, which sets a solid baseline for addressing hallucination issues of LLMs. Utilizing our curated dataset, which comprises 1500 questions about executive management roles along with their respective accurate answers, we specifically address factual hallucinations that emerge due to existing knowledge constraints within LLMs, as delineated by Kasai *et al.*[39]. Testing our framework on this dataset, we could achieve an accuracy of 0.8636, an F1 score of 0.8798 for the supported class and 0.8483 for the refuted class. We demonstrate that the framework has a strong ability to detect and verify claims within the responses of LLMs; however, it also shows that there is still room for further improvement, as some false positives and true negatives were observed within our evaluation data set. We outline further improvement possibilities in Section 6.3. Moreover, our ablation study revealed that the results predominantly derived from Google Search JSON API significantly contribute to the framework's efficacy. Evaluation of the framework utilizing identical datasets sans Wikidata yielded comparable outcomes. This phenomenon can be attributed, in part, to the employment of static queries, which inherently limit the scope of information retrieval. Consequently, we deduce that the novel knowledge generated by LLMs predominantly pertains to verification through news sources.

6.2 Limitations

We achieved excellent results with the developed framework; however, we had limited resources available for this work. Therefore, we list here all the limitations of the framework whose processing no longer fits into the scope of the work. The scope limits us in terms of time, human resources, and financial support.

Dataset Scope and Question Sample A targeted evaluation was conducted to enable a comprehensive analysis of the framework using a selectively curated, high-quality dataset, with prospects for future validation extending across various datasets. However, this dataset imposes constraints on the evaluation, restricting it to a single category of questions and a specific topic area. We used the set of representative questions to demonstrate feasibility and effectiveness, which can be extended to broader domains in future iterations.

Dataset Date Limitation The dataset was assembled on a designated date. Given that shifts in management positions can occur sporadically, the dataset does not entirely represent the current status of all management positions; however, it remains largely accurate. Subsequent changes were acknowledged; however, amending the dataset was not feasible, as it would require a review of each company to ensure the currency of the data. Consequently, this snapshot-based dataset mirrors the real-world conditions prevalent at the time of its collection.

Evaluation Scope The holistic end-to-end evaluation methodology provides a thorough understanding of pipeline performance and emphasizes its applicability in real-world contexts. However, this study did not perform an individual assessment of each step. Although such an analysis could have been carried out with additional effort, it did not align with the scope of this investigation. We consider our comprehensive evaluation to be adequate to substantiate the practicability and advantages of the system.

LLM Model Choice At the beginning of our development, we experimented with several LLMs; later in the process, we opted for OpenAI as our central LLM. We optimized the prompting strategy for OpenAI models, ensuring strong initial performance; however, it is adaptable to other LLMs with minor adjustments. Furthermore, OpenAI introduced GPT-5 on August 07, 2025 [124]. Due to a lack of temporal resources, we were unable to conduct a thorough analysis to assess the potential improvements the model could have contributed to the framework.

Hyperparameter Configuration The hyperparameters of the LLM remained unoptimized in our study. Optimization is feasible exclusively with a limited set of models; however, such a model was not used in our research. In the context of a substantial number of models, fine-tuning is unnecessary [25], as the extensive context window enables the model to effectively adapt to the specific application requirements through varied prompting techniques.

Runtime The comprehensive verification process prioritizes accuracy and transparency over speed. With a mean processing time of 15.68 seconds for claim detection and a mean processing time per claim of 55.29 seconds for verification, we establish a sufficient baseline. This fosters opportunities for optimization in future development.

Prompt Engineering The established framework, characterized by a substantial number of LLM calls, demonstrates a significant reliance on the crafted prompts. The effectiveness of prompt engineering is essentially uncertain and depends on subjective perception. An enhancement is achieved predominantly through iterative processes of trial and error. Consequently, the definitive manifestation of a complete solution remains elusive, signaling a continuous potential for further refinement.

6.3 Future Work

Throughout the development process, numerous opportunities have emerged for conducting in-depth research. This section aims to explore these possibilities. Specific topics arise as a direct continuation of the limitations identified in this study.

In a particular development stage, the work was continued exclusively with OpenAI models. Evaluating the performance of this framework with models from other manufacturers could provide valuable insights. Thus, we propose assessing the adaptability of the framework in various other LLMs. In addition, we refer to our holistic evaluation approach in the limitations section, which overlooks the evaluation of each step individually. Therefore, we suggest exploring additional evaluation techniques to substantiate the claim detection component. Furthermore, we recommend examining the application of the pipeline in a broader range of domains and datasets. Extending the dataset to include other domains and various types of questions should also be a fundamental objective for future research. Subsequently, a comprehensive reranking algorithm was implemented within this framework. The evaluation of this reranking model was not conducted individually, as it does not contribute to the primary function of the framework, which is to validate the agreement of all sources and to distill a definitive stance from them. The primary purpose of relevance sorting is to provide a source for user insight. It is recommended to conduct an in-depth examination of this method to enhance and evaluate a relevance sorting algorithm. Finally, we recommend further examination of the source retrieval process. Although parallelization and pipeline speed improvements have been implemented, it remains of interest to investigate potential advancements in optimizing the information retrieval procedure.

6.4 Conclusion

This study presents a potential method for addressing hallucinations in LLMs. Leveraging advanced prompting strategies, including role prompting, prompt chaining, and structured output, within our developed framework, we can detect and verify claims following a NLP framework [65]. By obtaining and analyzing current Google search results, we extract sufficient information for claim verification. Additionally, we employ the Wikidata knowledge graph to retrieve structured data using four static queries. Verbalization of the results from the KG allows for ranking of all sources from the search engine and KG based on criteria such as recency, entailment, and semantic similarity, thereby ordering

the results according to their relevance. From this ordered list, a position is expressed and a justification is formulated to assist LLM chat users in assessing the validity of the claim.

In the development of this framework, we followed the DSF guidelines established by Hevner [16], and justify its relevance in relation to the factuality concerns of LLMs highlighted by [7]. Our approach incorporated an extensive knowledge base, including software documentation, existing frameworks, and academic literature. The evaluation of the developed framework was conducted using a meticulously curated dataset featuring C-level executives from all Fortune 500 companies. The assessment yielded robust results when compared to the established gold standard dataset. With an accuracy of 0.8636 and a macro F1-score of 0.8611, our findings substantiate the validity of the constructed framework, while also recognizing the potential for further enhancement.

Through this work, we strive to contribute to the scientific discourse on hallucination detection and mitigation methodologies, thereby establishing a solid foundation for subsequent research in this domain.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Appendix

7.1 Static SPARQL Queries

In this subsection, we present the queries for retrieving Wikidata evidence. They are stored in a dictionary within a Python script, which facilitates easy access to the placeholders and allows for the use of queries.

Listing 7.1: The Static Wikidata Queries

```

queries = {
  "get_employers_for_person": {
    "description": "Get employers for a person",
    "params": ["PERSON_ID"],
    "query": """
      SELECT ?employerId ?employerIdLabel ?startOfEmployment ?
      endOfEmployment ?source WHERE {
        wd:{PERSON_ID} p:P108 ?statement.
        ?statement ps:P108 ?employerId.
        OPTIONAL { ?statement pq:P580 ?startOfEmployment. }
        OPTIONAL { ?statement pq:P582 ?endOfEmployment. }
        OPTIONAL {
          ?statement prov:wasDerivedFrom ?refNode.
          ?refNode pr:P854 ?source.
        }
        SERVICE wikibase:label { bd:serviceParam wikibase:
language "en". }
      }
    """,
  },
  "get_ceos_for_company": {
    "description": "Get CEOs for a company",
    "params": ["COMPANY_ID"],
  }
}

```

```

    "query": ""
      SELECT ?ceoId ?ceoIdLabel ?startDate ?endDate ?source
WHERE {
  wd:{COMPANY_ID} p:P169 ?statement.
  ?statement ps:P169 ?ceoId.
  OPTIONAL { ?statement pq:P580 ?startDate. }
  OPTIONAL { ?statement pq:P582 ?endDate. }
  OPTIONAL {
    ?statement prov:wasDerivedFrom ?refNode.
    ?refNode pr:P854 ?source.
  }
  SERVICE wikibase:label { bd:serviceParam wikibase:
language "en". }
}
ORDER BY (?startDate)
"",
},
"get_companies_for_person": {
  "description": "Get companies where a person has been CEO",
  "params": ["PERSON_ID"],
  "query": ""
    SELECT ?companyId ?companyIdLabel ?start ?end ?source
WHERE {
  wd:{PERSON_ID} p:P39 ?statement.
  ?statement ps:P39 wd:Q484876;
  pq:P108 ?companyId.
  OPTIONAL { ?statement pq:P580 ?start. }
  OPTIONAL { ?statement pq:P582 ?end. }
  OPTIONAL {
    ?statement prov:wasDerivedFrom ?refNode.
    ?refNode pr:P854 ?source.
  }
  SERVICE wikibase:label { bd:serviceParam wikibase:
language "en". }
}
"",
},
"get_parent_and_owner_of_companies": {
  "description": "Get parent companies for a company",
  "params": ["COMPANY_ID"],
  "query": ""
    SELECT ?relationType ?relatedEntityId ?
relatedEntityIdLabel ?start ?end ?proportion ?source WHERE {
  {
    wd:{COMPANY_ID} p:P127 ?ownershipStatement.
    ?ownershipStatement ps:P127 ?relatedEntityId.
    BIND("owner" AS ?relationType)
    OPTIONAL { ?ownershipStatement pq:P1107 ?proportion.
}
}

```

```

OPTIONAL { ?ownershipStatement pq:P585 ?start. }
OPTIONAL {
  ?ownershipStatement prov:wasDerivedFrom ?refNode.
  ?refNode pr:P854 ?source.
}
}
UNION
{
  wd:{COMPANY_ID} p:P749 ?parentStatement.
  ?parentStatement ps:P749 ?relatedEntityId.
  BIND("parent organization" AS ?relationType)
  OPTIONAL { ?parentStatement pq:P580 ?start. }
  OPTIONAL { ?parentStatement pq:P582 ?end. }
  OPTIONAL {
    ?parentStatement prov:wasDerivedFrom ?refNode.
    ?refNode pr:P854 ?source.
  }
}
}
SERVICE wikibase:label { bd:serviceParam wikibase:
language "en". }
}
ORDER BY (?relationType) (?start)
""",
},
}

```

7.2 Used Prompts in the Framework

We present all the prompts utilized in our developed framework in this section. We have sorted the prompts according to the order in which they are called up in the framework. Last, we outline the prompt that was used to extract the correct executive management data from a given webpage.

You are an expert fact-checker specializing in analyzing text about corporate executives, including CEOs, CFOs, and other high-ranking managers. Your task is the following:

Definition of a Factual Claim:

A verifiable factual claim must meet all of the following criteria:

1. Specific: The claim makes a clear assertion (not vague or generalized).
 2. Verifiable: It can be checked against publicly available sources such as:
 - Company reports (e.g., SEC filings, earnings reports) financial statements
 - press releases
 - reputable news sources (e.g., Bloomberg, Reuters, WSJ or the company's official website)
 3. Non-speculative: It is not a prediction, opinion, or uncertain statement.
1. Please evaluate sentences generated by an LLM and determine whether they contain a verifiable factual claim based on publicly available information.
 2. Identify the specific part of the sentence that constitutes the verifiable claim.
 3. Reason about the verifiability of the claim based on the provided context and publicly available sources.

Classification & JSON Output Format:

- If the sentence starts with "As of June 2024," or similar phrases, ignore this part and focus on the rest of the sentence.
- If the sentence contains a verifiable factual claim, set "value": true.
- The "claim" should be the specific part of the sentence that is a verifiable factual claim.
- If the "claim" contains a pronoun, replace it with the name of the person or company being referred to.
- If a role is mentioned, mention the role in the "claim" field.
- The "original_claim" should be the claim without pronoun replacements and original capitalization, but only the claim part."
- If the sentence does NOT contain a verifiable factual claim, set "value": false and provide a "reason":
 - Opinion → Subjective or evaluative statements.
 - Speculation → Hypothetical or forward-looking statements.
 - Ambiguous → Vague or lacking clear details.
 - Unverifiable → No clear way to confirm via public sources.

Be strict! If a claim cannot be easily verified, classify it as "False".

Figure 7.1: Claim Detection Prompt

You are an AI assistant tasked with verifying or refuting a claim about C-level executives using the content of a given website.

Input:

- Claim: A statement about C-level executives.
- Website Content (Markdown): The extracted text from a webpage.
- Website URL: The source of the website content.

Instructions:

- Look for a passage in the website content that directly supports or refutes the claim.
- If a supporting passage is found, set "label": supported, extract the passage, and provide reasoning.
- If no supporting passage is found, set "label": not_enough_evidence, leave the passage as an empty string, and explain why verification was not possible.
- If contradictory information is found, set "label": "refuted" and provide a justification.
- If the content states a publishing date, include it in the "date" field.

Output Format:

Return a JSON object with the following structure:

```
{
  "passage": "Extracted passage that verifies or refutes the claim or an empty string if not found.",
  "justification": "Explanation of why the passage verifies or refutes the claim, or why verification was not possible.",
  "label": <supported, refuted, not_enough_evidence>,
  "date": "Date in format YYYY-MM-DD mentioned when the content was published, if applicable, else None"
}
```

Do not infer or assume verification beyond what is explicitly stated in the website content. Be strict in your evaluation.

Keep responses concise and factual. Provide clear and specific justifications.

Figure 7.2: Claim Verification: Google Search Passage Extraction

You are an assistant that helps determine which Wikidata queries can be used to verify claims about C-level executives. You are given a claim in plain language. Based on the claim, respond with a JSON object indicating which of the following Wikidata queries could help verify the claim:

1. "get_employers_for_person": Returns all employers of a person including start and end dates of employment (if available).
2. "get_ceos_for_company": Returns all CEOs of a company, including start and end dates (if available).
3. "get_companies_for_person": Returns all companies where a person held the position of chief executive officer, including start and end dates (if available).
4. "get_parent_and_owner_of_companies": Returns all parent organizations and owners of a company, including start and end dates (if available).

Each query result also includes sources, if available.

Sometimes a claim can be checked using more than one of the queries. Return all relevant queries as a JSON object with boolean values indicating whether the query can help verify the claim.

Figure 7.3: Prompt to Decide if Wikidata Would be Helpful

You are an expert in verifying factual claims about C-level executives. Your task is to generate a question whose correct answer is the given claim. The question should be neutral, concise, and structured to elicit the claim as a direct response.

Figure 7.4: Wikidata: Question Generation Prompt

You are an expert in constructing SPARQL queries for Wikidata. Your task is to identify relevant Wikidata entities (subjects and objects) needed to construct a SPARQL query to verify a given claim. Additionally, a directly related question helps you. You should analyze the claim and return a structured list of entities, categorized by type. Do not generate the SPARQL query itself. Instead, focus on listing the necessary Wikidata items required to build the query. Ensure your response follows the format below.

Response format:

Subjects: Entities that are the main focus of the claim.

Predicates: Relationships or properties linking entities.

Objects: Values or entities that serve as the target of the relationship.

Qualifiers: Additional details that refine the statement, if applicable.

Provide the most relevant entities based on common Wikidata usage patterns.

Concentrate on subjects and objects. Only return subjects and objects which are people and companies. Return the names as how you would think it would appear on wikidata. If you are unsure, return more than one company or person. Do not include predicates or qualifiers in your response. Use the following json format with example persons and companies:

```
{"type": "Person",  
  "entity": "Mark Zuckerberg"},  
{"type": "Company",  
  "entity": "Tesla"}
```

Figure 7.5: Wikidata: Prompt to Retrieve the Needed Entities for Wikidata

You are a fact verification assistant.
 Your task is to decide whether the following claim is supported, refuted, or has not enough evidence based on structured data from Wikidata.
 Also, provide a short justification in natural language based on the data.

Claim:
 {claim}

Wikidata Output:
 {wikidata_output}

Instructions:

- If the structured data clearly supports the claim (e.g., a person currently holds the position or is employed by the company), label it as supported.
- If the data includes an end date indicating the person is no longer in the role (e.g., stepped down as CEO), label the claim as refuted.
- If the data is incomplete, missing, or unrelated to the claim, label it as not_enough_evidence.

Note:

- The presence of an end date typically means the role or employment has ended.
- For current positions, make sure the end date is either missing or in the future.

Output Format:

```

  {{
    "found_via": "wikidata",
    "passage": "Formulate a passage that is a summary of the relevant evidence you
    found in the structured data.",
    "justification": "A short explanation in natural language.",
    "label": "supported" or "refuted" or "not_enough_evidence",
    "link": "A source URL to the evidence you found in the structured data. If no source
    is found", leave it empty."
  }}
  
```

Figure 7.6: Wikidata: Stance Prediction and Verbalization

You are a stance prediction assistant in a claim verification process. Your task is to classify a claim based on a set of provided evidence. The evidence is sorted by relevance with the most relevant evidence first.

The possible labels are:

- supported
- refuted
- conflicting evidence
- not enough evidence

Use the following decision logic:

- If the evidence only supports the claim, label it supported.
- If the evidence only refutes the claim, label it refuted.
- If the evidence includes both supporting and refuting items, label the claim as conflicting evidence.
- If the evidence mostly supports or refutes the claim, take care on the date of the evidence and label it accordingly. Newer evidence should be prioritized.
- If the evidence is both supporting and refuting, but the supporting evidence is more recent or stronger, label it supported.
- If the evidence is both supporting and refuting, but the refuting evidence is more recent or stronger, label it refuted.
- If there is no clear supporting or refuting evidence, or the evidence is too weak or ambiguous, label it not enough evidence.

Always return only the label, and an explanation of why you chose that label. No extra text.

Figure 7.7: The Stance Prediction Prompt

You are a Justification Generation Assistant. Your job is to generate a clear and concise justification for a claim, based on provided evidence and a labeled stance.

The input includes:

- A claim (a factual or inferential statement)
- One or more pieces of evidence (text snippets or statements)
- A stance (either "SUPPORTED", "REFUTED", "CONFLICTING EVIDENCE" or "NOT ENOUGH INFORMATION")

You must produce a justification that:

- Clearly explains why the stance is appropriate
- Uses only the evidence provided (do not speculate or add new information)
- Is logically sound and grounded in the content of the evidence
- Is 1–3 sentences in length and focused on clarity

Additionally, based on the input, who would be the correct person to be mentioned in the justification? If there is not enough evidence or you cannot find a person, return "unknown".

Do not repeat the claim or stance in your response. Only output the justification.

Figure 7.8: The Justification Production Prompt

Extract a structured list of executives from the following text:

{text}

Please clearly label if you were able to extract the information or not by setting the successful boolean. Only set it to true if you found more than 3 executives and their positions. Furthermore, only set the value to true if the extracted information includes more than Directors. It should include positions like Chief Executive Officer and Chief Financial Officer.

Figure 7.9: Dataset Construction: the Manager Positions Extraction Prompt

Overview of Generative AI Tools Used

Grammarly: This tool contributed to this thesis by responding to the following prompts:

- “Make it sound academic.”
- “Improve it.”

Writeful: This tool contributed to this thesis by changing the style of my written input to a scientific tone or summarizing text passages. It is integrated into Overleaf and utilizes its own TexGPT model to support researchers.

GitHub Copilot: This tool contributed to this thesis with its autocomplete feature. It helps improve developer efficiency by suggesting recurring code components.

OpenAI: We use the models o4-mini, GPT4.1, GPT4.1-mini, GPT4.1-nano, GPT4o, GPT4o-mini for the following tasks:

- Utilization within the Framework.
- Asking questions regarding LaTeX errors.
- Asking questions regarding Prompt Optimization.
- Asking questions regarding Evaluation Metrics.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Figures

1.1	The Three Cycle View of Design Science Research by Hevner [16]	4
1.2	Methodology, Approach, and Evaluation of This Work	6
3.1	A Complete Overview of the Developed Framework	16
3.2	A Screenshot of the Chat Interface	17
3.3	Procedure of the Claim Verification	21
4.1	A Screenshot of the Chat Interface with Detected and Verified Claims . .	28
4.2	Directory Tree of the Backend	29
4.3	The List of All Blacklisted Domains for the Google Search	33
5.1	Question Template	47
7.1	Claim Detection Prompt	64
7.2	Claim Verification: Google Search Passage Extraction	65
7.3	Prompt to Decide if Wikidata Would be Helpful	66
7.4	Wikidata: Question Generation Prompt	66
7.5	Wikidata: Prompt to Retrieve the Needed Entities for Wikidata	67
7.6	Wikidata: Stance Prediction and Verbalization	68
7.7	The Stance Prediction Prompt	69
7.8	The Justification Production Prompt	70
7.9	Dataset Construction: the Manager Positions Extraction Prompt	70



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Tables

3.1	The Used Weights For Different Scores in the Reranking Process.	26
4.1	LLM Model Comparison	31
5.1	LLM Preliminary Evaluation	48
5.2	Ablation Study Comparing the Framework with Wikidata and Google Search Against Google Search Only	49
5.3	Precision, Recall and F1-Score	49



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acronyms

API	Application Programming Interface
CoT	Chain-of-Thought
DS	Design Science
DSF	Design Science Framework
GPT	Generative Pretrained Transformer
JSON	JavaScript Object Notation
KG	Knowledge Graph
LLM	Large Language Model
LM	Language Model
NLG	Natural Language Generation
NLP	Natural Language Processing
npm	Node Package Manager
PLM	Pretrained Language Models
QA	Question Answering
RAG	Retrieval Augmented Generation
RDF	Resource Description Framework
SCM	Source Code Management
SFT	Supervised Fine-Tuning
SPARQL	SPARQL Protocol and RDF Query Language
TPM	Token Per Minute
UI	User Interface
URL	Uniform Resource Locator



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliography

- [1] OpenAI, *Introducing ChatGPT*, Nov. 2022. [Online]. Available: <https://openai.com/index/chatgpt/>.
- [2] Microsoft, *GitHub Copilot – November 30th Update · GitHub Changelog*, Nov. 2023. [Online]. Available: <https://github.blog/changelog/2023-11-30-github-copilot-november-30th-update/>.
- [3] H. Touvron, T. Lavril, G. Izacard, *et al.*, *LLaMA: Open and Efficient Foundation Language Models*, Feb. 2023. DOI: 10.48550/arXiv.2302.13971.
- [4] Gemini Team Google, R. Anil, J.-B. Alayrac, *et al.*, *Gemini: A Family of Highly Capable Multimodal Models*, May 2025. DOI: 10.48550/arXiv.2312.11805.
- [5] A. Yang, B. Yang, B. Zhang, *et al.*, *Qwen2.5 Technical Report*, Jan. 2025. DOI: 10.48550/arXiv.2412.15115. arXiv: 2412.15115.
- [6] D. Kalla, N. Smith, F. Samaah, and S. Kuraku, *Study and Analysis of Chat GPT and its Impact on Different Fields of Study*, SSRN Scholarly Paper, Rochester, NY, Mar. 2023. [Online]. Available: <https://papers.ssrn.com/abstract=4402499>.
- [7] I. Augenstein, T. Baldwin, M. Cha, *et al.*, „Factuality challenges in the era of large language models and opportunities for fact-checking“, *Nature Machine Intelligence*, vol. 6, no. 8, pp. 852–863, Aug. 2024, ISSN: 2522-5839. DOI: 10.1038/s42256-024-00881-z.
- [8] OpenAI, *OpenAI o3 and o4-mini System Card*, Apr. 2025. [Online]. Available: <https://openai.com/index/o3-o4-mini-system-card/>.
- [9] S. Chen, Y. Zhao, I.-C. Chern, S. Gao, P. Liu, and J. He, *FELM: Benchmarking Factuality Evaluation of Large Language Models*, Nov. 2023. DOI: 10.48550/arXiv.2310.00741.
- [10] S. Feng, V. Balachandran, Y. Bai, and Y. Tsvetkov, „FactKB: Generalizable Factuality Evaluation using Language Models Enhanced with Factual Knowledge“, in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 933–952. DOI: 10.18653/v1/2023.emnlp-main.59.

- [11] A. S. George, A. S. H. George, and A. G. Martin, „A Review of ChatGPT AI’s Impact on Several Business Sectors“, *Partners Universal International Innovation Journal*, vol. 1, no. 1, pp. 9–23, Feb. 2023, ISSN: 2583-9675. DOI: 10.5281/zenodo.7644359.
- [12] R. Nakano, J. Hilton, S. Balaji, *et al.*, *WebGPT: Browser-assisted question-answering with human feedback*, Jun. 2022. DOI: 10.48550/arXiv.2112.09332.
- [13] J. Venable, J. Pries-Heje, and R. Baskerville, „A Comprehensive Framework for Evaluation in Design Science Research“, in *Design Science Research in Information Systems. Advances in Theory and Practice*, vol. 7286, Springer Berlin Heidelberg, May 2012, pp. 423–438, ISBN: 978-3-642-29862-2. DOI: 10.1007/978-3-642-29863-9_31.
- [14] J. Pries-Heje, R. Baskerville, and J. Venable, „Strategies for Design Science Research Evaluation“, in *16th European Conference on Information Systems, ECIS 2008*, Jan. 2008, pp. 255–266.
- [15] A. R. Hevner, S. T. March, J. Park, and S. Ram, „Design science in information systems research“, *MIS Q.*, vol. 28, no. 1, pp. 75–105, Mar. 2004, ISSN: 0276-7783.
- [16] A. Hevner, „A Three Cycle View of Design Science Research“, *Scandinavian Journal of Information Systems*, vol. 19, Jan. 2007.
- [17] J. Iivari, „A paradigmatic analysis of information systems as a design science“, *Scandinavian Journal of Information Systems*, vol. 19, p. 39, Jan. 2007.
- [18] OpenAI, J. Achiam, S. Adler, *et al.*, *GPT-4 Technical Report*, Mar. 2024. DOI: 10.48550/arXiv.2303.08774.
- [19] Y. Tan, D. Min, Y. Li, *et al.*, „Can ChatGPT Replace Traditional KBQA Models? An In-Depth Analysis of the Question Answering Performance of the GPT LLM Family“, in *The Semantic Web – ISWC 2023*, T. R. Payne, V. Presutti, G. Qi, *et al.*, Eds., Cham: Springer Nature Switzerland, 2023, pp. 348–367, ISBN: 978-3-031-47240-4. DOI: 10.1007/978-3-031-47240-4_19.
- [20] W. Jiaqi, L. Xinliang, L. Zhengliang, *et al.*, „LLM Reasoning: From OpenAI O1 to DeepSeek R1“, May 2025. [Online]. Available: <https://hal.science/hal-05058659>.
- [21] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, „Attention is All you Need“, in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, vol. 30, Red Hook, NY, USA: Curran Associates, Inc., 2017, pp. 6000–6010. DOI: 10.5555/3295222.3295349.

- [22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding“, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.
- [23] J. Wei, Y. Tay, R. Bommasani, *et al.*, *Emergent Abilities of Large Language Models*, Oct. 2022. DOI: 10.48550/arXiv.2206.07682.
- [24] W. X. Zhao, K. Zhou, J. Li, *et al.*, *A Survey of Large Language Models*, Mar. 2025. DOI: 10.48550/arXiv.2303.18223.
- [25] T. Brown, B. Mann, N. Ryder, *et al.*, „Language Models are Few-Shot Learners“, in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: <https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfc4967418bfb8ac142f64a-Abstract.html>.
- [26] Z. Ji, N. Lee, R. Frieske, *et al.*, „Survey of Hallucination in Natural Language Generation“, *ACM Comput. Surv.*, vol. 55, no. 12, 248:1–248:38, Mar. 2023, ISSN: 0360-0300. DOI: 10.1145/3571730.
- [27] V. Rawte, A. Sheth, and A. Das, *A Survey of Hallucination in Large Foundation Models*, Sep. 2023. DOI: 10.48550/arXiv.2309.05922.
- [28] N. Dziri, A. Madotto, O. Zaiane, and A. J. Bose, „Neural Path Hunter: Reducing Hallucination in Dialogue Systems via Path Grounding“, in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 2197–2214. DOI: 10.18653/v1/2021.emnlp-main.168.
- [29] J. Maynez, S. Narayan, B. Bohnet, and R. McDonald, „On Faithfulness and Factuality in Abstractive Summarization“, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, May 2020, pp. 1906–1919. DOI: 10.18653/v1/2020.acl-main.173.
- [30] Y. Huang, X. Feng, X. Feng, and B. Qin, *The Factual Inconsistency Problem in Abstractive Text Summarization: A Survey*, Apr. 2023. DOI: 10.48550/arXiv.2104.14839.
- [31] Y. Zhang, Y. Li, L. Cui, *et al.*, „Siren’s Song in the AI Ocean: A Survey on Hallucination in Large Language Models“, *Computational Linguistics*, pp. 1–46, 2023. DOI: 10.1162/COLI.a.16.
- [32] L. Huang, W. Yu, W. Ma, *et al.*, „A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions“, *ACM Trans. Inf. Syst.*, vol. 43, no. 2, 42:1–42:55, Jan. 2025, ISSN: 1046-8188. DOI: 10.1145/3703155.

- [33] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, „On the Dangers of Stochastic Parrots - Can Language Models Be Too Big?“, in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, ser. FAccT '21, New York, NY, USA: Association for Computing Machinery, Mar. 2021, pp. 610–623, ISBN: 978-1-4503-8309-7. DOI: 10.1145/3442188.3445922.
- [34] A. Paullada, I. D. Raji, E. M. Bender, E. Denton, and A. Hanna, „Data and its (dis)contents: A survey of dataset development and use in machine learning research“, *Patterns*, vol. 2, no. 11, Nov. 2021, ISSN: 2666-3899. DOI: 10.1016/j.patter.2021.100336.
- [35] P. Narayanan Venkit, S. Gautam, R. Panchanadikar, T.-H. Huang, and S. Wilson, „Nationality Bias in Text Generation“, in *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, A. Vlachos and I. Augenstein, Eds., Dubrovnik, Croatia: Association for Computational Linguistics, May 2023, pp. 116–122. DOI: 10.18653/v1/2023.eacl-main.9.
- [36] F. Ladhak, E. Durmus, M. Suzgun, *et al.*, „When Do Pre-Training Biases Propagate to Downstream Tasks? A Case Study in Text Summarization“, in *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, A. Vlachos and I. Augenstein, Eds., Dubrovnik, Croatia: Association for Computational Linguistics, May 2023, pp. 3206–3219. DOI: 10.18653/v1/2023.eacl-main.234.
- [37] N. Kandpal, H. Deng, A. Roberts, E. Wallace, and C. Raffel, „Large Language Models Struggle to Learn Long-Tail Knowledge“, in *Proceedings of the 40th International Conference on Machine Learning*, PMLR, Jul. 2023, pp. 15 696–15 707. DOI: 10.5555/3618408.3619049.
- [38] A. Mallen, A. Asai, V. Zhong, R. Das, D. Khashabi, and H. Hajishirzi, „When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories“, in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds., Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 9802–9822. DOI: 10.18653/v1/2023.acl-long.546.
- [39] J. Kasai, K. Sakaguchi, Y. Takahashi, *et al.*, *RealTime QA: What’s the Answer Right Now?*, Feb. 2024. DOI: 10.48550/arXiv.2207.13332.
- [40] S. Min, S. Gururangan, E. Wallace, *et al.*, *SILO Language Models: Isolating Legal Risk In a Nonparametric Datastore*, Jul. 2024. DOI: 10.48550/arXiv.2308.04430.
- [41] Z. Gekhman, G. Yona, R. Aharoni, *et al.*, „Does Fine-Tuning LLMs on New Knowledge Encourage Hallucinations?“, in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Miami, Florida, US: Association for Computational Linguistics, Nov. 2024, pp. 7765–7784. DOI: 10.18653/v1/2024.emnlp-main.444.

- [42] D. Chiang and P. Cholak, „Overcoming a Theoretical Limitation of Self-Attention“, in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, S. Muresan, P. Nakov, and A. Villavicencio, Eds., Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 7654–7664. DOI: 10.18653/v1/2022.acl-long.527.
- [43] C. Wang and R. Sennrich, „On Exposure Bias, Hallucination and Domain Shift in Neural Machine Translation“, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schlueter, and J. Tetreault, Eds., Online: Association for Computational Linguistics, Jul. 2020, pp. 3544–3552. DOI: 10.18653/v1/2020.acl-main.326.
- [44] M. Zhang, O. Press, W. Merrill, A. Liu, and N. A. Smith, *How Language Model Hallucinations Can Snowball*, May 2023. DOI: 10.48550/arXiv.2305.13534. arXiv: 2305.13534 [cs].
- [45] H. Zhang, S. Diao, Y. Lin, *et al.*, „R-Tuning: Instructing Large Language Models to Say ‘I Don’t Know’“, in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 7113–7139. DOI: 10.18653/v1/2024.naacl-long.394.
- [46] J. Wei, D. Huang, Y. Lu, D. Zhou, and Q. V. Le, „Simple synthetic data reduces sycophancy in large language models“, in *ICLR 2025*, Feb. 2024. [Online]. Available: <https://openreview.net/forum?id=WDheQxWAO4>.
- [47] X. Chen, M. Li, X. Gao, and X. Zhang, „Towards Improving Faithfulness in Abstractive Summarization“, *Advances in Neural Information Processing Systems*, vol. 35, pp. 24516–24528, Dec. 2022.
- [48] L. Berglund, M. Tong, M. Kaufmann, *et al.*, *The Reversal Curse: LLMs trained on "A is B" fail to learn "B is A"*, May 2024. DOI: 10.48550/arXiv.2309.12288.
- [49] I.-C. Chern, S. Chern, S. Chen, *et al.*, *FacTool: Factuality Detection in Generative AI – A Tool Augmented Framework for Multi-Task and Multi-Domain Scenarios*, Jul. 2023. DOI: 10.48550/arXiv.2307.13528.
- [50] S. Huo, N. Arabzadeh, and C. L. A. Clarke, *Retrieving Supporting Evidence for LLMs Generated Answers*, Jun. 2023. DOI: 10.48550/arXiv.2306.13781.
- [51] S. Min, K. Krishna, X. Lyu, *et al.*, „FACTScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation“, in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Singapore: Association for Computational Linguistics, Dec. 2023, pp. 12076–12100. DOI: 10.18653/v1/2023.emnlp-main.741.
- [52] J. Wei, C. Yang, X. Song, *et al.*, „Long-form factuality in large language models“, in *Proceedings of the 38th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA: Curran Associates Inc., Nov. 2024, pp. 80756–80827. DOI: 10.48550/arXiv.2403.18802.

- [53] J. Ni, M. Shi, D. Stambach, M. Sachan, E. Ash, and M. Leippold, „AFaCTA: Assisting the Annotation of Factual Claim Detection with Reliable LLM Annotators“, in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, L.-W. Ku, A. Martins, and V. Srikumar, Eds., Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 1890–1912. DOI: 10.18653/v1/2024.acl-long.104.
- [54] Y. Wang, R. Gangi Reddy, Z. M. Mujahid, *et al.*, „Factcheck-Bench: Fine-Grained Evaluation Benchmark for Automatic Fact-checkers“, in *Findings of the Association for Computational Linguistics: EMNLP 2024*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds., Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 14199–14230. DOI: 10.18653/v1/2024.findings-emnlp.830.
- [55] Y. Song, Y. Kim, and M. Iyyer, „VeriScore: Evaluating the factuality of verifiable claims in long-form text generation“, in *Findings of the Association for Computational Linguistics: EMNLP 2024*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds., Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 9447–9474. DOI: 10.18653/v1/2024.findings-emnlp.552.
- [56] S. Dhuliawala, M. Komeili, J. Xu, *et al.*, „Chain-of-Verification Reduces Hallucination in Large Language Models“, in *Findings of the Association for Computational Linguistics: ACL 2024*, Bangkok, Thailand: Association for Computational Linguistics, 2023, pp. 3563–3578. DOI: 10.18653/v1/2024.findings-acl.212.
- [57] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, „Language Models are Unsupervised Multitask Learners“, *OpenAI blog*, Feb. 2019. [Online]. Available: <https://api.semanticscholar.org/CorpusID:160025533>.
- [58] K. Meng, D. Bau, A. Andonian, and Y. Belinkov, „Locating and Editing Factual Associations in GPT“, *Advances in Neural Information Processing Systems*, vol. 35, pp. 17359–17372, Dec. 2022.
- [59] N. De Cao, W. Aziz, and I. Titov, „Editing Factual Knowledge in Language Models“, in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, Eds., Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 6491–6506. DOI: 10.18653/v1/2021.emnlp-main.522.
- [60] V. Karpukhin, B. Oguz, S. Min, *et al.*, „Dense Passage Retrieval for Open-Domain Question Answering“, in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds., Online: Association for Computational Linguistics, Nov. 2020, pp. 6769–6781. DOI: 10.18653/v1/2020.emnlp-main.550.
- [61] P. Lewis, E. Perez, A. Piktus, *et al.*, „Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks“, in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 9459–9474. [Online].

Available: <https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html>.

- [62] Z. Yang, P. Qi, S. Zhang, *et al.*, „HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering“, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, Eds., Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 2369–2380. DOI: 10.18653/v1/D18-1259.
- [63] A. Fan, Y. Jernite, E. Perez, D. Grangier, J. Weston, and M. Auli, „ELI5: Long Form Question Answering“, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum, and L. Màrquez, Eds., Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 3558–3567. DOI: 10.18653/v1/P19-1346.
- [64] L. Gao, Z. Dai, P. Pasupat, *et al.*, „RARR: Researching and Revising What Language Models Say, Using Language Models“, in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds., Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 16477–16508. DOI: 10.18653/v1/2023.acl-long.910.
- [65] Z. Guo, M. Schlichtkrull, and A. Vlachos, „A Survey on Automated Fact-Checking“, *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 178–206, Feb. 2022, ISSN: 2307-387X. DOI: 10.1162/tacl_a_00454.
- [66] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, „FEVER: A Large-scale Dataset for Fact Extraction and VERification“, in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, M. Walker, H. Ji, and A. Stent, Eds., New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 809–819. DOI: 10.18653/v1/N18-1074.
- [67] N. Hassan, C. Li, and M. Tremayne, „Detecting Check-worthy Factual Claims in Presidential Debates“, in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, ser. CIKM ’15, New York, NY, USA: Association for Computing Machinery, Oct. 2015, pp. 1835–1838, ISBN: 978-1-4503-3794-6. DOI: 10.1145/2806416.2806652.
- [68] A. Barrón-Cedeño, T. Elsayed, P. Nakov, *et al.*, „Overview of CheckThat! 2020: Automatic Identification and Verification of Claims in Social Media“, in *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 11th International Conference of the CLEF Association, CLEF 2020, Thessaloniki, Greece, September 22–25, 2020, Proceedings*, Berlin, Heidelberg: Springer-Verlag, Sep. 2020, pp. 215–236, ISBN: 978-3-030-58218-0. DOI: 10.1007/978-3-030-58219-7_17.

- [69] P. Nakov, A. Barrón-Cedeño, T. Elsayed, *et al.*, „Overview of the CLEF-2018 CheckThat! Lab on Automatic Identification and Verification of Political Claims“, in *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, P. Bellot, C. Trabelsi, J. Mothe, *et al.*, Eds., Cham: Springer International Publishing, 2018, pp. 372–387, ISBN: 978-3-319-98932-7. DOI: 10.1007/978-3-319-98932-7_32.
- [70] Y. Liu, M. Ott, N. Goyal, *et al.*, *RoBERTa: A Robustly Optimized BERT Pre-training Approach*, Jul. 2019. DOI: 10.48550/arXiv.1907.11692.
- [71] C. Samarinas, W. Hsu, and M. L. Lee, „Improving Evidence Retrieval for Automated Explainable Fact-Checking“, in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, A. Sil and X. V. Lin, Eds., Online: Association for Computational Linguistics, Jun. 2021, pp. 84–91. DOI: 10.18653/v1/2021.naacl-demos.10.
- [72] W. Ferreira and A. Vlachos, „Emergent: A novel data-set for stance classification“, in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California: Association for Computational Linguistics, 2016, pp. 1163–1168. DOI: 10.18653/v1/N16-1138.
- [73] I. Augenstein, C. Lioma, D. Wang, *et al.*, „MultiFC: A Real-World Multi-Domain Dataset for Evidence-Based Fact Checking of Claims“, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 4685–4697. DOI: 10.18653/v1/D19-1475.
- [74] R. Reinanda, E. Meij, and M. de Rijke, „Knowledge Graphs: An Information Retrieval Perspective“, *Foundations and Trends® in Information Retrieval*, vol. 14, no. 4, pp. 289–444, Oct. 2020, ISSN: 1554-0669, 1554-0677. DOI: 10.1561/15000000063.
- [75] C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber, and P. Cimiano, „Template-based question answering over RDF data“, in *Proceedings of the 21st International Conference on World Wide Web*, ser. WWW '12, New York, NY, USA: Association for Computing Machinery, Apr. 2012, pp. 639–648, ISBN: 978-1-4503-1229-5. DOI: 10.1145/2187836.2187923.
- [76] A. Abujabal, M. Yahya, M. Riedewald, and G. Weikum, „Automated Template Generation for Question Answering over Knowledge Graphs“, in *Proceedings of the 26th International Conference on World Wide Web*, ser. WWW '17, Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, Apr. 2017, pp. 1191–1200, ISBN: 978-1-4503-4913-0. DOI: 10.1145/3038912.3052583.

- [77] J. Li, B. Hui, G. Qu, *et al.*, „Can LLM Already Serve as A Database Interface? A BIG Bench for Large-Scale Database Grounded Text-to-SQLs“, in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, vol. 36, Curran Associates Inc., Dec. 2023, pp. 42 330–42 357. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/hash/83fc8fab1710363050bbd1d4b8cc0021-Abstract-Datasets_and_Benchmarks.html.
- [78] Z. Hong, Z. Yuan, Q. Zhang, *et al.*, *Next-Generation Database Interfaces: A Survey of LLM-based Text-to-SQL*, Jul. 2024. DOI: 10.48550/arXiv.2406.08426.
- [79] X. Liu, F. Wu, T. Xu, *et al.*, *Evaluating the Factuality of Large Language Models using Large-Scale Knowledge Graphs*, Apr. 2024. [Online]. Available: <http://arxiv.org/abs/2404.00942>.
- [80] P. A. K. K. Diallo, S. Reyd, and A. Zouaq, „A Comprehensive Evaluation of Neural SPARQL Query Generation From Natural Language Questions“, *IEEE Access*, vol. 12, pp. 125 057–125 078, 2024, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2024.3453215.
- [81] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, *BERTScore: Evaluating Text Generation with BERT*, Feb. 2020. DOI: 10.48550/arXiv.1904.09675.
- [82] Y. Zhang, D. Long, G. Xu, and P. Xie, *HLATR: Enhance Multi-stage Text Retrieval with Hybrid List Aware Transformer Reranking*, May 2022. DOI: 10.48550/arXiv.2205.10569.
- [83] A. Dong, Y. Chang, Z. Zheng, *et al.*, „Towards recency ranking in web search“, in *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, ser. Wsdm '10, New York, New York, USA: Association for Computing Machinery, 2010, pp. 11–20, ISBN: 978-1-60558-889-6. DOI: 10.1145/1718487.1718490.
- [84] M. Potthast, J. Kiesel, K. Reinartz, J. Bevendorff, and B. Stein, „A Stylometric Inquiry into Hyperpartisan and Fake News“, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, I. Gurevych and Y. Miyao, Eds., Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 231–240. DOI: 10.18653/v1/P18-1022.
- [85] L. Graves, „Understanding the promise and limits of automated fact-checking“, *Reuters Institute for the Study of Journalism*, 2018. [Online]. Available: 10.60625/risj-nqnx-bg89.
- [86] J. E. Uscinski and R. W. Butler, „The Epistemology of Fact Checking“, *Critical Review*, vol. 25, no. 2, pp. 162–180, Jun. 2013, ISSN: 0891-3811. DOI: 10.1080/08913811.2013.843872.
- [87] Z. C. Lipton, „The Mythos of Model Interpretability“, *Queue*, vol. 16, no. 3, pp. 31–57, Jun. 2018. DOI: 10.1145/3236386.3241340.

- [88] K. Popat, S. Mukherjee, A. Yates, and G. Weikum, „DeClarE: Debunking Fake News and False Claims using Evidence-Aware Deep Learning“, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 22–32. DOI: 10.18653/v1/D18-1003.
- [89] M. H. Gad-Elrab, D. Stepanova, J. Urbani, and G. Weikum, „ExFaKT: A Framework for Explaining Facts over Knowledge Graphs and Text“, in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, ser. WSDM '19, New York, NY, USA: Association for Computing Machinery, Jan. 2019, pp. 87–95, ISBN: 978-1-4503-5940-5. DOI: 10.1145/3289600.3290996.
- [90] P. Atanasova, J. G. Simonsen, C. Lioma, and I. Augenstein, „Generating Fact Checking Explanations“, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds., Online: Association for Computational Linguistics, Jul. 2020, pp. 7352–7364. DOI: 10.18653/v1/2020.acl-main.656.
- [91] Anthropic, „Claude 4.1 System Card“, Aug. 2025.
- [92] Mistral AI Team, *The all new le Chat: Your AI assistant for life and work / Mistral AI*, Feb. 2025. [Online]. Available: <https://mistral.ai/news/all-new-le-chat>.
- [93] OpenAI, *Introducing GPT-4.1 in the API*, Apr. 2025. [Online]. Available: <https://openai.com/index/gpt-4-1/>.
- [94] J. Wei, M. Bosma, V. Y. Zhao, *et al.*, „Finetuned Language Models Are Zero-Shot Learners“, in *International Conference on Learning Representations*, arXiv, 2022. DOI: 10.48550/arXiv.2109.01652.
- [95] Z. M. Wang, Z. Peng, H. Que, *et al.*, „RoleLLM: Benchmarking, Eliciting, and Enhancing Role-Playing Abilities of Large Language Models“, in *Findings of the Association for Computational Linguistics: ACL 2024*, Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 14743–14777. DOI: 10.18653/v1/2024.findings-acl.878.
- [96] A. Kong, S. Zhao, H. Chen, *et al.*, *Better Zero-Shot Reasoning with Role-Play Prompting*, Mar. 2024. DOI: 10.48550/arXiv.2308.07702.
- [97] J. Wei, X. Wang, D. Schuurmans, *et al.*, „Chain-of-Thought Prompting Elicits Reasoning in Large Language Models“, in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, Curran Associates Inc., 2023, pp. 24824–24837. DOI: 10.5555/3600270.3602070.
- [98] M. Pokrass, *Introducing Structured Outputs in the API*, Aug. 2024. [Online]. Available: <https://openai.com/index/introducing-structured-outputs-in-the-api/>.

- [99] L. Liu and T. Koo, *Mastering Controlled Generation with Gemini 1.5: Schema Adherence for Developers- Google Developers Blog*, Sep. 2024. [Online]. Available: <https://developers.googleblog.com/en/mastering-controlled-generation-with-gemini-15-schema-adherence/>.
- [100] Z. Xie, *Order Matters in Hallucination: Reasoning Order as Benchmark and Reflexive Prompting for Large-Language-Models*, Aug. 2024. DOI: 10.48550/arXiv.2408.05093.
- [101] D. Castillo, *Structured outputs can hurt the performance of LLMs*, Dec. 2024. [Online]. Available: <https://dylancastillo.co/posts/say-what-you-mean-sometimes.html>.
- [102] Z. R. Tam, C.-K. Wu, Y.-L. Tsai, C.-Y. Lin, H.-y. Lee, and Y.-N. Chen, „Let Me Speak Freely? A Study on the Impact of Format Restrictions on Performance of Large Language Models“, in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, Miami, Florida, US: Association for Computational Linguistics, Nov. 2024, pp. 1218–1236. DOI: 10.18653/v1/2024.emnlp-industry.91.
- [103] K. Boland, P. Fafalios, A. Tchechmedjiev, S. Dietze, and K. Todorov, „Beyond facts – a survey and conceptualisation of claims in online discourse analysis“, *Semantic Web*, vol. 13, no. 5, pp. 793–827, Aug. 2022, ISSN: 1570-0844. DOI: 10.3233/SW-212838.
- [104] M. Lippi and P. Torroni, „Argument Mining from Speech: Detecting Claims in Political Debates“, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, Mar. 2016, ISSN: 2374-3468. DOI: 10.1609/aaai.v30i1.10384.
- [105] L. Konstantinovskiy, O. Price, M. Babakar, and A. Zubiaga, *Towards Automated Factchecking: Developing an Annotation Schema and Benchmark for Consistent Automated Claim Detection*, Aug. 2020. DOI: 10.48550/arXiv.1809.08193.
- [106] M. Wanner, S. Ebner, Z. Jiang, M. Dredze, and B. V. Durme, „A Closer Look at Claim Decomposition“, in *Proceedings of the 13th Joint Conference on Lexical and Computational Semantics (*SEM 2024)*, Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 153–175. DOI: 10.18653/v1/2024.starsem-1.13.
- [107] Q. Hu, Q. Long, and W. Wang, „Decomposition Dilemmas: Does Claim Decomposition Boost or Burden Fact-Checking Performance?“, in *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, Association for Computational Linguistics, Apr. 2025, pp. 6313–6336. DOI: 10.18653/v1/2025.naacl-long.320.
- [108] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, *Large Language Models are Zero-Shot Reasoners*, Jan. 2023. DOI: 10.48550/arXiv.2205.11916.
- [109] Z. Zhang, A. Zhang, M. Li, and A. Smola, *Automatic Chain of Thought Prompting in Large Language Models*, Oct. 2022. DOI: 10.48550/arXiv.2210.03493.

- [110] M. Potthast, M. Hagen, T. Gollub, *et al.*, „Overview of the 5th International Competition on Plagiarism Detection“, *CEUR Workshop Proceedings*, vol. 1180, pp. 845–876, 2013.
- [111] G. Karadzhov, P. Nakov, L. Màrquez, A. Barrón-Cedeño, and I. Koychev, „Fully Automated Fact Checking Using External Sources“, in *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, R. Mitkov and G. Angelova, Eds., Varna, Bulgaria: INCOMA Ltd., Sep. 2017, pp. 344–353. DOI: 10.26615/978-954-452-049-6_046.
- [112] F. Wang, Z. Shi, B. Wang, N. Wang, and H. Xiao, *ReaderLM-v2: Small Language Model for HTML to Markdown and JSON*, Mar. 2025. DOI: 10.48550/arXiv.2503.01151.
- [113] R. Fielding, „Representational State Transfer (REST)“, Ph.D. dissertation, 2000.
- [114] M. Chalabi, „Welcome to Wikidata! Now what?“, *The Guardian*, Apr. 2013, ISSN: 0261-3077. [Online]. Available: <https://www.theguardian.com/news/datablog/2013/apr/26/wikidata-launch>.
- [115] M. Lewis, Y. Liu, N. Goyal, *et al.*, „BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension“, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds., Online: Association for Computational Linguistics, Jul. 2020, pp. 7871–7880. DOI: 10.18653/v1/2020.acl-main.703.
- [116] A. Williams, N. Nangia, and S. Bowman, „A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference“, in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, M. Walker, H. Ji, and A. Stent, Eds., New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 1112–1122. DOI: 10.18653/v1/N18-1101.
- [117] Y. Chen, Q. Fu, Y. Yuan, *et al.*, „Hallucination Detection: Robustly Discerning Reliable Answers in Large Language Models“, in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, ser. CIKM ’23, New York, NY, USA: Association for Computing Machinery, Oct. 2023, pp. 245–255, ISBN: 979-8-4007-0124-5. DOI: 10.1145/3583780.3614905.
- [118] C.-Y. Lin, „ROUGE: A Package for Automatic Evaluation of Summaries“, in *Text Summarization Branches Out*, Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: <https://aclanthology.org/W04-1013/>.
- [119] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, „Bleu: A Method for Automatic Evaluation of Machine Translation“, in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, P. Isabelle, E. Charniak, and D. Lin, Eds., Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 311–318. DOI: 10.3115/1073083.1073135.

- [120] D. Chen, A. Fisch, J. Weston, and A. Bordes, „Reading Wikipedia to Answer Open-Domain Questions“, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, R. Barzilay and M.-Y. Kan, Eds., Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 1870–1879. DOI: 10.18653/v1/P17-1171.
- [121] M. Schlichtkrull, Z. Guo, and A. Vlachos, „AVeriTeC: A Dataset for Real-world Claim Verification with Evidence from the Web“, in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, vol. 36, Red Hook, NY, USA: Curran Associates Inc., 2023, pp. 65 128–65 167. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/cd86a30526cd1aff61d6f89f107634e4-Paper-Datasets_and_Benchmarks.pdf.
- [122] I. Mohr, A. Wüthrich, and R. Klinger, „CoVERT: A Corpus of Fact-checked Biomedical COVID-19 Tweets“, in *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, Marseille, France: European Language Resources Association, Jun. 2022, pp. 244–257. [Online]. Available: <https://aclanthology.org/2022.lrec-1.26/>.
- [123] Fortune, *Fortune 500*, 2024. [Online]. Available: <https://fortune.com/ranking/fortune500/>.
- [124] OpenAI, *Introducing GPT-5 for developers*, Aug. 2025. [Online]. Available: <https://openai.com/index/introducing-gpt-5-for-developers/>.