CHECKEMBED: Effective Verification of LLM Solutions to Open-Ended Tasks

Maciej Besta* ETH Zurich
Lorenzo Paleari ETH Zurich
Ales Kubicek ETH Zurich
Piotr Nyczyk Cledar
Robert Gerstenberger ETH Zurich
Patrick Iff ETH Zurich
Tomasz Lehmann Cledar, Warsaw University of Technology
Hubert Niewiadomski Cledar
Torsten Hoefler ETH Zurich

Abstract

Large Language Models (LLMs) are revolutionizing various domains, yet verifying their answers remains a significant challenge, especially for intricate open-ended tasks such as consolidation, summarization, and extraction of knowledge. In this work, we propose CHECKEMBED: an *accurate*, *scalable*, and *simple* LLM verification approach. CHECKEMBED is driven by a straightforward yet powerful idea: in order to compare LLM solutions to one another or to the ground-truth (GT), *compare their corresponding answer-level embeddings obtained with a model such as GPT Text Embedding Large*. This reduces a complex textual answer to a single embedding, facilitating straightforward, fast, and meaningful verification. We develop a comprehensive verification pipeline implementing the CHECKEMBED methodology. The CHECKEMBED pipeline also comes with metrics for assessing the truthfulness of the LLM answers, such as embedding heatmaps and their summaries. We show how to use these metrics for deploying practical engines that decide whether an LLM answer is satisfactory or not. We apply the pipeline to real-world document analysis tasks, including term extraction and document summarization, showcasing significant improvements in accuracy, cost-effectiveness, and runtime performance compared to existing token-, sentence-, and fact-level schemes such as BERTScore or SelfCheckGPT.

Website & code: `https://github.com/spcl/CheckEmbed`

1 Introduction

Large Language Models (LLMs) [34, 60] are transforming the world. One particular ongoing challenge in the LLM design is hallucination detection [13, 35, 58] and the corresponding overall verification of LLM answers [6, 39]. Numerous works tried to address this issue, focusing on – for example – grounding knowledge or explainability, and even giving rise to questions regarding methodology and epistemology of artificial intelligence (AI) in general [10].

Recent verification methods and their building blocks, such as SelfCheckGPT [32] and BERTScore [57], focus on individual fact checking and token- as well as sentence-level analysis. For example, SelfCheckGPT, albeit introduced recently, is an established mechanism for LLM verification. As one of its methods, it uses BERTScore to compare LLM answers. In BERTScore, one first obtains contextual embeddings of all words in two compared sentences, computes cosine similarity scores between all pairs of embeddings from two sentences, and then greedily matches

*corresponding author

Preprint. Under review.

each token from one sentence to the most similar token in the other sentence. Weighted averages of these matchings are used to obtain precision, recall, and the final F1 score that enables comparing different sentences.

However, the problem of verifying LLM answers to more complex tasks, such as open-ended document analyses, still poses a challenge. As an example of such a task, consider extracting legal terms and their definitions from a document. The difficulty of verifying the answers to such a task is due to the inherent lack of structure, even assuming one has the ground-truth answer. Namely, the output of such a request would be a potentially long list of definitions. To verify this answer, one needs to verify that each term is listed and appropriately defined. Harnessing existing methods such as SelfCheckGPT or BERTScore is not scalable, because their token-, sentence-, and fact-based approaches scale poorly with growing task size. Moreover, we observe that while two different LLM answers can comprise of very different sets of sentences, their *meaning* could indeed be very similar. This aspect is not well reflected by sentence- and token-level schemes.

In this work, we propose CHECKEMBED: an approach for *simple*, *scalable*, and *accurate* verification of LLM solutions to such tasks (**contribution 1**). The **key idea** behind CHECKEMBED is to *obtain and compare embeddings of full LLM answers*, or their sizeable chunks, instead of focusing on individual sentences, facts, or tokens. CHECKEMBED relies on the fact that modern embedding models are highly capable; for example, they can be based on powerful Decoder-only LLMs [19]. Thus, they provide high-dimensional embeddings that can faithfully reflect the *meaning* of the embedded text. To motivate this idea and assumption, consider Figure 1. In this figure, we illustrate two *very different* passages of text that still describe the *same* concept, and two *very similar* passages of text that describe two *very different* concepts. Interestingly, the cosine similarities as proposed in CHECKEMBED between the embeddings of two different and two similar passages are – respectively – low and very high, supporting the key idea behind CHECKEMBED. Contrarily, BERTScore and SelfCheckGPT give scores that do not match the similarity of the passages.

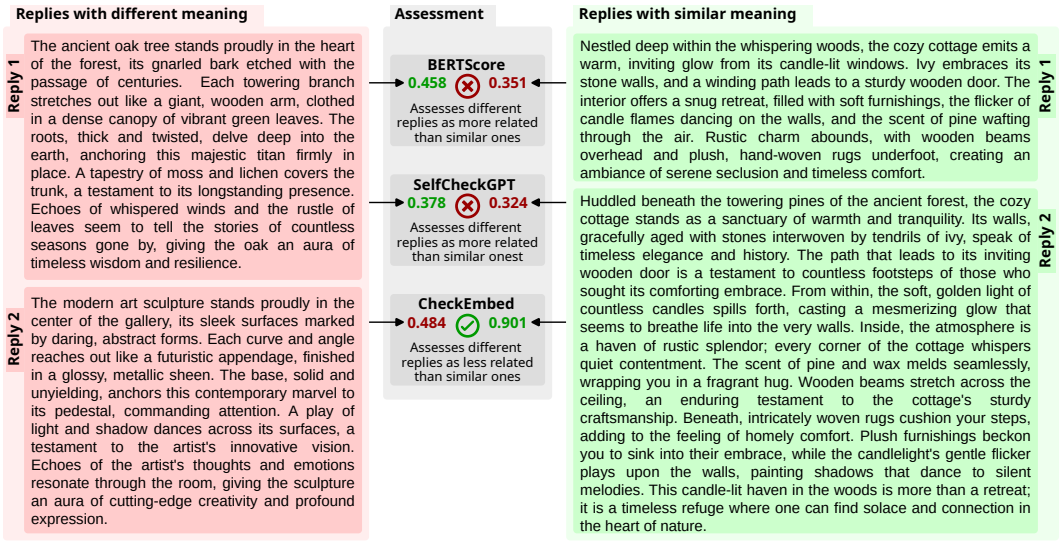

Figure 1: We show two sets of two LLM replies each: Replies explaining different concepts using similar wording (left) and ones explaining similar concepts using different wording (right); the queries used to generate these replies can be found in the Appendix. While BERTscore and SelfCheckGPT assess the semantically unrelated replies as more related than the related ones (because these two baselines have been designed to mostly target the verification of individual sentences or facts), **CHECKEMBED** *correctly differentiates between semantically related and unrelated replies*. We use ChatGPT-4o with temperature = 1.0 for replies and gpt-embedding-large for embeddings.

We design and implement a comprehensive verification pipeline based on CHECKEMBED (**contribution 2**). The pipeline uses the notion of "stability" of the LLM answer, introduced by SelfCheckGPT, for the verification of the considered tasks. The idea behind "stability" is to prompt an LLM to reply to a given question several times. If the LLM repeatedly outputs the same solution, it means that it has high confidence in its answer and the hallucination risk is low (i.e., high stability

of the LLM answers). Contrarily, if there is a large variance in the LLM answers (i.e., low stability of the LLM answers), the risk of hallucinations is high. In CHECKEMBED, we harness this approach for comparing embeddings of whole LLM answers, or their sizeable chunks, pairwise to one another, and to the potential ground-truth (GT), if available. Using such answer-level embeddings enables extracting the *meaning* of a given whole reply and to compare it effectively to others and to GT. We show that this strategy is effective and results in embeddings that are close to each other with respect to different distance metrics in cases where the LLM gives correct answers, and with embeddings that are far away, if the LLM is uncertain of the answer or the answer is not of high quality.

As a part of the CHECKEMBED pipeline, we offer assessment metrics that illustrate both how each of the LLM answers compares to any other answer and to the potential GT, and succinct summaries. The former is provided in the form of embedding heatmaps. The latter are statistical summaries that can be used as user-specified thresholds to drive decision engines in practical deployments on whether a given LLM answer is good enough and can be accepted, or not and thus has to be re-generated.

We apply our verification pipeline that implements the CHECKEMBED idea to several real-world use cases in document analysis, namely extracting terms and definitions as well as summarizing documents (**contribution 3**). In addition to the high accuracy, a large advantage of this approach is its *speed* and *simplicity*: all one has to do is to embed the LLM answers and compare them to one another using cosine similarity or other vector distance measures.

Our evaluation and analysis illustrate high advantages in accuracy and runtimes (**contribution 4**). When the ground-truth is available, CHECKEMBED offers closely matching scores for LLM answers. Specifically, we obtain very high scores for high-quality LLM answers and low scores when the LLM answer is a mismatch. This provides an advantage over comparison baselines that often provide mismatching scores.

2 The CHECKEMBED Design & Pipeline

We now describe the design, the processing and the evaluation of the CHECKEMBED pipeline, which is summarized in Figure 2.

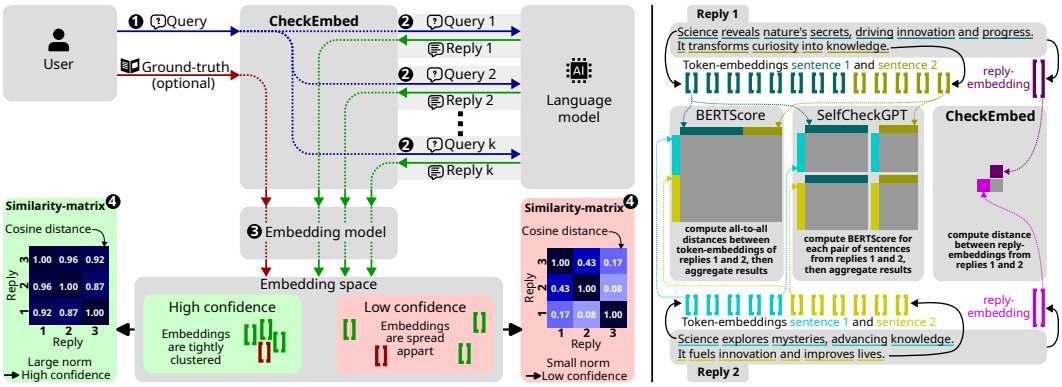

Figure 2: Overview of the CHECKEMBED pipeline (left) and comparison between BERTScore, SelfCheckGPT, and CHECKEMBED (right).

The CHECKEMBED pipeline for verification of LLM's responses consists of the following key parts. First, a user sends a **question to the LLM** ❶ providing all the essential input data. The pipeline enables batching these questions, i.e., it is possible to send multiple questions in the same pipeline and they pass through each of the next stages individually. Next, the pipeline **prompts the LLM several times** ❷ with the same question; this number (k) is a user parameter. Each reply has no prior knowledge of the previous answer guaranteeing that there is no bias. k introduces a tradeoff: more responses (higher k) means more compute time and cost (more tokens used), but also a better check of correctness. However, as we show in Section 4, CHECKEMBED enables high level of confidence in its verification outcome even when k is low. The next stage of the pipeline is the **embedding of the answers** ❸. Each reply is embedded, using a pre-specified embedding model (another user input). The potential ground-truth answer is also embedded. In the final stage, the **embeddings of the replies are compared pairwise** ❹. We use established metrics, most importantly

the cosine similarity; we also experiment with Pearson correlation. Other measures are possible as the pipeline enables seamless integration. The pairwise similarity scores of embeddings are grouped into a (symmetric) heatmap matrix, which is summarized using a selected measure in order to provide a simple threshold number that can be used to drive decision making in practical deployments.

3 Scalability Analysis

We provide a brief scalability analysis showing why CHECKEMBED is fundamentally faster than BERTScore and SelfCheckGPT. We denote the number of answers requested from the LLM with k. We assume the same dimensionality of all used embeddings and that computing a score of two embeddings is negligible and takes $O(1)$ time (e.g., Numpy supports highly efficient Pearson correlation and cosine similarity). Without loss of generality, we also assume that a single reply or the ground-truth contain s sentences, and each sentence contains t tokens. When comparing the baselines, we consider counts of two most compute intense operations within the pipeline: the number of embeddings to be constructed and the number of similarity operations to be conducted.

In CHECKEMBED, there are k embeddings in total to construct, and $O(k^2)$ similarity operations to be conducted.

Next, one can apply BERTScore straightforwardly to two passages treated as long sentences, each such passage consists of st tokens. This means $O((st)^2) = O(s^2t^2)$ embedding comparisons have to be performed for any two passages (for each pair of compared sentences, one compares every pair of individual tokens/words), resulting in a total of $O(k^2s^2t^2)$ embedding comparisons as this is done for $O(k^2)$ pairs of LLM answers, and a total of $O(k^2)$ embedding constructions.

Finally, SelfCheckGPT assesses a given LLM reply by comparing it to all sample replies collected. To simplify the following derivations, assume that in an individual comparison of two LLM replies, these replies consist of s_1 and s_2 sentences, respectively. Now, for each such comparison, SelfCheckGPT uses BERTScore, where the two input passages x and y to BERTScore consist of s_1s_2 sentences each, i.e., both passage x and passage y contain all the sentences from its corresponding LLM reply, repeated as many times as the number of sentences in the other LLM reply (this is condicted to enable comparing all sentences from each reply pairwise). This gives (using the above BERTScore formulae) $O(ks^2)$ embedding constructions (there are k LLM replies) and $O(ks^2s^2t^2) = O(ks^4t^2)$ embedding comparisons.

4 Evaluation

We now show the advantages of CHECKEMBED over the state of the art.

Comparison Baselines We compare CHECKEMBED to two key baselines, **SelfCheckGPT** (a modern verification pipeline that harnesses the stability of LLM replies as its main method) and **BERTScore** (representing embedding based schemes for assessing similarity of text passages).

Considered Models We vary the language models used for both generating replies and for creating embeddings. First, when prompting the LLM to obtain a reply, we explore GPT-3.5, GPT-4, and GPT-4o. Second, when embedding LLM replies, we experiment with different embedding models, namely Salesforce/SFR-Embedding-Mistral (SFR) [33], intfloat/e5-mistral-7b-instruct (E5) [47, 48], Alibaba-NLP/gte-Qwen1.5-7B-instruct (GTE) [22], and GPT Text Embedding Large (GPT) [63]. For BERTScore and SelfCheckGPT, we use the best possible models available for these baselines (i.e., microsoft/deberta-xlarge-mnli [12] and roberta-large [25]). We use the default embedding sizes (listed in the Appendix A.2).

Considered Similarity Measures We consider both cosine similarity and the Pearson correlation score. However, these two follow the same accuracy patterns, and we only show the data for the cosine similarity.

4.1 Distinguishing Similar and Different Text Passages Faithfully

We start the evaluation by extending the motivating example from Figure 1. Specifically, we analyze whether a given verification method is able to clearly distinguish two passages of text that (1) look similar, but come with very different meanings ("Different replies", see the left side of Figure 1 for an

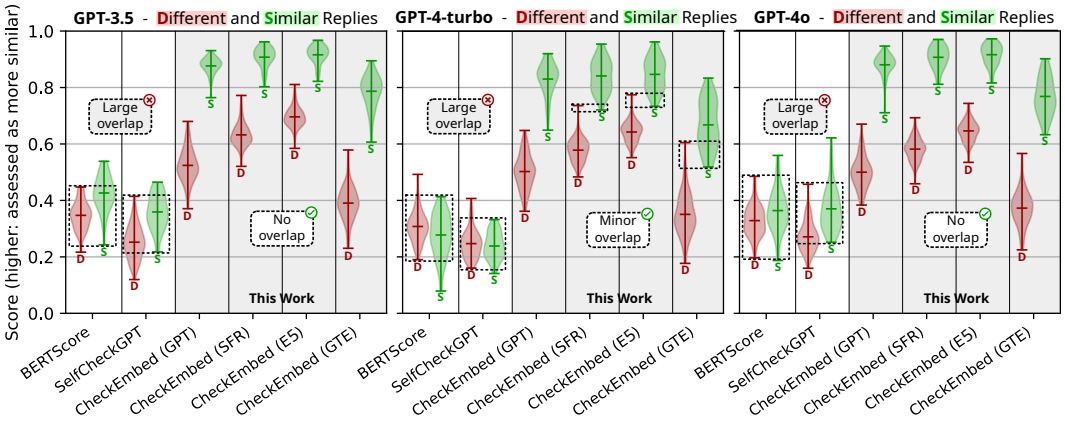

Figure 3: **Analysis of distinguishing similar and different LLM replies (the "Generic" dataset)**, details explained in Section 4.1. CHECKEMBED is highly effective at appropriately recognizing the similarities and differences in the meaning of the verified text passages. This can be seen from little or no overlap between groups of data points corresponding to scores for – respectively – similar and different LLM replies, regardless of the model used. Contrarily, there is a significant overlap between these groups of data points for both BERTScore and SelfCheckGPT, indicating that these baselines perform worse in distinguishing such replies effectively.

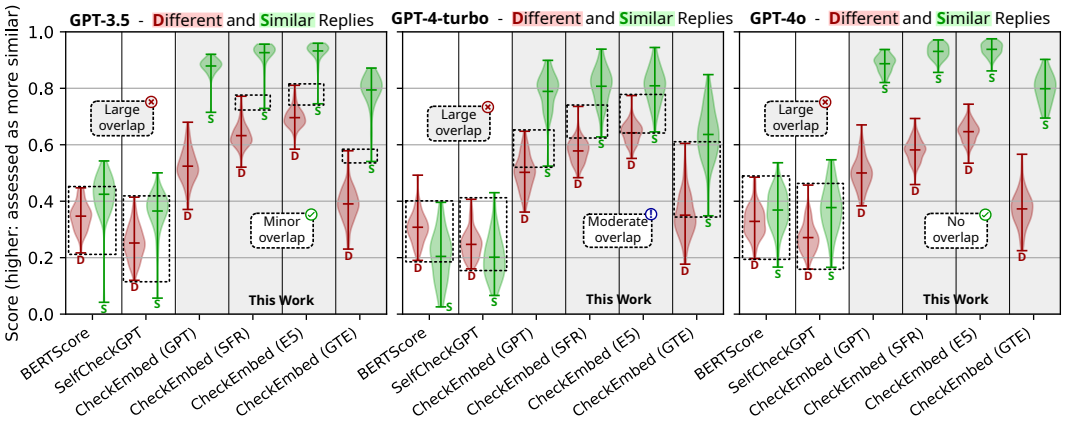

Figure 4: **Analysis of distinguishing similar and different LLM replies (the "Precise" dataset)**, details explained in Section 4.1. CHECKEMBED is effective at appropriately recognizing the similarities and differences in the meaning of the verified text passages. This can be seen from moderate to no overlap between groups of data points corresponding to scores for – respectively – similar and different LLM replies, regardless of the model used. Contrarily, there is a large overlap between these groups of data points for both BERTScore and SelfCheckGPT, indicating that these baselines perform worse in distinguishing such replies effectively.

example), as well as (2) look different, but have similar or identical meanings ("Similar replies", see the right side of Figure 1 for an example). The used prompts can be found in the Appendix A.1. The prompt sizes used for these two groups are in the range of 25–250 and 100–200 tokens, respectively. To broaden the analysis, we further consider two subtypes of such passages: "Generic" and "Precise". The former are brief while the latter are rich in detailed information (e.g., "Vintage bike" vs. "Old, rusted bicycle leaning against a weathered fence"). We illustrate the results for these two subtypes in Figures 3 and 4, respectively.

Importantly, CHECKEMBED comes with *no* (or *very minor*) overlap of scores for similar and different replies. Similar replies come with consistently high similarity scores, while different replies have consistently lower similarity scores. Thus, the **key takeaway** is that CHECKEMBED is highly effective at appropriately recognizing the similarities and differences in the *meaning* of the considered text passages, regardless of their length and style, and also regardless of the harnessed generative and

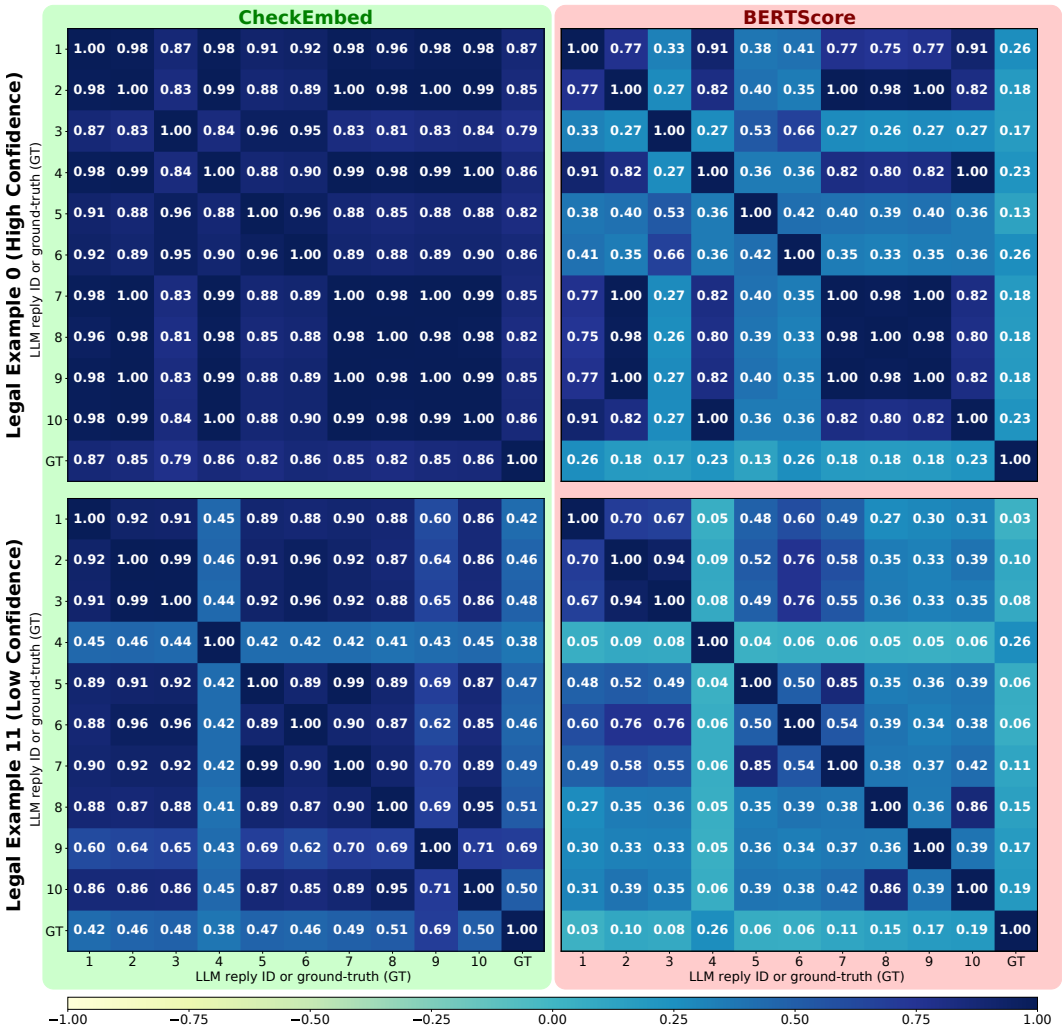

Figure 5: **Analysis of the verification of LLM answers (GPT-3.5)**, details explained in Section 4.2. We compare to BERTScore; SelfCheckGPT comes with significantly higher runtimes (detailed in Section 4.4) and less competitive scores as it does not focus on open-ended answer-level analysis. The results form a heatmap of the CHECKEMBED's, or BERTScore's, cosine similarity between all LLM replies, and between each reply and the human expert prepared ground-truth (GT). Rows correspond to two representative legal documents, that come with – respectively – high and low LLM confidence in its replies. Embedding model used in both rows: GPT Text Embedding Large.

embedding models. Contrarily, both BERTScore and SelfCheckGPT have high overlaps for these passages; thus, CHECKEMBED improves upon the state of the art.

An interesting feature of CHECKEMBED is that, while it *does* distinguish similar and different passages very effectively, it gives *relatively high* scores to the *different* passages; these scores are usually *higher* than the BERTScore or SelfCheckGPT scores for *similar* passages. Despite this, it is still straightforward to distinguish between answers implying similar or different passages, because the CHECKEMBED scores for *similar* passages are *consistently* very high (e.g., with means higher than 0.9 for SFR or E5).

Interestingly, GPT-4-turbo generates replies that are 'the most difficult to distinguish", i.e., it comes with visible (still very low) overlap between similar and different ones, across all embedding models. Contrarily, GPT-4o comes with no overlap whatsoever, while GPT-3.5 has very minor overlap.

Figure 6: **Analysis of the verification of LLM answers (GPT-4)**, details explained in Section 4.2. We compare to BERTScore; SelfCheckGPT comes with significantly higher runtimes (detailed in Section 4.4) and less competitive scores as it does not focus on open-ended answer-level analysis. The results form a heatmap of the CHECKEMBED's, or BERTScore's, cosine similarity between all LLM replies, and between each reply and the human expert prepared ground-truth (GT). Rows correspond to two representative legal documents, that come with – respectively – high and low LLM confidence in its replies. Embedding model used in both rows: GPT Text Embedding Large.

4.2 Verifying LLM Answers Effectively

Next, we illustrate how CHECKEMBED enables effective verification of LLM answers. As a use case, we consider extracting terms and their definitions from legal documents; the used data is real and it comes from an in-house legal analytics project. In this use case, a prompt to the LLM consists of the contents of a legal document (e.g., an NDA), as well as a request to extract respective terms and their definitions. The prompts can also be found in the Appendix A.1. The prompt sizes used in this task are in the range of 25–600 tokens (we split the documents into chunks as whole documents are often very long and come with total token counts that significantly exceed the recommended maximal sizes for the input of the used embedding models). CHECKEMBED asks the LLM to generate 10 replies ($k = 10$). We illustrate the results in Figures 5 (for GPT-3.5) and 6 (for GPT-4). Each figure shows the cosine similarity between all respective LLM replies, and also between each reply and the ground-truth (GT) reply that has been prepared by a human expert.

The results illustrate that whenever CHECKEMBED has very high confidence in its answer (top rows in Figures 5 and 6), which is visible by consistently having very high similarities between different replies, it corresponds to very high similarity scores between the LLM replies and the ground-truth. This is the case for all the considered models. Other baselines show mixed results for individual replies, and low similarities between their replies and GT. It shows that, whenever CHECKEMBED has high confidence it the LLM replies, there is high likelihood that these replies are close to the corresponding GT.

In the bottom rows of both figures, we provide example results where CHECKEMBED indicates low or mixed LLM's confidence. While many scores are still relatively high (e.g., 0.85), many are much lower (e.g., 0.5). We manually verified that these particularly low individual scores correspond to LLM replies of very low quality (e.g., only a single term with its definition has been extracted). The low scores overall indicate model's low confidence, which is further supported by corresponding low similarity scores to GT. Here, BERTScore also has low confidence – overall, its scores are much lower than those of CHECKEMBED, but its relative drop in similarity to GT is similarly as low as that of CHECKEMBED.

Note that the results in the heatmaps directly correspond to the results from Section 4.1 and Figures 3 and 4 in that very high CHECKEMBED scores (e.g., 0.9) indicate high confidence while scores that are lower (e.g., 0.75) – but still higher than BERTScore – consistently mean low LLM's confidence. This indicates that whenever using such baselines together, one may want to consider rescaling the results accordingly.

A useful simple CHECKEMBED measure that indicates the low quality of the LLM answer is a selected summarization measure for a heatmap, for example mean or a matrix norm combined with a standard deviation (std). Whenever the mean is *very high* (e.g., $>$0.9) and the std is *low* (e.g., $<$0.05), the answer is of high quality with very high likelihood. Otherwise, one may want to investigate a given situation in more detail. For example, for GPT-3.5 in the top row (example 0), the LLM is very certain of what the answer is; the mean is 0.93 with very low std of 0.06; BERTScore seems to imply hallucinations with low scores and even more importantly, an std of 0.27.

4.3 Detecting Fine-Grained Hallucinations

While CHECKEMBED is primarily targeted at verification of open-ended tasks, we also investigate whether CHECKEMBED can be used to detect small fine-grained hallucinations, such as mistakes in individual facts. The results are in Figure 7 and 8 and the used prompts can be found in the Appendix A.1. The task analyzed is summarizing scientific and legal articles. For each article considered, we generate a summary with no errors (labeled as "ground truth"), and we also ask the LLM to summarize these documents, while forcing deliberate small fact-level mistakes, from 0 to 10 mistakes per summary. CHECKEMBED is able to recognize when samples contains no errors, as illustrated by very large scores for GT. Moreover, interestingly, it can also recognize hallucinations after introducing a single error, as visible by no overlap between the GT and the consecutive data points. Finally, we can observe that the amount of low-confidence scores is somewhat increasing with the growing number of introduced errors. However, this increase only starts to be distinctive beyond 5 errors. The trends for BERTScore and SelfCheckGPT are similar, which illustrates that these baselines perform well for their intended use case.

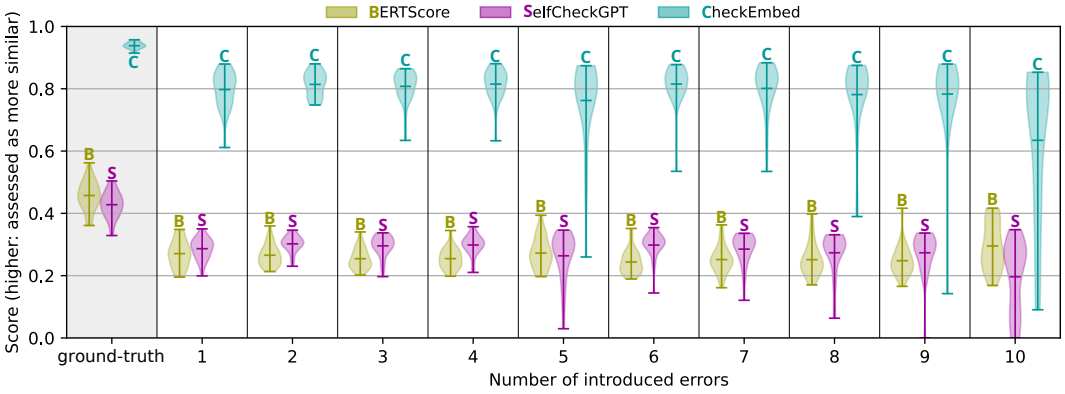

Figure 7: **Analysis of fine-grained hallucination verification of LLM answers (GPT-4o) when summarizing scientific documents**, details explained in Section 4.3.

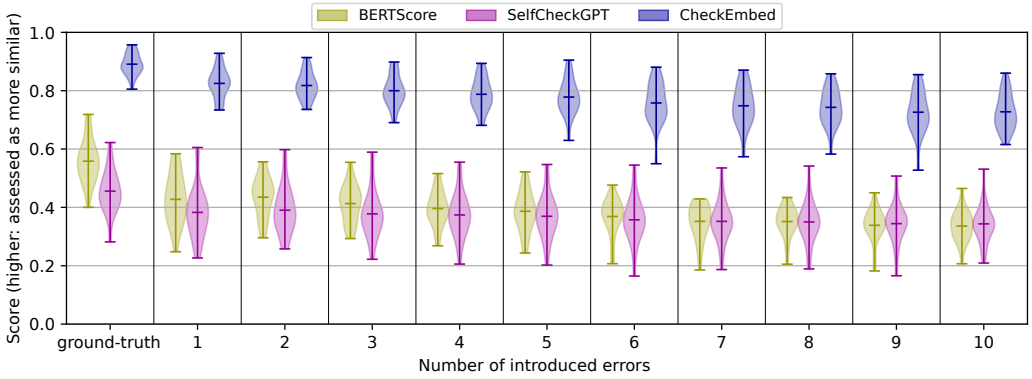

Figure 8: **Analysis of fine-grained hallucination verification of LLM answers (GPT-4o) when summarizing legal documents**, details explained in Section 4.3.

4.4 Lowering Runtimes

Finally, we investigate the running times of all the considered baselines. Example results are in Figure 9. The numbers correspond to the total runtime required to construct 100 embeddings and to compute similarity scores between all embedding pairs. We show runtimes for CHECKEMBED for different embedding models used. Overall, CHECKEMBED comes with much smaller runtimes, even $>30\times$ faster than BERTScore. The somewhat larger runtime of the GPT based CHECKEMBED is caused by the OpenAI API calls. Note, however, that the best available bidirectional embedding models that can be used with BERTScore and SelfCheckGPT are much smaller than the models used by CHECKEMBED (e.g., microsoft/deberta-xlarge-mnli has 750M parameters while Salesforce/SFR-Embedding-Mistral and intfloat/e5-mistral-7b-instruct have $\approx$7B parameters). This further showcases the high performance of CHECKEMBED, rooted in its simplicity: all that is required to compute is a single embedding of a textual answer or its chunk.

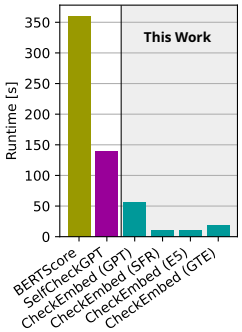

Figure 9: **Comparison of running times of CHECKEMBED and other baselines**

5 Related Work

Trustworthy AI is a broad research area focusing on the transparency, fairness, and reliability of AI systems. Efforts in this field aim to develop frameworks and guidelines that ensure AI

systems are trustworthy and align with human values [26, 43]. Initiatives like differential privacy [3], fairness constraints in machine learning models [16], and transparent reporting of AI capabilities and limitations [23] are prominent in this context. These approaches strive to build AI systems that are not only effective, but also ethically sound and socially acceptable.

Explainable AI (XAI) [28] is another critical area of research with the goal of making AI systems more transparent and interpretable to users. Several works have developed methods to enhance explainability in AI systems [29, 59]. For instance, self-explaining models that generate explanations alongside predictions have been explored to improve user trust and understanding [14, 30]. Other approaches include post-hoc explanation methods, which provide insights into model decisions after predictions are made, thus facilitating better human-AI interaction [18, 45]. These advancements are crucial for deploying AI in sensitive areas where understanding the rationale behind decisions is imperative.

The rise of AI has also prompted methodological and epistemological inquiries. Researchers are examining the foundational questions regarding how AI systems generate knowledge and the implications of these processes [10]. Discussions in this domain focus on the nature of machine learning [40], the validity of AI-generated knowledge [31], and the ethical considerations surrounding AI deployment [21, 37]. These inquiries are essential for framing the theoretical underpinnings of AI and addressing concerns related to bias, fairness, and accountability in AI systems.

The problem of hallucinations in LLMs has gathered significant attention [1, 13, 15, 39, 58]. Chrysostomou et al. [9] find that hallucinations are less prevalent in pruned LLM for summarization tasks, which they attribute to an increased dependence on the original source. Various methods on detecting hallucation have been proposed, including SelfCheckGPT [32], fact checking [8, 55] and others [41, 42, 54]. Another focus is the reduction of hallucinations. Ever [17] dynamically verifies generated content against evidence during the generation process. Zhang et al. [56] propose the use of the human user and knowledge bases to align their knowledge to let the LLM answer truthfully. One of the goals of Retrieval Augemented Generation (RAG) [61] has been hallucination reduction by fetching relevant information for the LLM context. Benchmark efforts have also been proposed [20, 44, 62].

LLM-based agents represent a burgeoning area [51], where LLMs are utilized as autonomous agents to perform complex tasks. These agents leverage the generative capabilities of LLMs to interact with users, perform tasks, and make decisions, often resorting to different prompt engineering techniques [4, 5, 27, 36, 49, 50, 53]. Recent studies focus on enhancing the autonomy and effectiveness of these agents by improving their ability to understand and respond to nuanced user inputs [2]. Techniques such as fine-tuning on specific tasks [7] and incorporating external knowledge sources [11, 24] are employed to enhance the performance of LLM-based agents in real-world applications.

Finally, evaluating LLMs is an ongoing challenge given their complexity and the diverse range of tasks they can perform [34, 60]. Traditional evaluation metrics often fall short in capturing the full spectrum of LLM capabilities. Hence, researchers are developing new benchmarks and evaluation frameworks that better reflect real-world use cases [6]. These include task-specific evaluations, user-centric assessments [46], and adversarial testing [38, 52] to ensure that LLMs perform reliably across different scenarios and are resilient to manipulation.

6 Conclusion

Large Language Models (LLMs) are revolutionizing various domains, yet effective verification for open-ended tasks remains a significant challenge. Established methods, which focus on token- and sentence-level analysis, fall short in scalability and effectiveness. Addressing this gap is crucial as applications of LLMs expand, necessitating robust mechanisms to ensure the accuracy and reliability of their outputs.

To this end, we introduce CHECKEMBED, a scalable approach to LLM verification. CHECKEMBED leverages the effectiveness of answer-level embeddings to compare LLM answers with one another and the potential ground-truth. By transforming complex textual answers into individual embeddings using modern decoder-only based models like GPT Text Embedding Large, CHECKEMBED makes the verification process simple, accurate, and scalable. This straightforward methodology integrates seamlessly with modern data analytics infrastructure, highlighting its practical applicability and ease of deployment.

CHECKEMBED comes with a comprehensive verification pipeline that includes metrics and tools for assessing the veracity of LLM answers, such as heatmaps of similarites between embeddings of answers, the ground-truth, and statistical summaries. These tools provide detailed insights into the quality of LLM outputs and facilitate practical decision-making in real-world deployments. The simplicity of our approach allows for the extension of these metrics to various other applications, further enhancing its utility and flexibility.

Our pipeline has been tested on document analysis tasks, including term extraction. The results demonstrated significant improvements in accuracy and runtime performance compared to existing methods such as BERTScore [57] and SelfCheckGPT [32]. These findings underscore the potential of CHECKEMBED to transform LLM verification in industrial settings, ensuring that LLM outputs are both reliable and scalable.

Acknowledgements

We thank Hussein Harake, Colin McMurtrie, Mark Klein, Angelo Mangili, and the whole CSCS team granting access to the Ault and Daint machines, and for their excellent technical support. We thank Timo Schneider for immense help with infrastructure at SPCL. This project received funding from the European Research Council (Project PSAP, No. 101002047), and the European High-Performance Computing Joint Undertaking (JU) under grant agreement No. 955513 (MAELSTROM). This project was supported by the ETH Future Computing Laboratory (EFCL), financed by a donation from Huawei Technologies. This project received funding from the European Union's HE research and innovation programme under the grant agreement No. 101070141 (Project GLACIATION). We also acknowledge Polish high-performance computing infrastructure PLGrid (HPC Center: ACK Cyfronet AGH) for providing computer facilities and support within computational grant no. PLG/2024/017103.

References

[1] Zechen Bai, Pichao Wang, Tianjun Xiao, Tong He, Zongbo Han, Zheng Zhang, and Mike Zheng Shou. 2024. Hallucination of Multimodal Large Language Models: A Survey. arXiv:2404.18930

[2] Saikat Barua. 2024. Exploring Autonomous Agents through the Lens of Large Language Models: A Review. arXiv:2404.04442

[3] Rouzbeh Behnia, Mohammadreza Reza Ebrahimi, Jason Pacheco, and Balaji Padmanabhan. 2022. EW-Tune: A Framework for Privately Fine-Tuning Large Language Models with Differential Privacy. In *Proceedings of the 2022 IEEE International Conference on Data Mining Workshops* (Orlando, FL, USA) *(ICDMW '22)*. IEEE Press, New York, NY, USA, 560–566. https://doi.org/10.1109/ICDMW58026.2022.00078

[4] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of Thoughts: Solving Elaborate Problems with Large Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence* 38, 16 (March 2024), 17682–17690. https://doi.org/10.1609/aaai.v38i16.29720

[5] Maciej Besta, Florim Memedi, Zhenyu Zhang, Robert Gerstenberger, Nils Blach, Piotr Nyczyk, Marcin Copik, Grzegorz Kwaśniewski, Jürgen Müller, Lukas Gianinazzi, Ales Kubicek, Hubert Niewiadomski, Aidan O'Mahony, Onur Mutlu, and Torsten Hoefler. 2024. Demystifying Chains, Trees, and Graphs of Thoughts. arXiv:2401.14295

[6] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2024. A Survey on Evaluation of Large Language Models. *ACM Trans. Intell. Syst. Technol.* 15, 3, Article 39 (March 2024), 45 pages. https://doi.org/10.1145/3641289

[7] Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. 2024. Agent-FLAN: Designing Data and Methods of Effective Agent Tuning for Large Language Models. arXiv:2403.12881

[8] I-Chun Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, and Pengfei Liu. 2023. FacTool: Factuality Detection in Generative AI – A Tool Augmented Framework for Multi-Task and Multi-Domain Scenarios. arXiv:2307.13528

[9] George Chrysostomou, Zhixue Zhao, Miles Williams, and Nikolaos Aletras. 2024. Investigating Hallucinations in Pruned Large Language Models for Abstractive Summarization. arXiv:2311.09335

[10] Will Fleisher. 2022. Understanding, Idealization, and Explainable AI. *Episteme* 19, 4 (Dec. 2022), 534–560. https://doi.org/10.1017/epi.2022.39

[11] Jian Guan, Wei Wu, Zujie Wen, Peng Xu, Hongning Wang, and Minlie Huang. 2024. AMOR: A Recipe for Building Adaptable Modular Knowledge Agents Through Process Feedback. arXiv:2402.01469

[12] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: Decoding-Enhanced BERT With Disentangled Attention. In *Proccedings of the Ninth International Conference on Learning Representations* (Virtual) *(ICLR '21)*. https://openreview.net/forum?id=XPZIaotutsD

[13] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. arXiv:2311.05232

[14] Shiyuan Huang, Siddarth Mamidanna, Shreedhar Jangam, Yilun Zhou, and Leilani H. Gilpin. 2023. Can Large Language Models Explain Themselves? A Study of LLM-Generated Self-Explanations. arXiv:2310.11207

[15] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of Hallucination in Natural Language Generation. *ACM Comput. Surv.* 55, 12, Article 248 (March 2023), 38 pages. https://doi.org/10.1145/3571730

[16] Tonni Das Jui and Pablo Rivas. 2024. Fairness issues, current approaches, and challenges in machine learning models. *International Journal of Machine Learning and Cybernetics* (Jan. 2024). https://doi.org/10.1007/s13042-023-02083-2

[17] Haoqiang Kang, Juntong Ni, and Huaxiu Yao. 2024. Ever: Mitigating Hallucination in Large Language Models through Real-Time Verification and Rectification. arXiv:2311.09114

[18] Nicholas Kroeger, Dan Ley, Satyapriya Krishna, Chirag Agarwal, and Himabindu Lakkaraju. 2024. Are Large Language Models Post Hoc Explainers? arXiv:2310.05797

[19] Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. NV-Embed: Improved Techniques for Training LLMs as Generalist Embedding Models. arXiv:2405.17428

[20] Junyi Li, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023. HaluEval: A Large-Scale Hallucination Evaluation Benchmark for Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing* (Singapore) *(EMNLP '23)*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Kerrville, TX, USA, 6449–6464. https://doi.org/10.18653/v1/2023.emnlp-main.397

[21] Ni Li. 2023. Ethical Considerations in Artificial Intelligence: A Comprehensive Discussion from the Perspective of Computer Vision. In *Proceedings of the 6th International Conference on Humanities Education and Social Sciences (ICHESS '23)*. SHS Web Conf., Vol. 179. EDP Sciences, Les Ulis Cedex A, France, 04024. https://doi.org/10.1051/shsconf/202317904024

[22] Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards General Text Embeddings with Multi-stage Contrastive Learning. arXiv:2308.03281

[23] Q. Vera Liao and Jennifer Wortman Vaughan. 2023. AI Transparency in the Age of LLMs: A Human-Centered Research Roadmap. arXiv:2306.01941

[24] Iou-Jen Liu, Xingdi Yuan, Marc-Alexandre Côté, Pierre-Yves Oudeyer, and Alexander Schwing. 2022. Asking for Knowledge (AFK): Training RL Agents to Query External Knowledge Using Language. In *Proceedings of the 39th International Conference on Machine Learning*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.). Proceedings of Machine Learning Research, Vol. 162. PMLR, 14073–14093. https://proceedings.mlr.press/v162/liu22t.html

[25] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv:1907.11692

[26] Yang Liu, Yuanshun Yao, Jean-Francois Ton, Xiaoying Zhang, Ruocheng Guo, Hao Cheng, Yegor Klochkov, Muhammad Faaiz Taufiq, and Hang Li. 2024. Trustworthy LLMs: a Survey and Guideline for Evaluating Large Language Models' Alignment. arXiv:2308.05374

[27] Jieyi Long. 2023. Large Language Model Guided Tree-of-Thought. arXiv:2305.08291

[28] Luca Longo, Mario Brcic, Federico Cabitza, Jaesik Choi, Roberto Confalonieri, Javier Del Ser, Riccardo Guidotti, Yoichi Hayashi, Francisco Herrera, Andreas Holzinger, Richard Jiang, Hassan Khosravi, Freddy Lecue, Gianclaudio Malgieri, Andrés Páez, Wojciech Samek, Johannes Schneider, Timo Speith, and Simone Stumpf. 2024. Explainable Artificial Intelligence (XAI) 2.0: A Manifesto of Open Challenges and Interdisciplinary Research Directions. *Information Fusion* 106 (June 2024), 102301. https://doi.org/10.1016/j.inffus.2024.102301

[29] Haoyan Luo and Lucia Specia. 2024. From Understanding to Utilization: A Survey on Explainability for Large Language Models. arXiv:2401.12874

[30] Andreas Madsen, Sarath Chandar, and Siva Reddy. 2024. Are self-explanations from Large Language Models faithful? arXiv:2401.07927

[31] Kyle Mahowald, Anna A. Ivanova, Idan A. Blank, Nancy Kanwisher, Joshua B. Tenenbaum, and Evelina Fedorenko. 2024. Dissociating language and thought in large language models. *Trends in Cognitive Sciences* 28, 6 (March 2024), 517–540. https://doi.org/10.1016/j.tics.2024.01.011

[32] Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing* (Singapore) *(EMNLP '23)*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Kerrville, TX, USA, 9004–9017. https://doi.org/10.18653/v1/2023.emnlp-main.557

[33] Rui Meng, Ye Liu, Shafiq Rayhan Joty, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. 2024. SFR-Embedding-Mistral: Enhance Text Retrieval with Transfer Learning. Salesforce AI Research Blog. https://blog.salesforceairesearch.com/sfr-embedded-mistral/ Accessed: 2024-05-22.

[34] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large Language Models: A Survey. arXiv:2402.06196

[35] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language Models as Knowledge Bases?. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing* (Hong Kong, China) *(EMNLP-IJCNLP '19)*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, Kerrville, TX, USA, 2463–2473. https://doi.org/10.18653/v1/D19-1250

[36] Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2023. Reasoning with Language Model Prompting: A Survey. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL '23)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki

Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 5368–5393. https://doi.org/10.18653/v1/2023.acl-long.294

[37] Petar Radanliev and Omar Santos. 2023. Ethics and Responsible AI Deployment. arXiv:2311.14705

[38] Bhaktipriya Radharapu, Kevin Robinson, Lora Aroyo, and Preethi Lahoti. 2023. AART: AI-Assisted Red-Teaming with Diverse Data Generation for New LLM-powered Applications. arXiv:2311.08592

[39] Vipula Rawte, Amit Sheth, and Amitava Das. 2023. A Survey of Hallucination in Large Foundation Models. arXiv:2309.05922

[40] Murray Shanahan. 2023. Talking About Large Language Models. arXiv:2212.03551

[41] Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Scott Wen tau Yih. 2024. Trusting Your Evidence: Hallucinate Less with Context-aware Decoding. arXiv:2405.14739

[42] Weihang Su, Changyue Wang, Qingyao Ai, Yiran HU, Zhijing Wu, Yujia Zhou, and Yiqun Liu. 2024. Unsupervised Real-Time Hallucination Detection based on the Internal States of Large Language Models. arXiv:2403.06448

[43] Lichao Sun, Yue Huang, Haoran Wang, Siyuan Wu, Qihui Zhang, Yuan Li, Chujie Gao, Yixin Huang, Wenhan Lyu, Yixuan Zhang, Xiner Li, Zhengliang Liu, Yixin Liu, Yijue Wang, Zhikun Zhang, Bertie Vidgen, Bhavya Kailkhura, Caiming Xiong, Chaowei Xiao, Chunyuan Li, Eric Xing, Furong Huang, Hao Liu, Heng Ji, Hongyi Wang, Huan Zhang, Huaxiu Yao, Manolis Kellis, Marinka Zitnik, Meng Jiang, Mohit Bansal, James Zou, Jian Pei, Jian Liu, Jianfeng Gao, Jiawei Han, Jieyu Zhao, Jiliang Tang, Jindong Wang, Joaquin Vanschoren, John Mitchell, Kai Shu, Kaidi Xu, Kai-Wei Chang, Lifang He, Lifu Huang, Michael Backes, Neil Zhenqiang Gong, Philip S. Yu, Pin-Yu Chen, Quanquan Gu, Ran Xu, Rex Ying, Shuiwang Ji, Suman Jana, Tianlong Chen, Tianming Liu, Tianyi Zhou, William Wang, Xiang Li, Xiangliang Zhang, Xiao Wang, Xing Xie, Xun Chen, Xuyu Wang, Yan Liu, Yanfang Ye, Yinzhi Cao, Yong Chen, and Yue Zhao. 2024. TrustLLM: Trustworthiness in Large Language Models. arXiv:2401.05561

[44] YuHong Sun, Zhangyue Yin, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Hui Zhao. 2024. Benchmarking Hallucination in Large Language Models Based on Unanswerable Math Word Problem. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation* (Torino, Italy) *(LREC-COLING '24)*, Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue (Eds.). ELRA and ICCL, Paris, France, 2178–2188. https://aclanthology.org/2024.lrec-main.196

[45] Daniel Vale, Ali El-Sharif, and Muhammed Ali. 2022. Explainable artificial intelligence (XAI) post-hoc explainability methods: risks and limitations in non-discrimination law. *AI and Ethics* 2, 4 (March 2022), 815–826. https://doi.org/10.1007/s43681-022-00142-y

[46] Jiayin Wang, Fengran Mo, Weizhi Ma, Peijie Sun, Min Zhang, and Jian-Yun Nie. 2024. A User-Centric Benchmark for Evaluating Large Language Models. arXiv:2404.13940

[47] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024. Text Embeddings by Weakly-Supervised Contrastive Pre-training. arXiv:2212.03533

[48] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Improving Text Embeddings with Large Language Models. arXiv:2401.00368

[49] Zekun Wang, Ge Zhang, Kexin Yang, Ning Shi, Wangchunshu Zhou, Shaochun Hao, Guangzheng Xiong, Yizhi Li, Mong Yuan Sim, Xiuying Chen, Zhenzhu Zhu, Qingqing Yang, Adam Nik, Qi Liu, Chenghua Lin, Shi Wang, Ruibo Liu, Wenhu Chen, Ke Xu, Dayiheng Liu, Yike Guo, and Jie Fu. 2023. Interactive Natural Language Processing. arXiv:2305.13246

[50] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903

[51] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. 2023. The Rise and Potential of Large Language Model Based Agents: A Survey. arXiv:2309.07864

[52] Xilie Xu, Keyi Kong, Ning Liu, Lizhen Cui, Di Wang, Jingfeng Zhang, and Mohan Kankanhalli. 2024. An LLM can Fool Itself: A Prompt-Based Adversarial Attack. In *Proceedings of the Twelfth International Conference on Learning Representations* (Vienna, Austria) *(ICLR '24)*. https://openreview.net/forum?id=VVgGbB9TNV

[53] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *Proceedings of the Thirty-seventh Annual Conference on Neural Information Processing Systems (NeurIPS '23)*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.). Advances in Neural Information Processing Systems, Vol. 36. Curran Associates, Inc., New York, NY, USA, 11809–11822. https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf

[54] Jiaxin Zhang, Zhuohang Li, Kamalika Das, Bradley Malin, and Sricharan Kumar. 2023. SAC^3: Reliable Hallucination Detection in Black-Box Language Models via Semantic-aware Cross-check Consistency. In *Findings of the Association for Computational Linguistics: EMNLP 2023* (Singapore), Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Kerrville, TX, USA, 15445–15458. https://doi.org/10.18653/v1/2023.findings-emnlp.1032

[55] Jiawei Zhang, Chejian Xu, Yu Gai, Freddy Lecue, Dawn Song, and Bo Li. 2024. KnowHalu: Hallucination Detection via Multi-Form Knowledge Based Factual Checking. arXiv:2404.02935

[56] Shuo Zhang, Liangming Pan, Junzhou Zhao, and William Yang Wang. 2023. The Knowledge Alignment Problem: Bridging Human and External Knowledge for Large Language Models. arXiv:2305.13669

[57] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. In *Proceedings of the Eighth International Conference on Learning Representations* (Virtual) *(ICLR '20')*. https://openreview.net/forum?id=SkeHuCVFDr

[58] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2023. Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models. arXiv:2309.01219

[59] Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. 2024. Explainability for Large Language Models: A Survey. *ACM Trans. Intell. Syst. Technol.* 15, 2, Article 20 (Feb. 2024), 38 pages. https://doi.org/10.1145/3639372

[60] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models. arXiv:2303.18223

[61] Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Haonan Chen, Zhicheng Dou, and Ji-Rong Wen. 2024. Large Language Models for Information Retrieval: A Survey. arXiv:2308.07107

[62] Zhiying Zhu, Yiming Yang, and Zhiqing Sun. 2024. HaluEval-Wild: Evaluating Hallucinations of Language Models in the Wild. arXiv:2403.04307

[63] Juntang Zhuang, Paul Baltescu, Joy Jiao, Arvind Neelakantan, Andrew Braunstein, Jeff Harris, Logan Kilpatrick, Leher Pathak, Enoch Cheung, Ted Sanders, Yutian Liu, Anushree Agrawal, Andrew Peng, Ian Kivlichan, Mehmet Yatbaz, Madelaine Boyd, Anna-Luisa Brakman, Florencia Leoni Aleman, Henry Head, Molly Lin, Meghan Shah, Chelsea Carlson, Sam Toizer, Ryan Greene, Alison Harmon, Denny Jin, Karolis Kosas, Marie Inuzuka, Peter Bakkum, Barret Zoph, Luke Metz, Jiayi Weng, Randall Lin, Yash Patil, Mianna Chen, Andrew Kondrich, Brydon Eastman, Liam Fedus, John Schulman, Vlad Fomenko, Andrej Karpathy, Aidan Clark, and Owen Campbell-Moore. 2024. OpenAI Text-Embedding-Large: New Embedding Models and API Updates. OpenAI Research. https://openai.com/index/new-embedding-models-and-api-updates/ Accessed: 2024-05-17.

A Appendix / supplemental material

A.1 Prompts

Table 1: Prompt template used for the query generation of the "similar description" use case. A list of "Generic" and "Precise" topics is used to replace ### HERE ### with an actual topic. The aim is to generate two passages of text that look different, but are the same content-wise.

```
### INSTRUCTION ###

Hello. Please generate two passages of text. They should both describe the same thing (### HERE
###). However, these two passages should differ VASTLY in their length, style.
I want you to give an answer using the following format:
<formatting>
### DESCRIPTION 1 ###
the actual description here...
### DESCRIPTION 2 ###
the actual description here...
</formatting>

### ANSWER ###
```

Table 2: Prompt template used for the query generation of the "different description" use case. A list of different topics is used to replace ### HERE 1 ### and ### HERE 2 ### with two actual topics. The aim is to generate two passages of text that seem alike, but are completely different content-wise.

```
### INSTRUCTION ###

Hello. Please generate two passages of text. They should describe two different things:
1. ### HERE 1 ###
2. ### HERE 2 ###

However, these two passages should have the same length and style.
I want you to give an answer using the following format:
<formatting>
### DESCRIPTION 1 ###
the actual description here...
### DESCRIPTION 2 ###
the actual description here...
</formatting>

### ANSWER ###
```

Table 3: Prompt template used to "extract respective terms and their definitions" from chunks of legal documentation. Given the complexity of the task, we provide the concrete format as well as an in-context example. [### REPLACE WITH CONTEXT ###] gets replaced by a text chunk from the legal definitions dataset.

INSTRUCTION

You are a lawyer.

QUESTION

Based on the provided context extract all the legal definitions. Answer using the following formatting.
<formatting>
Term.Definition
Term.Definition
...
</formatting>
<example>
[...]
CONTEXT

Preliminary Note
The Stock Purchase Agreement sets forth the basic terms of the purchase and sale of the preferred stock to the investors (such as the purchase price, closing date, conditions to closing) and identifies the other financing documents. Generally this agreement does not set forth either (1) the characteristics of the stock being sold (which are defined in the Certificate of Incorporation) or (2) the relationship among the parties after the closing, such as registration rights, rights of first refusal and co-sale and voting arrangements (these matters often implicate persons other than just the Company and the investors in this round of financing and are usually embodied in separate agreements to which those others persons are parties, or in some cases in the Certificate of Incorporation). The main items of negotiation in the Stock Purchase Agreement are therefore the price and number of shares being sold, the representations and warranties that the Company must make to the investors and the closing conditions for the transaction.
SERIES A PREFERRED STOCK PURCHASE AGREEMENT
THIS SERIES A PREFERRED STOCK PURCHASE AGREEMENT (this "Agreement"), is made as of [], 20[], by and among [_____], a Delaware corporation (the "Company"), and the investors listed on Exhibit A attached to this Agreement (each a "Purchaser" and together the "Purchasers"). The parties hereby agree as follows:

ANSWER

Agreement. THIS SERIES A PREFERRED STOCK PURCHASE AGREEMENT
Company. Delaware corporation
Purchaser. Company or the investors listed on Exhibit A
Purchasers. Company and the investors listed on Exhibit A together
</example>

CONTEXT

[###REPLACE WITH CONTEXT###]

ANSWER

Table 4: Prompt template used for the ground-truth generation query of the "hallucination test" use case. A list of mostly scientific topic is used to replace ### TOPIC ###.

```
### INSTRUCTION ###

Hello. Please generate a passage of text that talks about (### TOPIC ###).

Please, use the following format for answering:
<formatting>
### PASSAGE ###
The passage here....
</formatting>
```

Table 5: Prompt template used for the hallucination generation query of the "hallucination test " use case. A list of mostly scientific topic is used to replace ### TOPIC ###. ### NUMBER ### is replaced according to an user-specified range of numbers. ### ERRORS ### is used during the hallucination generation process, but is removed from the sample output before the embeddings are created.

```
### INSTRUCTION ###

Hello. Please generate ### NUMBER ### completely false information (fact hallucinations) on
(### TOPIC ###).
Then insert the errors inside a passage of text that talks about (### TOPIC ###).
You should convince a reader that the false information are actually correct ones.

Please, use the following format for answering:

<formatting>
### ERRORS ###
List of fact hallucinations to be later included in the passage...
### PASSAGE ###
The passage here....
</formatting>
```

A.2 Embedding Length and Parameter Size

Table 6: Embedding length and number of parameters for each model used during the evaluation.

Model Name	Length	#Parameters
GPT Text Embedding Large	3072	not public
Salesforce/SFR-Embedding-Mistral	4096	7.11B
intfloat/e5-mistral-7b-instruct	4096	7.11B
Alibaba-NLP/gte-Qwen1.5-7B-instruct	4096	7.72B
microsoft/deberta-xlarge-mnli	1024	750M
roberta-large	1024	355M

A.3 Compute Resources

Running the pipeline for the dataset of legal definitions for three LLMs (GPT-3.5, GPT-4 and GPT-4o as well as the baselines SelfCheckGPT and BERTScore) on a single NVIDIA Tesla V100-PCIE-32GB GPU took roughly 90 minutes. That dataset was used to create the heatmap figures 5 and 6. The pipeline for the datasets with similar and different descriptions, used for the violin plots, was executed on the same hardware in around 80 minutes.