Pollice Verso at SemEval-2024 Task 6: The Roman Empire Strikes Back

Konstantin Kobs[†]
anacision GmbH
konstantin.kobs@anacision.de

Jan Pfister[†] and **Andreas Hotho**
Data Science Chair, CAIDAS
University of Würzburg (JMU)
{lastname}@informatik.uni-wuerzburg.de

Abstract

We present an intuitive approach for hallucination detection in LLM outputs that is modeled after how humans would go about this task. We engage several LLM "experts" to independently assess whether a response is hallucinated. For this we select recent and popular LLMs smaller than 7B parameters. By analyzing the log probabilities for tokens that signal a positive or negative judgment, we can determine the likelihood of hallucination. Additionally, we enhance the performance of our "experts" by automatically refining their prompts using the recently introduced OPRO framework. Furthermore, we ensemble the replies of the different experts in a uniform or weighted manner, which builds a quorum from the expert replies. Overall this leads to accuracy improvements of up to 10.6 p.p. compared to the challenge baseline. We show that a Zephyr 3B model is well suited for the task. Our approach can be applied in the model-agnostic and model-aware subtasks without modification and is flexible and easily extendable to related tasks.

1 Introduction

Language Models Are Outstanding, but[2] they can hallucinate, i.e. generate texts that are not supported by the input or the context. Hallucinations can undermine the credibility and usefulness of LLMs, especially for applications that require high accuracy and reliability, such as summarization, question answering, or dialogue. Therefore, there is a pressing need for developing methods to detect and mitigate hallucinations in LLMs, as well as to understand the causes and effects of this phenomenon.

In this SemEval challenge (Mickus et al., 2024), the task is to detect LLM hallucinations based on the input task given to the LLM, the LLM response, and the ground truth answer. Here, the input tasks can be *definition modeling (DM)*, *machine translation (MT)* or *paraphrase generation (PG)*. Each task contains multiple examples that are fed through an LLM and its response is classified as hallucination or not by five annotators. There are two subtasks in this challenge: In the *model-aware* subtask, access to the generating model is given, while in the *model-agnostic* subtask, the generating model is unknown. We approach both subtasks in the same way, by removing the model information from the model-aware subtask. While we are certain that access to the generating model can be beneficial, we argue that the model-agnostic setting has better transferability in practice.

As the competition baseline (which uses SelfCheckGPT by Manakul et al. (2023)), we frame the task of hallucintion detection as a "consistency checking" problem with the given information, where the goal is to check whether an LLM generation is supported by the ground truth. If the generation is not supported by the ground truth, new and thus probably false information must be present; the LLM has hallucinated its response.

For building an intuition for our approach, we imagine the same setting in the real world: A person responds to a question and our goal is to detect if this is a hallucination, i.e., the response is not supported by the truth. With this real-world setting in mind, we formulate three intuitions that we later transfer to the challenge baseline:

(I) Instead of one person, we ask multiple different experts to check the response's consistency with the truth and weight the different expert responses based on their past performance to make a final decision.

(II) Each expert gives a certainty for their response, so we can take this into account

[†] These authors contributed equally to this work.
[1] https://en.wikipedia.org/wiki/Pollice_verso
[2] https://x.com/ChrisGPotts/status/1686802492104028160

Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024), pages 1529–1536

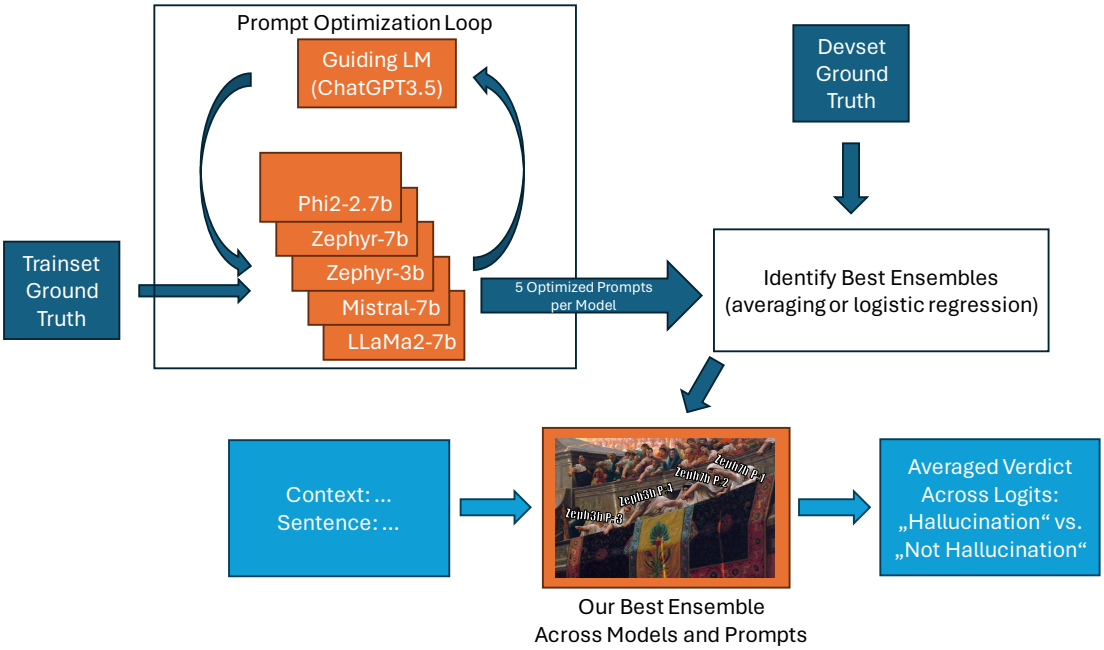

Figure 1: Overview of our approach. We automatically optimize the prompts of multiple "expert" LLMs to check the consistency of the given model output with the ground truth. We combine multiple experts (models-prompt-combinations) in a uniform or weighted manner and select the best ensemble on an internal validation set. This ensemble performs the final collective verdict, as illustrated by a part of Jean-Leon Gerome's painting "Pollice Verso" (1872), which represents the Ancient Roman gesture for judgment on defeated gladiators[1].

when combining multiple experts' responses into one final judgement.

(III) Since each expert is trained differently and has different skills, a suitable explanation of the task to the expert is beneficial.

2 System Description

Given this intuition regarding a potential approach to the task in the real world, we now connect these steps to our submitted system. An overview is found in Figure 1.

(I) We construct ensembles of multiple LLMs that are independently asked to check the response's consistency with the truth. For each ensemble of models, we combine the responses of the LLMs using averaging or by training a logistic regression (see Section 2.5).

(II) Compared to the task baseline method, we modify the procedure for how the LLMs produce their output to obtain better probability estimates (see Section 2.3).

(III) In addition to the baseline prompt, we use an automatic prompt optimization technique to create five additional, well-working prompts and use them as additional options for our ensemble selection (see Section 2.4).

2.1 Provided Baseline

In general, our system is based on the baseline provided by the task organizers. Here, a Mistral 7B model is given the following prompt:

```
Context: [GROUNDTRUTH]
Sentence: [MODELOUTPUT]
Is the Sentence supported by the context
above? Answer using ONLY yes or no:
```

The next token is then generated by the LLM and checked whether it is "yes" or "no" (possibly with additional whitespace or capitalization). If it is a "yes", the output label is set to "Not Hallucination" and its log-probability $logprob$ is converted to the probability $p(Hallucination) = 1 - e^{logprob}$. If it is a "no", the output label is set to "Hallucination" and its log-probability $logprob$ is converted to $p(Hallucination) = e^{logprob}$. If the next token is neither a "yes" or "no", the output label is chosen randomly and $p(Hallucination)$ is set to 0.5.

In the following, we apply our three intuitions to the baseline setup in order to achieve better results. To have a better understanding of which datasets

are used, we introduce them in the following section.

2.2 Data Splits

The task organizers provide multiple datasets from which we use the labeled "val.model-agnostic.json" and "val.model-aware.v2.json" files. We randomly split the model-agnostic file into two equal datasets, stratified by the task, such that both datasets have similar numbers of items with the same task.

These two datasets are our "training" and "validation" datasets, respectively. The full model-aware file is used as our internal "test" dataset, in order to estimate the performance of our system without making a submission.

The "training" dataset is used to optimize the prompts of the models. The "validation" dataset is used to train logistic regressions to weight each member of the ensemble. The "test" dataset is then used to evaluate our results internally without submitting all ensembles of models to the competition leaderboard.

2.3 Intuition (II): Better Output Generation

The task baseline from the organizers generates the most probable next token and checks if it is "yes" or "no". When running the task baseline code, we find that in 21 of the 499 examples from the "val.model-agnostic.json" file, the model does not return a "yes" or "no" directly as the highest scoring token, which means that the output label is chosen randomly and "P(Hallucination)" is set to 0.5.

We argue that it is not necessary to rely on the model to generate a "valid" token at the beginning and only hope for a definite answer. Instead, we take the models' output probability distribution over all available tokens. From this, the combined probability of relevant tokens can be accessed and computed, so even if "yes" or "no" are not the most probable tokens, a definite answer can be derived.

Let $\mathbf{T}$ be a mapping of all available LLM tokens to their corresponding log probabilities, which is accessed using $\mathbf{T}[x]$ for token x. Out of the LLM vocabulary, we identify tokens indicating a positive and negative reply, i.e. all tokens that boil down to "yes" or "no" in any capitalization and with any added whitespace. We call the sets of tokens $\mathcal{P}$ and $\mathcal{N}$ for positive and negative tokens, respectively. The probability for the answer being positive is then computed using a modified softmax function s, which takes only the positive and negative tokens into account:

$$s(\mathbf{T}) = \frac{\sum_{p \in \mathcal{P}} \exp(\mathbf{T}[p])}{\sum_{t \in (\mathcal{P} \cup \mathcal{N})} \exp(\mathbf{T}[t])} \quad (1)$$

This way, even if the token with the highest probability is not a "yes" or "no", a meaningful probability $s(\mathbf{T}) \in [0, 1]$ can be computed. This makes our system more reproducible than the organizer's baseline code, since no randomly selected labels can occur. The predicted output label is then "Not Hallucination" when $s(\mathbf{T}) \geq 0.5$ and "Hallucination" else.

2.4 Intuition (III): Prompt Optimization

We further improve the performance of our approach by "finetuning" the used prompts for each model we use in our ensemble independently. To this end, we follow the OPRO approach (Yang et al., 2023), which aims to automatically optimize the prompts with the help of a "guiding" language model. First, we take the baseline prompt as an initial starting prompt and evaluate the accuracy of the model on a split of our training dataset. Next, the guiding language model, in turn, is prompted to optimize the prompt that the model is acting upon. For this, it has access to the 20 best previously evaluated prompts, as well as the accuracy the model achieved when using this prompt. The task of the guiding language model is now to generate a new prompt that outperforms all previous prompts. Finally, the new prompt is evaluated and added to the list of tried prompts for the next optimization step.

We employ this approach to optimize the prompts for every used model separately, as the optimal prompt for one model does not have to be working well for other models. As guiding language model we select ChatGPT3.5-Turbo[3] and evaluate the prompts on our holdout set. We slightly adapt the original OPRO optimization prompt as we found the "meta" prompt submitted to the guiding language model to be very hard to decipher in our case. This stems from nested references to "below instructions" which in turn referenced the "Context above", although the samples are usually appended. As there are no clearly reserved delimiters in this prompt, this can be confusing when reading this optimization prompt — even for a human. Hence we slightly change this prompt by introducing a json-structure where fit (empty newlines have been stripped here):

[3] https://platform.openai.com/docs/models/gpt-3-5-turbo

Table 1: The best five prompts found by the prompt optimization method OPRO for the Zephyr 3B model. Overall, the optimization was run for ten iterations. Shown are also the iteration in which they were proposed and their respective accuracies on our holdout set.

Iter.	Acc.	Prompt
2	0.714	Decide whether the given sentence is directly supported by the provided context. Answer with a simple "yes" or "no".
8	0.714	Determine if the sentence provided is supported by the given context. Respond with a clear "yes" or "no".
1	0.694	Based on the given context and sentence, determine if the statement is supported or not. Please respond with a simple yes or no.
1	0.694	Based on the given context, determine if the sentence is correctly supported. Respond with a simple 'yes' or 'no'.
1	0.694	Is the Sentence consistent with the provided Context? Answer with either "yes" or "no".

```
Your task is to generate the instruction
<INS>. Below are some previous
instructions with their scores. The
score ranges from 0 to 100.
[
 {
  "<INS>": "Is the Sentence supported by
the Context above? Answer using ONLY yes
or no:",
  "score": 74
 },
 ...
]
Below are some problems commonly
solved incorrectly when using above
instructions.
[three incorrectly solved examples and
their correct label formatted as json]
Generate an instruction that is different
from all the instructions <INS> above,
and has a higher score than all the
instructions <INS> above. The instruction
should begin with <INS> and end with
</INS>. The instruction should be concise,
effective, and generally applicable to
all exemplary problems above.
```

Table 1 shows the five best performing prompts for the Zephyr 3B model found by OPRO.

2.5 Intuition (I): Ensemble Strategy

Our intuition is to ask multiple experts instead of one to assess whether the model output is hallucinated. This is implemented as an ensemble approach, where different models are asked to identify hallucinations. Their outputs are later combined to one output label and probability.

Considered Models Since there are plenty of open source language models available, we limit ourselves to five different models. We select these models from the "New & Noteworthy" section of LM Studio, a desktop application that allows to run LLMs efficiently on CPUs using quantized model weights.[4] We define several criteria to select our final models:

- In order to keep inference times low, we only consider models smaller than or equal to 7B parameters.
- To make use of the newest models, the selected models can at most be half a year old. Based on our selection date of January 10, 2024, the models have to be released (according to LM Studio) after July 10, 2023.
- We only take the newest version of a model (e.g. Mistral v0.2 is used instead of v0.1).
- We try to diversify the model architectures and training datasets by eliminating mostly Llama 2/Mistral finetuned models.
- We select general purpose LLM for the English language, i.e., no explicit code generation models or models for creative writing.

This selection process gives five models for which we download their weights in the "Q6_K"[5] quantized version: Phi 2[6], Mistral 7B Instruct v0.2[7], StableLM Zephyr 3B[8], Zephyr 7B β[9], Llama 2 7B Chat[10].

[4] The list of models can be found at https://github.com/lmstudio-ai/model-catalog/tree/205a13027c9fcd7d0c4a1874d6bb0ae45922deee/models (accessed: 2024-01-10)

[5] more information can be found here https://github.com/ggerganov/llama.cpp/pull/1684

[6] https://hf.co/TheBloke/phi-2-GGUF

[7] https://hf.co/TheBloke/Mistral-7B-Instruct-v0.2-GGUF

[8] https://hf.co/TheBloke/stablelm-zephyr-3b-GGUF

[9] https://hf.co/TheBloke/zephyr-7B-beta-GGUF

[10] https://hf.co/TheBloke/Llama-2-7B-Chat-GGUF

Table 2: Which ensembles we searched for and how we found them. Missing ones from this pattern are duplicates from other ensembles. Note the absence of LLaMas, Mistrals and Phis

				Validation		Test		Official Task Results			
								model-agnostic		model-aware	
set	**metr**	**agg**	**models**	acc	rho	acc	rho	acc	rho	acc	rho
val	acc	single	Zeph 3B P:0	0.748	0.585	0.739	0.603	0.783	0.655	0.750	0.601
val	rho	single	Zeph 3B P:4	0.736	**0.619**	0.743	**0.622**	0.776	0.687	0.747	0.597
val	acc	mean	Zeph 7B P:3 + Zeph 3B P:3	0.772	0.573	**0.766**	0.595	0.797	0.658	**0.777**	0.601
val	rho	mean	Zeph 3B P:4	0.736	**0.619**	0.743	**0.622**	0.776	0.687	0.747	0.597
val	acc	logreg	LLaMa2 7B P:3 + Mistr 7B P:0 + Zeph 3B + Zeph 7B P:0 + Zeph 7B P:2	**0.780**	0.510	0.741	0.540	0.793	0.594	0.763	0.519
val	rho	logreg	Zeph 3B P:4	0.736	**0.619**	0.743	**0.622**	0.777	0.687	0.747	0.597
test	acc	single	Zeph 7B P:2	0.708	0.490	0.749	0.482	0.746	0.525	0.743	0.418
test	rho	single	Zeph 3B	0.732	0.597	0.727	0.622	0.756	**0.690**	0.735	0.590
test	acc	mean	Zeph 3B P:3 + Zeph 3B P:4 + Zeph 7B P:1 + Zeph 7B P:2	**0.756**	0.560	**0.772**	0.585	0.799	0.646	0.772	0.560
test	rho	mean	Zeph 3B + Zeph 3B P:4	0.740	0.615	0.729	**0.626**	0.769	0.689	0.744	0.598
test	acc	logreg	Zeph 3B P:4 + Zeph 3B + Zeph 7B P:0 + Zeph 7B P:1 + Zeph 7B P:3	0.740	0.589	**0.772**	0.603	**0.803**	0.676	0.771	**0.602**
test	rho	logreg	Zeph 3B + Zeph 3B P:4	0.744	**0.617**	0.737	0.626	0.775	0.689	0.745	0.598
—	—	—	Mistral 7B (organizer's baseline)	0.644	0.338	0.695	**0.462**	0.697	0.403	0.745	0.488
—	—	—	Mistral 7B (organizer's baseline) with better output generation	**0.648**	**0.380**	**0.707**	0.452	—	—	—	—

Combining LLM Responses For each of the five models, we test overall six prompts: The baseline prompt from the organizers as well as the five best performing prompts found by OPRO. For each prompt, we compute the output labels ("Hallucination" or "Not Hallucination") and probabilities "p(Hallucination)" for the validation and test datasets. Given these 30 outputs per dataset, we combine all subsets of up to five model responses by either averaging all hallucination probabilities (*mean*) or training a logistic regression on the validation dataset (*logreg*) for a more sophisticated combination. We then evaluate all combinations on our validation and test datasets.

3 Results

We overall have three dimensions in which we can select the best model/prompt ensemble for challenge submission:

1. validation (*val*) vs. test dataset (*test*) results
2. accuracy (*acc*) vs. correlation (*rho*) as evaluation metrics
3. mean ensemble (*mean*) vs. logistic regression ensemble (*logreg*) vs. single model (*single*)

The resulting model selections as well as the organizer's baseline with their metrics for our validation and test datasets as well as the official task results are shown in Table 2. Besides the model names, if a different prompt than the baseline prompt has been used, we encode the prompt used for the model in its name ("P:0" to "P:4" with "P:0" being the automatically optimized prompt that performed best on our holdout set). We can make multiple observations in the results table.

First, the last two rows of the table show that our proposed output generation method mostly improves the baseline scores slightly. Second, all of our model ensembles are better than the organizer's baseline, showing that the ensemble and logistic regression implementations help in this setting. Note that sometimes, the best model ensemble for different dimensions is the same, e.g. Zephyr 3B P:4 is the best model when evaluated on the validation correlation, regardless of the ensemble strategy.

Third, nearly all selected model ensembles contain the Zephyr 3B model in some form, which is surprising, as the most other models have more than twice the parameters. This shows that in this task, the model size does not correlate with performance. Overall, both Zephyr (3B and 7B) models are well-suited for the task, even though they are not trained by the same companies and thus not directly related.

In terms of official task results, the model ensemble chosen by the best test accuracy (and then by correlation) shows the best accuracy of our submissions on the model-agnostic task (80.3%). It is an ensemble consisting of two versions of the Zephyr 3B model as well as three versions of the Zephyr 7B model, combined through a logistic regression. It ranks 41 out of all 260 submissions for the model-agnostic task of the challenge.

For the model-aware task, our best model ensemble again consists of a Zephyr 3B and 7B version. These models are combined using the mean ensemble strategy, leading to 0.777 accuracy as official results. This submission ranks 103 out of 295 task submissions. Given that our approach is completely model-agnostic and thus not specifically designed for this subtask, the results are fairly good.

4 Analysis

We present a comparative analysis of the predicted versus actual probabilities of hallucination across the three tasks given in the challenge datasets. To this end, we plot the predictions for the ensemble that achieved the highest accuracy on our test set (test/acc/logreg in Table 2). In Figure 2 the overall gold label distribution for the probability of hallucination (p(Hallucination)) is depicted on the left, contrasting starkly with the more extreme predicted label distribution shown in the right plot. This polarized nature of predictions suggests a tendency for our model to forecast outcomes with heightened certainty, a trait observable across all ensemble models examined.

Particularly noteworthy is the paraphrase task's label distribution, which significantly deviates from the other tasks. This unique distribution is reflected not only in the ground truth data but also in our model's predictions, indicating a consistent model response to the characteristics of this task.

Figure 3 underscores a clear positive correlation between the predicted probabilities of hallucination and the ground truth scores for all task types. Hallucination detection on the machine translation task exhibits the strongest correlation, suggesting a higher predictive performance, whereas for definition modeling and paraphrase generation the scores correlate slightly less closely.

5 Discussion and Future Work

Our proposed system is easy to understand and implement, can be applied both in model-agnostic and model-aware scenarios, and is flexible in terms of different metrics: By using smaller and fewer models in the ensemble, the runtime can be achieved. By choosing the models based on a given metric, the performance given this metric can be optimized.

In the following, we want to discuss two areas: Runtime optimization and task realism. The runtime of our system depends on the number of models. Here, we have shown that small and single models can already lead to very good results. Currently, we only use one type of quantization for all models. Exploring the effects of different quantization methods might be interesting, since usually, with higher quantization, the model size gets smaller and the model gets faster, but performance degrades. Since the Zephyr 3B model is the best model for this task, maybe another quantization can optimize the runtime of our system even further. Additionally, since most of our models contain multiple versions of a prompt for the same model, we could employ batching of these prompts. This could also reduce the runtime of the system.

Currently, our prompt that is fed into the models contains the model output as well as the ground truth. The model then is instructed to check whether the model output is grounded in the ground truth. This approach follows the baseline provided by the organizers. Consequently, our prompt optimization only uses these two inputs as well. It might be interesting to evaluate, whether using the model input as an additional prompt input can increase the performance of the system.

Overall, we argue that in a realistic use case of our system, the ground truth is not known when checking for hallucinations. Instead, the system should check whether the provided model output is a correct answer to the model input. Since our approach is very flexible, it is possible to enable this use case in our system by altering the model prompt to contain both the model input and its generated output, removing the ground truth. Then, our system could be used as a validator step after the output of a LLM, which catches hallucinated inputs or at least outputs the "p(Hallucination)" score along with the LLM output.

6 Related Work

Hallucination detection and automated prompt optimization in LLMs are both vivid research topics, which became popular with the high demand for reliable LLM applications.

Hallucination Detection We mainly follow the survey by Huang et al. (2023) who categorize hallucinations in LLMs into two main categories: Factuality Hallucination, where the model output is factually incorrect, and Faithfulness Hallucination, where the model output might be correct but does not follow the user's directives or does not take provided context into account. For both types of hallucinations, there is research to detect them. External knowledge can help with identifying Factuality Hallucinations, since the model output can be checked against verified knowledge sources. In this challenge, external knowledge in form of the ground truth answer is given. Uncertainty estimation of the model output can also help with identifying Factuality Hallucinations, since the model is usually not certain when producing wrong output. For this, some methods use access to the model to identify

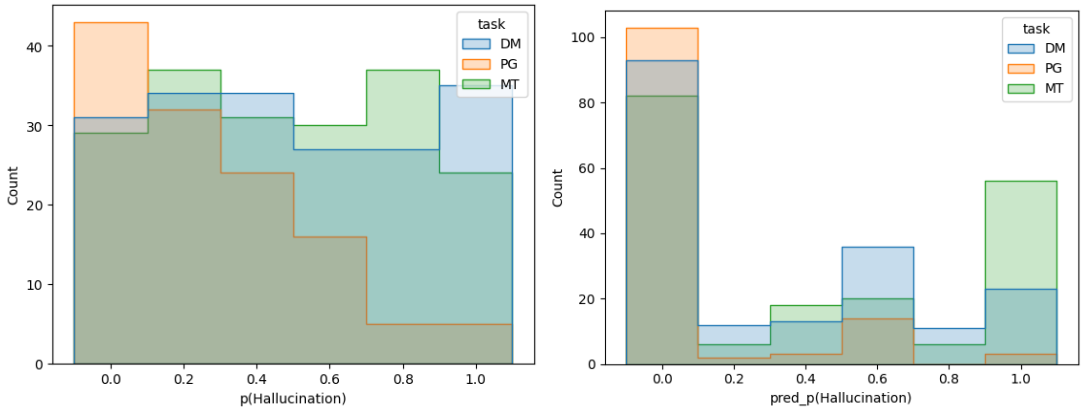

Figure 2: Gold label distributions vs. model predictions (left and right, respectively), with distinct behaviors observed for the different tasks.

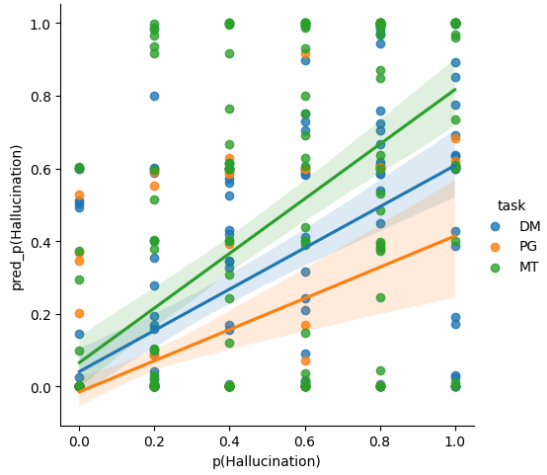

Figure 3: Positive correlation between predicted and actual hallucination probabilities for all tasks.

the uncertainty, other methods use the behavior of the model as an uncertainty indicator. The latter methods thus are model-agnostic.

For Faithfulness Hallucination, different metrics based on different methods such as word overlap or neural classifiers have been proposed. They try to identify logical misbehavior in the model output given the provided context by estimating the semantic or logical difference between the model input and its output. The idea is that Faithfulness Hallucinations stem from models not following the provided input and thus, the generated output is less consistent with its input than when the model does not hallucinate.

Prompt Optimization In the domain of prompt optimization for enhancing the reasoning capabilities of LLMs, a variety of strategies have emerged, notably in efforts to refine these models' performance through advanced prompting techniques (Qiao et al., 2023). Among these strategies, two prominent methods have recently gained attention for their novel approach to automatic prompt optimization, both leveraging a "guiding language model". This model, by having insight into the LLM's predictions, iteratively refines the prompt to achieve optimal outcomes.

The methodology introduced by Pryzant et al. (2023) draws inspiration from the principles of gradient descent and backpropagation. Initially, the guiding language model reviews the existing prompt alongside the LLM's errors, tasked with pinpointing specific shortcomings within the prompt — akin to identifying textual gradients. Following this, the same model is prompted to propose modifications that could rectify these identified issues, mirroring the process of backpropagation. This cycle of evaluation and refinement continues until the process reaches a state of "convergence".

Conversely, OPRO (Yang et al., 2023) simplifies this procedure by equipping the guiding language model with a repository of previously tested prompts and their corresponding effectiveness scores, in addition to a set of example problems. This repository provides the model with a richer context for decision-making, enabling it to discern more effectively between more and less successful prompts with each optimization iteration (as described in Section 2.4). This approach allows for a more informed and potentially more efficient refinement process.

7 Conclusion

We have introduced our system to detect hallucinations in LLMs using an ensemble strategy over multiple LLMs to decide whether the provided model output is hallucinated or not. Here, we employ a softmax over multiple relevant tokens to better capture the certainty of the models. We also employ an automatic prompt optimization scheme that finds well working prompts for each ensemble member.

We have found that the small Zephyr 3B model performs very well on this task, which motivates future exploration of its capabilities and reasons for them. Due to its simplicity, our system can be easily extended to new ensemble models and prompt templates as well as applied to new tasks, such as hallucination detection without access to ground truth.Future work might explore runtime optimizations such as batching or different model quantizations. Finally, instead of independently optimizing the model prompts, future work might jointly optimize all prompts for an ensemble, leading to different experts for different tasks.

Acknowledgements

This work is partially supported by anacision GmbH and the MOTIV research project funded by the Bavarian Research Institute for Digital Transformation (bidt), an institute of the Bavarian Academy of Sciences and Humanities. The authors are responsible for the content of this publication.

References

Jean-Léon Gérôme. 1872. Pollice verso. Oil on canvas. Phoenix Art Museum, Phoenix, Arizona. Retrieved from https://en.wikipedia.org/wiki/Pollice_Verso_(G%C3%A9r%C3%B4me).

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.

Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models.

Timothee Mickus, Elaine Zosa, Raúl Vázquez, Teemu Vahtola, Jörg Tiedemann, Vincent Segonne, Alessandro Raganato, and Marianna Apidianaki. 2024. Semeval-2024 shared task 6: Shroom, a shared-task on hallucinations and related observable overgeneration mistakes. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 1980–1994, Mexico City, Mexico. Association for Computational Linguistics.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968, Singapore. Association for Computational Linguistics.

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)*, pages 5368–5393, Toronto, Canada. Association for Computational Linguistics.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2023. Large language models as optimizers.