Using Combinatorial Testing for Prompt Engineering of LLMs in Medicine

Alexander Perko
Iulia Nica
Franz Wotawa
alexander.perko@ist.tugraz.at
inica@ist.tugraz.at
wotawa@ist.tugraz.at
Graz University of Technology, Institute of Software Technology
Graz, Austria

Abstract

Large Language Models (LLMs) like GPT-4o are of growing interest. Interfaces such as ChatGPT invite an ever-growing number of people to ask questions, including health advice, which brings in additional risks for harm. It is well known that tools based on LLMs tend to hallucinate or deliver different answers for the same or similar questions. In both cases, the outcome might be wrong or incomplete, possibly leading to safety issues. In this paper, we investigate the outcome of ChatGPT when we ask similar questions in the medical domain. In particular, we suggest using combinatorial testing to generate variants of questions aimed at identifying wrong or misleading answers. In detail, we discuss the general framework and its parts and present a proof-of-concept utilizing a medical query and ChatGPT.

Keywords

Large Language Models, ChatGPT, Prompt Engineering, Combinatorial Testing, Validation

1 Introduction

The use of LLMs in medicine has been of growing interest. In a recent publication [9], the authors discuss the future of LLMs in medical applications. Although using such a model may lead to improved communication and other advantages, some drawbacks prevent using such models and tools. It is well known that LLMs like ChatGPT [33] have shortcomings like hallucinations [46]. Hallucinations are answers with incorrect claims that do not depend on training data. Such answers, unfortunately, cannot be necessarily identified as wrong and, therefore, might be harmful, especially when dealing with medical questions. There are methods for detecting hallucinations, e.g., see [11]. Hence, verifying and validating tools based on LLMs to ensure a harmless use is of utmost importance.

When using LLMs for queries, the form of the query, i.e., the prompt, is of great importance. Although there has been much work on how to improve writing prompts in various setups, e.g., [28], there is only little scientific work, e.g., [24], providing statistical evidence. However, it is generally agreed that the query's structure has a significant impact on the output of a LLM. Therefore, we need to consider different prompts in any verification and validation procedure.

Information Society 2024, 7–11 October 2024, Ljubljana, Slovenia

https://doi.org/https://doi.org/10.70314/is.2024.chtm.10

In this paper, we contribute to verifying and validating LLMs focusing on the impact of prompts. In particular, we introduce and discuss a methodology based on combinatorial testing [21] for generating various versions of prompts for medical queries. We rely on testing because it is a methodology to straighten the view on finding interactions with a system under test (SUT) that leads to unexpected behavior. Hence, in testing, we want to provide interactions that make a system fail. Combinatorial testing is a test methodology that has proven to be effective in finding test cases, i.e., inputs given to a system, to provoke a failing behavior. In particular, combinatorial testing focuses on interacting parameter values that reveal faults. In previous work, Kuhn and colleagues [22] showed that strictly less than 7 interacting parameters must be considered for many applications.

Besides its effectiveness, combinatorial testing is a good testing methodology for LLMs that consider prompts. For the latter, we need different combinations of textual fragments to show differences in the outcome. Combinatorial testing provides such combinations and also avoids leading to a combinatorial explosion of potential prompts when restricting the number of considered fragment interactions.

We organize the paper as follows: We first introduce the foundations. For this purpose, we discuss related research on testing LLMs, and introduce the basic concepts behind combinatorial testing. Afterward, we introduce the general testing methodology for generating different prompts focusing on the medical domain. In addition, we illustrate the use of the methodology considering one particular medical query. Finally, we conclude the paper.

2 Related Research

In the past several years, considerable efforts have been made to evaluate LLMs. The first indicator is the wide variety of *benchmarks* which have emerged in order to test and compare their performance on various tasks. In [6], the authors compile a selection of 46 popular benchmarks. Among them, we can differentiate between benchmarks used for general language tasks, like Chatbot Arena [7], MT-Bench [49], HELM [25], or MMLU [13] and domain-specific benchmarks, like MATH [14], concentrated on assessing reasoning and problem-solving capabilities of AI models in mathematics, APPS [15] for evaluating code generation, or MultiMedQA [41] with focus on medical examinations, medical research, and consumer healthcare questions.

Further on, depending on the human involvement in the evaluation process, there are two common methods: *human evaluation* and *automatic evaluation*. Human evaluation becomes a natural choice in many non-standard cases, where the automated evaluation metrics are either not suitable or insufficient. For example, in [25], the evaluators analyze summarization and disinformation

scenarios, while in [2] analogical reasoning tasks. Also, Ziems et al. [50] used the annotations from researchers for generation. Although it can provide more comprehensive and accurate feedback than the automatic evaluation, the human evaluation may suffer from a high variance and instability due to cultural and individual biases. On the other hand, the automatic variant benefits from higher standardization. LLM-EVAL [26], for instance, is a unified multidimensional automatic evaluation method for open-domain conversations with LLMs. Jain et al. [18] proposed a self-supervised evaluation framework, and also PandaLM [45] obtained reproducible and automated language model assessment by training an LLM that acts as the 'judge' to assess different models. For more details on the specific key metrics and factors for both evaluation types, we refer the interested reader to [6].

In addition, *domain-specific evaluation* is also critical, as LLMs are often used in specific areas, such as healthcare or finance, that have specific requirements for the models. In the beginning, most evaluation research has been focused on natural language tasks. A popular direction here is, for instance, the sentiment analysis task, which analyzes and interprets the text to identify the emotional inclination. Further on, in the medical field, the application of LLM has recently gained significant attention. According to the literature ([6], [5]), most LLMs evaluations in the medical field focus on medical queries. ChatGPT, in particular, generated relatively accurate information for various medical queries from genetics [10], biomedicine [17], radiation oncology physics [16]. Furthermore, several studies have evaluated the performance and feasibility of ChatGPT in the medical education field. In [31], GPT-3.5 [4] and GPT-4 [34] models were evaluated in terms of their understanding of surgical clinical information and their potential impact on surgical education and training. These findings demonstrate that LLMs can be successfully used in clinical education, still further efforts may be needed to overcome limitations. For more details concerning the evaluation of other applications like social science, natural science, engineering agent applications, education, search and recommendation, and personality testing, we refer the reader to [6].

Another interesting taxonomy from [6] groups the encountered testing approaches into three directions: (1) from the objective calculation (benchmarking) to human-in-the-loop testing, (2) from static to crowd-sourcing test sets and (3) from unified to challenging test sets. While unified settings involve test sets with no preference for any specific task, challenging settings create test sets for specific tasks. Tools like DeepTest[43] use seeds to generate input modifications for testing, CheckList [38] builds test sets based on templates, whereas AdaFilter [36] constructs tests adversarially. Furthermore, despite the growing number of academic projects designed for prompting LLMs [19, 27, 3], just a few of them support systematic evaluation of textual responses [48, 1]. ChainForge [1] is a visual toolkit that offers on-demand hypothesis testing of the behavior of text-generating LLMs on open-domain tasks.

To our knowledge, the use of combinatorial testing (CT), in particular for the testing of LLMs, is reported in a single paper [12]. Based on a given original sentence, the authors derive new sentences by replacing words with synonyms according to a combinatorial test set. Assuming that the semantics of the original sentence are preserved in the derived sentences, a test oracle is created based on existing annotations. In the experimental evaluation from [12], the authors apply generated pairwise sentence test sets from the BoolQ benchmark set [8] against two LLMs (T5 [37] and LLaMa [44]). The results indicate that the accuracy of the responses remains roughly equivalent to those provided for the original test set.

3 Combinatorial Testing

Combinatorial testing aims to generate test cases by considering a system's input model. The input model comprises a set of parameters (or variables) $\{x_1, \ldots, x_n\}$ and a not necessarily different domain d_i for each parameter x_i. The domain itself is a finite set of values a parameter can take. A test case is a n-tuple specifying a value $v_i \in d_i$ for every parameter $x_i \in \{x_1, \ldots, x_n\}$. A test suite is a set of test cases. Usually, we write a test suite as a table where the columns are the parameters, and the rows have their corresponding values.

Given an input model, a complete test suite comprises a row for each possible value-parameter combination. Obviously, the upper bound of rows is of order $O(D^n)$ where D is the maximum size of all domains d_i, $i = 1, \ldots, n$, i.e., $D = \max_{i=1,\ldots,n}(d_i)$. Hence, computing a complete test suite is not feasible for software or systems comprising a larger number of input parameters. Moreover, applying all test cases is not feasible because the system's behavior must also be evaluated. In combinatorial testing, we do not have a test oracle. The focus is only on input generation. Hence, such a test oracle must be added to classify a test case as passing or failing, i.e., indicating whether a test case leads to a correct or wrong output, respectively. It is worth noting that such a test oracle can be automated, and we will discuss this when showing our application for validating LLMs considering medical queries.

Combinatorial testing avoids computing all possible test cases. The idea behind this is to consider not all parameter combinations but only those combinations of values for a fixed number k (smaller than n) of parameters. Hence, a combinatorial test suite covers all combinations of values for any subset of parameters of size k, which is usually substantially smaller. Such a test suite is said to be of strength k or to be a k-wise test suite. If k is 2, then the test suite is a pairwise test suite, and we speak about pairwise testing. Note that in practice, pairwise testing is not good enough (see [22, 23]). For more information on combinatorial testing and its foundations, we refer the interested reader to [30, 21]. There are many algorithms available, including ACTS [47], for computing combinatorial test suites for arbitrary input models and strengths. It is also worth mentioning that combinatorial testing has been successfully used in many application domains, including autonomous driving [20] and security testing [40].

In the following, we illustrate combinatorial testing using a small example. In this example, we assume four parameters a, b, c, d, all of them only taking values from the Boolean domain $\{T, F\}$ standing for true and false. A pairwise combinatorial test suite for this input model comprises 6 test cases:

	a	b	c	d
1	T	T	F	F
2	T	F	T	T
3	F	T	T	F
4	F	F	F	T
5	F	T	F	T
6	T	F	F	F

For any combination of two parameters, e.g., a and c, this table comprises all possible combinations of values. Rows 1, 2, 3, and 4 already cover all four combinations for these two parameters. For parameters b and d, rows 1, 2, 5, and 6 are required to cover all value combinations. It can be easily checked that this holds also

for any other pair of parameters. Note that pairwise testing in this case only requires 6 test cases. Considering all combinations, we would have $2^4 = 16$ test cases. For the remainder of this paper, as we introduce domains extending beyond boolean values, we will use indices when referring to parameter values. For $\{T, F\}$, the indices would be 0 and 1, respectively, and the first row of the table above would be represented as $[0, 0, 1, 1]$.

4 Validation Methodology

Figure 1 gives a high-level overview of our proposed validation methodology. The remainder of this section follows the numbers shown in Figure 1 and discusses the individual elements of our validation pipeline.

The domain of our combinatorial prompt generation pipeline can be seen in Table 1, where parameters are components of a prompt and values are (sub-)phrases. Our prototypical set of parameters comprises a) symptom presentation, which is an introductory sub-phrase to the prompt, b) diagnostic focus, which sets the horizon for which kind of diagnoses are expected, c) an additional hint to consider context information such as age, and d) constraints on how the output should be formulated. Each parameter can assume an indexed value from the given set, and every set of values includes an empty string, which is denoted by "-".

Table 1: Domain: Prompt Components and Values by Index

Parameter (i.e. Prompt Component)	IDX	Value
Symptom Presentation *"Given the following..."*	0	-
	1	list of symptoms
	2	symptoms
	3	high-level overview of symptoms
Diagnostic Focus	0	-
	1	a probable diagnosis
	2	a differential diagnosis
	3	an emergency diagnosis
	4	the three most likely diagnoses
	5	the ten most likely diagnoses
Contextual Information	0	-
	1	based on patient's age and gender
Constraints *"the diagnosis should..."*	0	-
	1	be concise
	2	be detailed including explanations
	3	have less than 100 words

In our prototype implementation, we use pairwise testing for all parameters seen in Table 1. Note that we treat the sets of symptoms separately from the rest of the input parameters: Symptoms are excluded from the pairwise combination as we rather want to combine each set of symptoms with all pairwise combinations of the other parameters. This results in the listed 24 pairwise combinations per set of symptoms. Finally, the output of our combinatorial testing pipeline is textual prompts in natural language that act as test cases for evaluating an LLM. The resulting test suite from pairwise combinations for our exemplary set of symptoms can be seen in the first two columns of Table 3. For our preliminary evaluation, we used GPT-4o [34, 35] exclusively. The model can be accessed either via ChatGPT [33] or the OpenAI API [32]. Having both a chatbot interface and a programmable API enables rapid prompt prototyping as well as executing larger test suites with the same underlying model. That said, it would be easy to swap out the SUT and test another LLM using the same validation methodology.

We use the expert system "Symptom-Checker" [39], which is curated by medical professionals and freely accessible via NetDoktor [29]. Given a set of symptoms, we can automatically retrieve diagnoses by traversing a decision tree and answering yes-no-questions. In addition to "yes" and "no" there is also the option to skip a question when the inquired information is not available. Further questions are asking for age, sex, and main symptoms, as well as the body part that is influenced the most. We compute a score by comparing results (i.e., diagnoses) from our SUT ChatGPT with our Golden Model NetDoktor. This is done semi-automatically by first retrieving the diagnoses in the same format and then assessing their overlap. As NetDoktor always yields three diagnoses, our score ranges from 0/3 (no overlap) to 3/3 (complete overlap). In cases where the SUT yields more or less than three diagnoses, we do not normalize the score. This means that a result comprising only one diagnosis cannot achieve a complete overlap with NetDoktor and thus cannot achieve a better score than 1/3. In turn, a result comprising 10 diagnoses cannot achieve a higher score than 3/3. It must be noted that duplicate diagnoses are only counted once, and semantic equivalence is considered when comparing diagnoses.

5 Medical Use Case

For our use case, we assume the downstream task of retrieving diagnoses from an LLM based on a given set of symptoms. As mentioned earlier in this paper, hallucination is a severe problem when using LLMs. Especially in the medical domain, it is of utmost importance that systems are tested and validated in a structured way, as faulty output might have dire consequences ranging from misinformation to taking the wrong medication. However, LLMs cannot be tested exhaustively due to their non-deterministic nature and other factors, such as computational and monetary costs. Even testing a restricted domain, such as prompt formulation, given a limited set of input parameters, may lead to a combinatorial explosion when testing all possible combinations. For instance, compare the 24 pairwise combinations, as described above, to all possible 192 combinations from the values in Table 1 used with reasonably sized corpus of pathologies, such as DDXPlus [42]: This corpus comprises 134,530 samples (i.e. sets of symptoms) from the real world, which yields 25,829,760 possible test cases as compared to 3,228,720 for pairwise testing. Taking into account cost, we look at 206,134.40$ versus 25,766.80$. Table 2 gives a breakdown of the number of tokens for prompts as well as their cost, assuming the current pricing of GPT-4o, which is 5.00$ and 15.00$ per million input and output tokens, respectively. It has to be mentioned that the input length can be fully controlled, whereas the length of the output can only be guided by the prompt. In this case, we always assumed 475 output tokens, which was the average in our preliminary experiments. The average number of input tokens was 171.1, consisting of 19.1, which are derived from combinations of values seen in Table 1, and 152 coming from our exemplary set of symptoms, discussed in the following Subsection 5.1.

5.1 Example

To further explain our proposed validation methodology, we use the following exemplary description of symptoms:

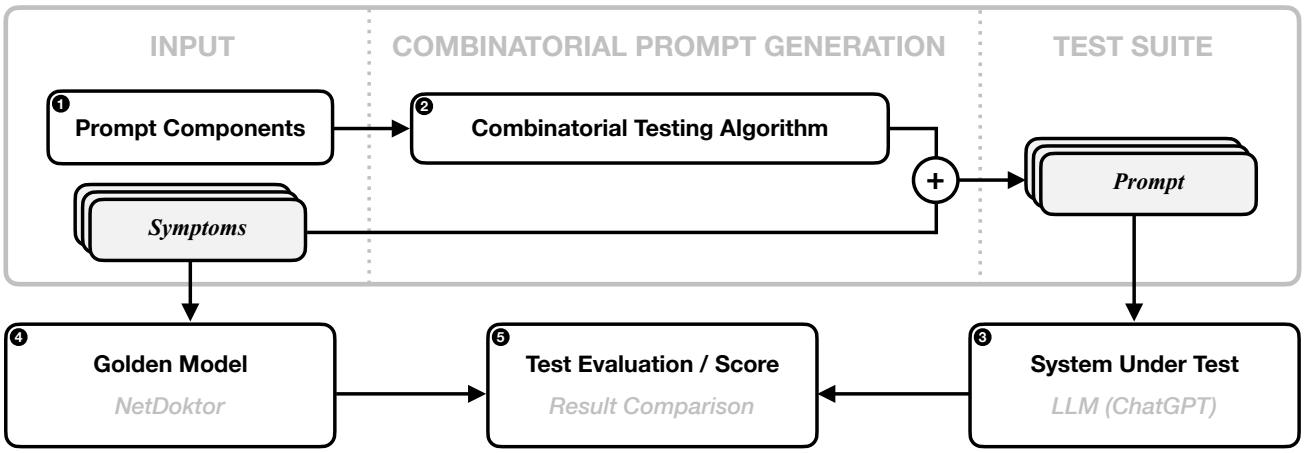

Figure 1: Basic Architecture of Our Validation Methodology.

Table 2: Cost per Size of Medical Corpus

Corpus Size	1	10	100	...	DDXPlus
All Combinations					
Combinations	192	1,920	19,200	...	25,829,760
Input Tokens	0.03M	0.33M	3.29M	...	4,419M
Input Cost [$]	0.16	1.64	16.43	...	22,097.36
Output Tokens	0.09M	0.91M	9.12M	...	12,269M
Output Cost [$]	1.37	13.68	136.80	...	184,037.04
Total Cost [$]	1.53	15.32	153.23	...	206,134.40
Pairwise Combinations					
Combinations	24	240	2,400	...	3,228,720
Input Tokens	0.004M	0.04M	0.41M	...	552M
Input Cost [$]	0.02	0.21	2.05	...	2,762.17
Output Tokens	0.01M	0.11M	1.14M	...	1,533M
Output Cost [$]	0.17	1.71	17.10	...	23,004.63
Total Cost [$]	0.19	1.92	19.15	...	25,766.80

> *An adult woman is experiencing symptoms in the breast gland area. Her most troubling symptom is fluid discharge, and she can feel a firm, painless lump.*

This set of symptom shall be seen as a sample from a corpus of medical pathologies (i.e. sets of symptoms). We use it to test different prompting strategies by combining the values from our domain, seen in Table 1. From this singular sample, we can generate 24 test cases as per our methodology. These test cases are then used to evaluate our SUT based on the output of our golden model. For this set of symptoms, our golden model diagnoses are:

- Breast cancer
- Cyst in the breast
- Mastopathy

Table 3 shows the test result of all 24 test cases. It can be easily spotted that test case 13 was the only prompt achieving a complete overlap with the NetDoktor diagnoses. When fully written-out, prompt 13 corresponding to the combination $[3, 5, 1, 0]$ was:

> *Given the following high-level overview of symptoms, provide the ten most likely diagnoses based on the patient's age and gender.*
> *An adult woman is experiencing symptoms in the breast gland area. Her most troubling symptom is fluid discharge, and she can feel a firm, painless lump.*

Other than the overlap score, there are severe differences in the output depending on the used prompt. Figure 2 illustrates those differences underlined by textual metrics, such as the number of words, while Figure 3 highlights the conciseness of the result as measured by the ratio between the number of words and the number of diagnoses. Most notably, the constraints to asking the LLM to provide concise diagnoses or limiting the number of words to 100 reduce the length drastically. As can be seen in Figure 2, the prompts 3, 4, 7, 8, 11, 12, 16, 17, 20, 22, 23, and 24 all yielded results with less than 200 words. However, prompt 7 exceeds the posed 100 word limit. Furthermore, none of these prompts fully overlapped. When comparing the results for conciseness in particular, Figure 3 shows that the ratio between the number of words and the number of diagnoses is less than 50 for all prompts querying the LLM to provide concise responses (i.e. 3, 4, 16, 17, 23, 24), whereas it is above 60 for all and above 100 for all but one of the prompts asking for a detailed response (i.e. 5, 6, 9, 10, 19).

In an effort to make our work as transparent and reproducible as possible, we provide all prompts and responses of our preliminary study as a replication package [1].

6 Conclusion

This paper highlights the importance of a structured and rigorous validation methodology for LLMs in the medical domain, particularly focusing on prompt engineering. The proposed validation pipeline makes use of pairwise combinatorial testing to systematically evaluate the responses of LLMs like ChatGPT to medical queries. The methodology generates test cases given sets of symptoms and combinations of prompt components. Combinatorial testing ensures that a wide range of prompt variations is tested per set of symptoms without causing a combinatorial

[1] https://zenodo.org/doi/10.5281/zenodo.13765131

Table 3: Overlaps of Diagnoses with Golden Model per Combination for One Exemplary Set of Symptoms. Each Combination Corresponds to One Prompt and Is Denoted by a Code Representing the Indices of the Assumed Values per Prompt Component, as Seen in Table 1. "Mast." stands for " "Mastopathy".

Test Suite		Golden Model Overlap			Score
ID	Combination	Cancer	Cyst	Mast.	
1	[0, 0, 0, 0]	✓	✓		2/3
2	[1, 1, 1, 0]	✓			1/3
3	[2, 2, 1, 1]	✓			2/3
4	[3, 3, 0, 1]	✓			1/3
5	[3, 4, 1, 2]	✓			1/3
6	[2, 5, 0, 2]	✓	✓		2/3
7	[1, 5, 0, 3]	✓	✓		2/3
8	[0, 4, 1, 3]	✓			1/3
9	[0, 3, 1, 2]	✓	✓		2/3
10	[1, 2, 0, 2]	✓	✓		2/3
11	[2, 0, 1, 3]	✓	✓		2/3
12	[3, 1, 0, 3]	✓	✓		2/3
13	**[3, 5, 1, 0]**	✓	✓	✓	**3/3**
14	[2, 4, 0, 0]		✓		1/3
15	[1, 3, 0, 0]	✓			1/3
16	[0, 1, 0, 1]	✓	✓		2/3
17	[1, 0, 0, 1]	✓	✓		2/3
18	[0, 2, 0, 0]	✓	✓		2/3
19	[2, 1, 0, 2]	✓			1/3
20	[3, 2, 0, 3]	✓			2/3
21	[3, 0, 0, 2]	✓	✓		2/3
22	[2, 3, 0, 3]	✓			2/3
23	[0, 5, 0, 1]	✓	✓		2/3
24	[1, 4, 0, 1]	✓			1/3

explosion. Doing so is more efficient and reduces costs as compared to testing all possible combinations of prompt parameters, which is especially important when evaluating LLMs on large medical corpora. The proposed validation pipeline implements semi-automated scoring based on a "golden model", which provides diagnoses curated by medical professionals. In a preliminary study, we demonstrate severe differences in output for prompt variations given the same set of symptoms. Out of 24 test cases, only one achieved a full overlap with our golden model when using GPT-4o. Once more, this highlights the dependence on well-formulated prompts and a need for thorough testing strategies, especially in critical domains like medicine.

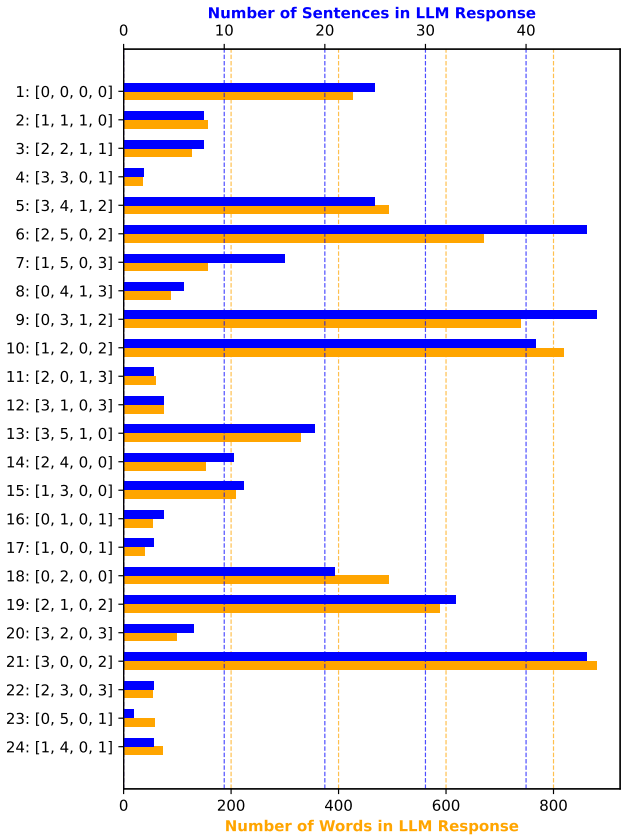

Figure 2: Textual Metrics of LLM Responses: Number of Sentences & Number of Words per Prompt

Acknowledgements

The work presented in this paper was partially funded by the European Union under Grant 101159214 – ChatMED. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] Ian Arawjo, Chelse Swoopes, Priyan Vaithilingam, Martin Wattenberg, and Elena L. Glassman. 2024. ChainForge: A Visual Toolkit for Prompt Engineering and LLM Hypothesis Testing. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (CHI '24). ACM, (May 2024), 1–18. DOI: 10.1145/3613904.3642016.

[2] Yejin Bang et al. 2023. A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity. (2023). https://arxiv.org/abs/2302.04023 arXiv: 2302.04023 [cs.CL].

[3] Stephen Brade, Bryan Wang, Mauricio Sousa, Sageev Oore, and Tovi Grossman. 2023. Promptify: text-to-image generation through interactive prompt exploration with large language models. (2023). https://arxiv.org/abs/2304.09337 arXiv: 2304.09337 [cs.HC].

[4] Tom B. Brown et al. 2020. Language models are few-shot learners. arXiv: 2005.14165 [cs.CL]. (2020).

[5] David Cecchini, Arshaan Nazir, Kalyan Chakravarthy, and Veysel Kocaman. 2024. Holistic evaluation of large language models: assessing robustness, accuracy, and toxicity for real-world applications. In *Proceedings of the 4th Workshop on Trustworthy Natural Language Processing (TrustNLP 2024)*. Anaelia Ovalle, Kai-Wei Chang, Yang Trista Cao, Ninareh Mehrabi, Jieyu Zhao, Aram Galstyan, Jwala Dhamala, Anoop Kumar, and Rahul Gupta, editors. Association for Computational Linguistics, Mexico City, Mexico, (June 2024), 109–117. DOI: 10.18653/v1/2024.trustnlp-1.11.

[6] Yupeng Chang et al. 2023. A survey on evaluation of large language models. (2023). https://arxiv.org/abs/2307.03109 arXiv: 2307.03109 [cs.CL].

[7] Wei-Lin Chiang et al. 2024. Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference. (2024). arXiv: 2403.04132 [cs.AI].

[8] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: exploring the surprising difficulty of natural yes/no questions. (2019). https://arxiv.org/abs/1905.10044 arXiv: 1905.10044 [cs.CL].

[9] J. Clusmann et al. 2023. The future landscape of large language models in medicine. *Communications Medicine*, 3, 141. DOI: https://doi.org/10.1038/s43856-023-00370-1.

[10] Dat Duong and Benjamin D. Solomon. 2023. Analysis of large-language model versus human performance for genetics questions. *medRxiv*. eprint: https://www.medrxiv.org/content/early/2023/01/28/2023.01.27.23285115.full.pdf. DOI: 10.1101/2023.01.27.23285115.

[11] Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630. https://doi.org/10.1038/s41586-024-07421-0.

[12] 2023. *Applying pairwise combinatorial testing to large language model testing*. Springer, (Sept. 2023), 247–256. ISBN: 978-3-031-43239-2. DOI: 10.1007/978-3-031-43240-8_16.

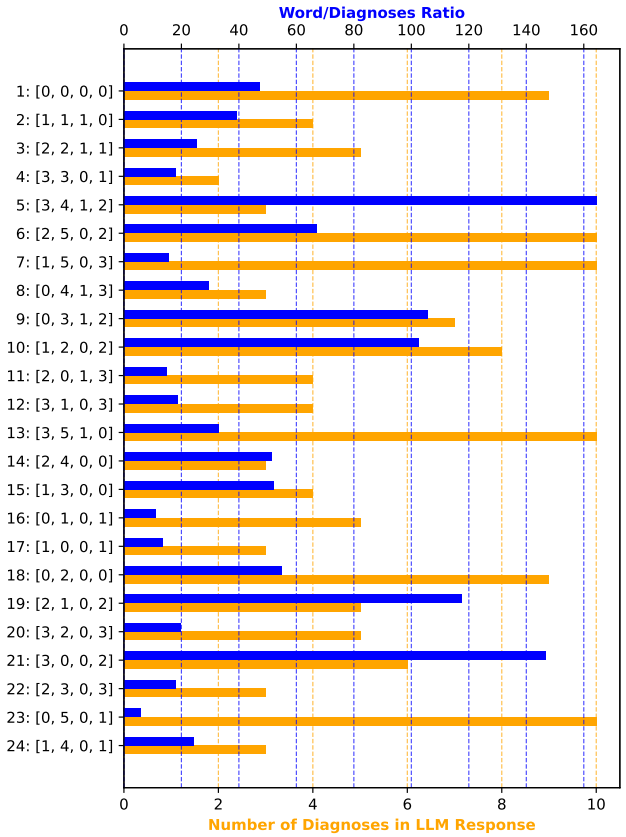

Figure 3: Conciseness of LLM Responses: Number of Diagnoses & Ratio between Number of Words and Number of Diagnoses per Prompt

[13] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. (2021). https://arxiv.org/abs/2009.03300 arXiv: 2009.03300 [cs.CY].

[14] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. (2021). https://arxiv.org/abs/2103.03874 arXiv: 2103.03874 [cs.LG].

[15] Dan Hendrycks et al. 2021. Measuring coding challenge competence with apps. (2021). https://arxiv.org/abs/2105.09938 arXiv: 2105.09938 [cs.SE].

[16] Jason Holmes et al. 2023. Evaluating large language models on a highly-specialized topic, radiation oncology physics. *Frontiers in Oncology*, 13, (July 2023). DOI: 10.3389/fonc.2023.1219326.

[17] Israt Jahan, Md Tahmid Rahman Laskar, Chun Peng, and Jimmy Huang. 2023. Evaluation of chatgpt on biomedical tasks: a zero-shot comparison with fine-tuned generative transformers. (2023). https://arxiv.org/abs/2306.04504 arXiv: 2306.04504 [cs.CL].

[18] Neel Jain, Khalid Saifullah, Yuxin Wen, John Kirchenbauer, Manli Shu, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2023. Bring your own data! self-supervised evaluation for large language models. (2023). https://arxiv.org/abs/2306.13651 arXiv: 2306.13651 [cs.CL].

[19] Ellen Jiang, Kristen Olson, Edwin Toh, Alejandra Molina, Aaron Donsbach, Michael Terry, and Carrie J Cai. 2022. Promptmaker: prompt-based prototyping with large language models. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (CHI EA '22) Article 35. Association for Computing Machinery, New Orleans, LA, USA, 8 pages. ISBN: 9781450391566. DOI: 10.1145/3491101.3503564.

[20] Florian Klück, Yihao Li, Jianbo Tao, and Franz Wotawa. 2023. An empirical comparison of combinatorial testing and search-based testing in the context of automated and autonomous driving systems. *Information and Software Technology*, 160, 107225. DOI: https://doi.org/10.1016/j.infsof.2023.107225.

[21] D.R. Kuhn, R.N. Kacker, and Y. Lei. 2013. *Introduction to Combinatorial Testing. Chapman & Hall/CRC Innovations in Software Engineering and Software Development Series*. Taylor & Francis.

[22] D.R. Kuhn, R.N. Kacker, Y. Lei, and J. Hunter. 2009. Combinatorial software testing. *Computer*, (Aug. 2009), 94–96.

[23] Rick Kuhn, Yu Lei, and Raghu Kacker. 2008. Practical combinatorial testing: beyond pairwise. *IT Professional*, 10, 3, 19–23.

[24] Boniphace Kutela, Kelvin Msechu, Norris Novat, Emmanuel Kidando, and Angela Kitali. 2023. Uncovering the influence of chatgpt's prompts on scientific writings using machine learning-based text mining approaches. *SSRN Electronic Journal*. http://dx.doi.org/10.2139/ssrn.4385895.

[25] Percy Liang et al. 2023. Holistic evaluation of language models. (2023). https://arxiv.org/abs/2211.09110 arXiv: 2211.09110 [cs.CL].

[26] Yen-Ting Lin and Yun-Nung Chen. 2023. Llm-eval: unified multi-dimensional automatic evaluation for open-domain conversations with large language models. (2023). https://arxiv.org/abs/2305.13711 arXiv: 2305.13711 [cs.CL].

[27] Aditi Mishra, Utkarsh Soni, Anjana Arunkumar, Jinbin Huang, Bum Chul Kwon, and Chris Bryan. 2023. Promptaid: prompt exploration, perturbation, testing and iteration using visual analytics for large language models. (2023). https://arxiv.org/abs/2304.01964 arXiv: 2304.01964 [cs.HC].

[28] M. Nazari and G. Saadi. 2024. Developing effective prompts to improve communication with chatgpt: a formula for higher education stakeholders. *Discov Educ*, 3. https://doi.org/10.1007/s44217-024-00122-w.

[29] 2024. Netdoktor. https://www.netdoktor.at. Accessed: 2024-09-03. (2024).

[30] Changhai Nie and Hareton Leung. 2011. A survey of combinatorial testing. *ACM Comput. Surv.*, 43, 2, Article 11, (Feb. 2011), 11:1–11:29.

[31] Namkee Oh, Gyu-Seong Choi, and Woo Yong Lee. 2023. Chatgpt goes to the operating room: evaluating gpt-4 performance and its potential in surgical education and training in the era of large language models. *Annals of Surgical Treatment and Research*, 104, (Apr. 2023), 269. DOI: 10.4174/astr.2023.104.5.269.

[32] OpenAI. 2023. API Reference. (2023). platform.openai.com/docs/api-reference.

[33] OpenAI. 2023. ChatGPT. (2023). chat.openai.com/chat.

[34] OpenAI. 2023. GPT-4 technical report. arXiv: 2303.08774 [cs.CL]. (2023).

[35] OpenAI. 2024. Introducing gpt-4o and more tools to chatgpt free users. (May 2024). https://openai.com/index/gpt-4o-and-more-tools-to-chatgpt-free/.

[36] Jason Phang, Angelica Chen, William Huang, and Samuel R. Bowman. 2021. Adversarially constructed evaluation sets are more challenging, but may not be fair. (2021). https://arxiv.org/abs/2111.08181 arXiv: 2111.08181 [cs.CL].

[37] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21, 140, 1–67. http://jmlr.org/papers/v21/20-074.html.

[38] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: behavioral testing of nlp models with checklist. (2020). https://arxiv.org/abs/2005.04118 arXiv: 2005.04118 [cs.CL].

[39] Jens Richter, Hans-Richard Demel, Florian Tiefenböck, Luise Heine, and Martina Feichter. 2024. Symptom-checker. https://www.netdoktor.at/symptom-checker/. Accessed: 2024-09-03. (2024).

[40] Dimitris E. Simos, Josip Bozic, Bernhard Garn, Manuel Leithner, Feng Duan, Kristoffer Kleine, Yu Lei, and Franz Wotawa. 2019. Testing TLS using planning-based combinatorial methods and execution framework. *Software Quality Journal*, 27. https://doi.org/10.1007/s11219-018-9412-z.

[41] Karan Singhal et al. 2022. Large language models encode clinical knowledge. (2022). https://arxiv.org/abs/2212.13138 arXiv: 2212.13138 [cs.CL].

[42] Arsene Fansi Tchango, Rishab Goel, Zhi Wen, Julien Martel, and Joumana Ghosn. 2022. DDXPlus: A New Dataset For Automatic Medical Diagnosis. (2022). https://arxiv.org/abs/2205.09148 arXiv: 2205.09148 [cs.CL].

[43] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars. (2018). https://arxiv.org/abs/1708.08559 arXiv: 1708.08559 [cs.SE].

[44] Hugo Touvron et al. 2023. LLaMA: Open and Efficient Foundation Language Models. (2023). https://arxiv.org/abs/2302.13971 arXiv: 2302.13971 [cs.CL].

[45] Yidong Wang et al. 2024. PandaLM: An Automatic Evaluation Benchmark for LLM Instruction Tuning Optimization. (2024). https://arxiv.org/abs/2306.05087 arXiv: 2306.05087 [cs.CL].

[46] Yijun Xiao and William Yang Wang. [n. d.] On hallucination and predictive uncertainty in conditional language generation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. DOI: 10.18653/v1/2021.eacl-main.236.

[47] Linbin Yu, Yu Lei, R.N. Kacker, and D.R. Kuhn. 2013. ACTS: A Combinatorial Test Generation Tool. In *Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on*, 370–375.

[48] J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. 2023. Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (CHI '23) Article 437. Association for Computing Machinery, Hamburg, Germany, 21 pages. ISBN: 9781450394215. DOI: 10.1145/3544548.3581388.

[49] Lianmin Zheng et al. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. https://openreview.net/forum?id=uccHPGDlao.

[50] Caleb Ziems, William Held, Omar Shaikh, Jiaao Chen, Zhehao Zhang, and Diyi Yang. 2024. Can large language models transform computational social science? (2024). https://arxiv.org/abs/2305.03514 arXiv: 2305.03514 [cs.CL].