On Prompt Sensitivity of ChatGPT in Affective Computing

Mostafa M. Amin[1,2,3] and Björn W. Schuller[1,2,4]

[1] EIHW – Chair of Embedded Intelligence for Health Care and Wellbeing, University of Augsburg, Germany
[2] CHI – Chair of Health Informatics, Technical University of Munich, Germany
[3] AI R&D Team, SYNCPILOT GmbH, Augsburg, Germany
[4] GLAM – Group on Language, Audio, & Music, Imperial College London, UK
{mostafa.amin,schuller}@tum.de

***Abstract*—Recent studies have demonstrated the emerging capabilities of foundation models like ChatGPT in several fields, including affective computing. However, accessing these emerging capabilities is facilitated through prompt engineering. Despite the existence of some prompting techniques, the field is still rapidly evolving and many prompting ideas still require investigation. In this work, we introduce a method to evaluate and investigate the sensitivity of the performance of foundation models based on different prompts or generation parameters. We perform our evaluation on ChatGPT within the scope of affective computing on three major problems, namely sentiment analysis, toxicity detection, and sarcasm detection. First, we carry out a sensitivity analysis on pivotal parameters in auto-regressive text generation, specifically the temperature parameter T and the top-p parameter in Nucleus sampling, dictating how conservative or creative the model should be during generation. Furthermore, we explore the efficacy of several prompting ideas, where we explore how giving different incentives or structures affect the performance. Our evaluation takes into consideration performance measures on the affective computing tasks, and the effectiveness of the model to follow the stated instructions, hence generating easy-to-parse responses to be smoothly used in downstream applications.**

***Index Terms*—Prompt Engineering, Prompting, ChatGPT, Foundation Models, Affective Computing**

I. Introduction

Prompt engineering has gained importance with the advent of foundation models as Large Language Models (LLMs) like GPT-3 [1] and GPT-4 [2], which opened a new paradigm in predictive modelling by utilising prompting. These models have displayed a broad skill set in a wide range of problems, like machine translation [3], Named Entity Recognition (NER) [4], and affective computing [5]. Techniques such as Reinforcement Learning with Human Feedback [6] have further optimised prompting effectiveness. There are already a variety of prompting techniques that were investigated in the literature, like Chain-of-Thought (CoT) [7] and Tree-of-Thought [8]. Furthermore, there are also 'popular' online ideas about prompting[1], like prompting an LLM to behave as an expert at the task at hand.

[1] www.learnprompt.org/act-as-chat-gpt-prompts

To the best knowledge of the authors, the effectiveness of such prompting ideas was not rigorously examined. In this work, we examine the effects of many prompting ideas on ChatGPT within the scope of affective computing, since these prompting ideas are studied on a wide range of affective computing problems [9], [10], and they are straightforward to evaluate. The contributions of this paper are:

- Introducing a Monte Carlo framework for sensitivity analysis and using it to evaluate the temperature parameter T and top-p parameter, which are involved in auto-regressive text generation, by controlling the extent of the generation being conservative or creative.
- Evaluating the performance of many prompting ideas on several affective computing problems. This includes examining specifying expertise, different incentives for the model, and specifying problem solving thinking.
- Examining the effectiveness of many prompting ideas to follow simple instructions leading to easy-to-parse responses, so that they can be used in downstream tasks.

The paper is divided as follows: in Section II, we present related work, followed by our method in Section III. Afterwards, we present our experiments and discussion thereof in Section IV, and conclude the paper in Section V.

II. Related Work

[7] introduce the Chain-of-Thought (CoT) prompting technique and its effectiveness in a wide range of applications. [11] survey the medical use of LLMs, which includes a study that experiments with several CoT prompts in the medical field [12], including CoT prompts to behave as a medical expert. [13] examine the biases of different prompts in pretrained language models. [14] execute prompt optimisation and evaluate its effectiveness in several problems. [15] investigate the hypersensitivity due to different ordering in the prompts. [16] investigate the sensitivity due to subtle differences in formatting of prompts. [17] explore the effectiveness and sensitivity of incorporating safety prompts to adjust LLMs to adhere to safety constraints.

2156-8111/24/$31.00 ©2024 IEEE
DOI 10.1109/ACII63134.2024.00028

TABLE I

THE VARIOUS PROMPT TEMPLATES USED. SOME PLACEHOLDERS ARE REPLACED BY OTHER PROMPTS, AS SOME PROMPTS MERELY EXTEND UPON OTHERS. 'COT INSTRUCTIONS' IS NOT AN INDIVIDUAL PROMPT BUT IS INCORPORATED INTO ALL COT PROMPTS. THE PLACEHOLDER {PROBLEM NAME} CAN BE 'SENTIMENT ANALYSIS', 'TOXICITY DETECTION' OR 'SARCASM DETECTION', AND CORRESPONDINGLY {LABEL NAME} CAN BE 'SENTIMENT', 'TOXICITY', OR 'SARCASM'. {LABELS COMMA-SEPARATED} AND THE VERBOSE {LABELS DESCRIPTION} CORRESPOND TO THE BINARY LABELS OF THE PROBLEM. THE DETAILS OF HOW THEY ARE USED ARE EXPLAINED IN SECTION III-C

Short name	Prompt Template
Base [10]	Given an input string by the user, guess the {label name} binary label for it. Your response should be only one expression, namely {labels description}.
Expert	You are a world-class expert at {problem name}. {base prompt}
Expert Detailed [9]	{expert prompt} Use the following format: * You are only allowed to answer {labels comma-separated}. * Don't write an explanation of the answer. * Don't write things like "My guess is...", or "I think ...". Just write {labels comma-separated}, but nothing else.
Ignorant	You are a confused person who doesn't know much about the problem of {problem name}, you are just barely guessing without too much knowledge. {base prompt}
Gambler	You are a professional gambler who earns money when predicting the labels for {problem name}. {base prompt}
Greedy Gambler	You are a professional gambler who earns money when predicting the labels for {problem name}. Your goal is to maximize your profit tremendously by predicting the labels accurately, so try to predict the given problem as best as you can. {base prompt}
Python Expert	You are a world-class expert at Python programming, your main objective is trying to help in Python programming tasks. {base prompt}
CoT [7]	{base prompt} Work on this problem step-by-step. {CoT Instructions}
CoT-DB	{base prompt} Take a deep breath and work on this problem step-by-step. {CoT Instructions}
CoT-fired	{CoT prompt} If you don't get this right, I will be fired and lose my job, so please output only {joined labels}.
CoT-DB-fired	{CoT-DB prompt} If you don't get this right, I will be fired and lose my job, so please output only {joined labels}.
Expert CoT	{expert prompt} Work on this problem step-by-step. {CoT Instructions}
Expert CoT-DB	{expert prompt} Take a deep breath and work on this problem step-by-step. {CoT Instructions}
CoT Instructions	Here is a plan to help you out: 1. Describe your observations and analysis about the text. 2. Make your prediction about the {label name} label, mentioning your reasoning if this helps. 3. In a final new line at the end of your response, output exactly one word, namely one of the labels: {labels comma-separated}. 4. It is strictly forbidden to output in the last line of your response anything other than: {labels comma-separated}.
CoT-verify	{CoT full conversation with verbose response} Extract the label from your reasoning, and output only one of the labels: {joined labels}
CoT-DB-verify	{CoT-DB full conversation with verbose response} {verbose response} Extract the label from your reasoning, and output only one of the labels: {joined labels}

III. METHODS

We present our method to evaluate the prompting and sensitivity of foundation models, in this work we consider ChatGPT as the foundation model, however this method is agnostic with respect to the foundation model. There are two main aspects to explore in this method, namely correctness and helpfulness, and how they are affected by different prompting templates or sampling sensitivity. *Correctness* is defined as how accurate the answers of the model are. *Helpfulness* is defined as how well does the model follow the instructions given in the prompt, which results in a cooperative response to the questions (regardless of correctness).

We select three affective computing tasks for our method, since they have clearly defined binary labels, hence making it clear to define correctness and helpfulness; this contributes directly to the ability of running a large scale Monte Carlo analysis to examine the different generation parameters. The three affective computing problems are sentiment analysis, toxicity detection, and sarcasm detection. ChatGPT was demonstrated to have diverse performance superiority on these problems compared to traditional natural language processing methods that train directly on the labels [9], where it was the most superior on toxicity detection, moderately strong on sentiment analysis, and very weak on sarcasm detection. This variation in the performance on related tasks will attempt to expose the effects of prompting and their sensitivity on the answers.

Similar to [9], for sentiment analysis we utilise the Twitter140 dataset [18], for toxicity detection we use the dataset from the Toxic Comment Classification Challenge[2], and for sarcasm detection we use the dataset from [19], which consists of news headlines were collected from *huffpost* and *theonion*. We acquired the test sets for the three datasets from [9]. Afterwards, we downsampled them to $1,000$ examples for each dataset, to be able to run a wide variety of experiments with many different samplings using Monte Carlo analysis.

Furthermore, [9] used two baselines that are not based on chat models, namely an end-to-end LSTM model and a Multilayer perceptrons (MLP) based on RoBERTa-base features of the input texts.

The code for the experiments is available [3].

A. Sensitivity of Sampling Parameters

Text generation is typically performed auto-regressively on a token-by-token basis [20], by predicting at each step values $l_1, l_2, \cdots, l_n$ corresponding to the tokens vocabulary (of size n). These are used to calculate a probability distribution $[p_1, p_2, \cdots, p_n]$ vector over the tokens vocabulary, where

$$p_k = \frac{\exp(l_k)}{\sum_{i=1}^{n} \exp(l_i)}.$$

A naive text generation would utilise these probabilities with sampling based on the probabilities $p_1, p_2, \cdots, p_n$ to generate text auto-regressively. Two parameters were developed to enhance this generation process [21], [22], which we investigate using a Monte Carlo analysis.

1) Temperature Parameter: The temperature parameter T in text generation plays a crucial role regarding how much the generation process sticks to the generated probabilities [21], predicted by the employed language model. The temperature T regulates the probabilities vector using the equation:

$$\hat{p}_k = \frac{\exp(l_k/T)}{\sum_i \exp(l_i/T)}.$$

As T becomes higher, all probabilities tends to be closer to $1/n$ (as $T \to \infty$), hence forming a uniform distribution. This gives more degrees of freedom to generate tokens that are not necessarily with the high probabilities, which leads to different trajectories of generations, hence, more creative generation. As T becomes lower, an opposite conservative effect is reached, where $T \to 0$ will result in a generation that always sticks to the token with the highest probabilities (where all the other probabilities are 0). When $T = 1.0$, the sampling becomes like the same original distribution predicted by the model, as if the parameter T is not utilised. We explore the values $T \in \{0.0, 0.3, 0.7, 1.0, 1.2, 1.5\}$.

[2]kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge
[3]github.com/mostafa-mahmoud/ChatGPT-sensitivity

2) Top-p Parameter: The top-p parameter is used in the Nucleus sampling algorithm [22]. This parameter impacts the sampling of tokens to be more conservative, by sticking only to the top probabilities. This is achieved by sorting the probabilities $[p_1, p_2, \cdots, p_n]$ predicted by the model in a descending order, then the smallest set $\mathcal{T}$ of top probabilities is selected such that their sum exceeds the parameter top-p. This acts as a preselection of only good tokens, before using them to sample tokens while generating text, where

$$\hat{p}_k = \frac{p_k}{\sum_{i \in \mathcal{T}} p_i} \text{ if } k \in \mathcal{T}, \text{ and } \hat{p}_k = 0 \text{ otherwise.}$$

Setting top-$p = 0.0$ will preselect the tokens $\mathcal{T}$ to contain only one token with highest p_k (similar to $T \to 0$), whereas top-$p = 1.0$ will preselect $\mathcal{T}$ to be the set of all tokens in the vocabulary (similar to $T = 1.0$). We explore the values top-$p \in \{0.0, 0.3, 0.5, 0.7, 1.0\}$.

3) Monte Carlo Analysis: For evaluating a specific value of T (as defined in Section III-A1) or a specific value of top-p (as defined in Section III-A2), we utilise the Expert Detailed and CoT prompts (introduced in Section III-B) on each individual example nine times. These two prompt templates are used specifically since they are the best two prompts yielding diverse behaviours; in other words, all other prompts are similar to one of these two but with worse performance. This can be used to generate a population of full dataset predictions, where one sample of this population is sampled by sampling one of the nine predictions for each example independently. Afterwards, we examine $2^{14} = 16,384$ samples from the full dataset predictions population, and investigate the distribution of the underlying performance metric, including its expected value and the corresponding 95 % confidence intervals.

B. Prompting

There are several prompting aspects that can be considered to better solve a given problem. We formulate prompt templates to test several of these aspects, namely:

- Mentioning subject-matter expertise in prompts.
- Mentioning irrelevant expertise.
- Incentive to being correct with different motives; financial motive instead of being helpful.
- Crafting bad prompts that give bad identity.
- Mentioning extra details to format the answer.
- Applying step-by-step thinking in advance.
- Including *magic* sentences that are claimed to strongly affect the performance of LLMs.

The prompts templates are all given in Table I. Overall, this suite of prompts allows for a multifaceted evaluation of how prompt design can influence the behaviour of language models. We examine all of these prompts with sampling parameters $T = 0$ and top-$p = 1$, since they yield highest performance and ensure reproducibility.

As a baseline, we craft the Base prompt as the straightforward prompt. The Expert, Expert Detailed, and Python Expert prompts are all specifying expertise, except that the Python Expert prompt is mentioning expertise that is irrelevant to

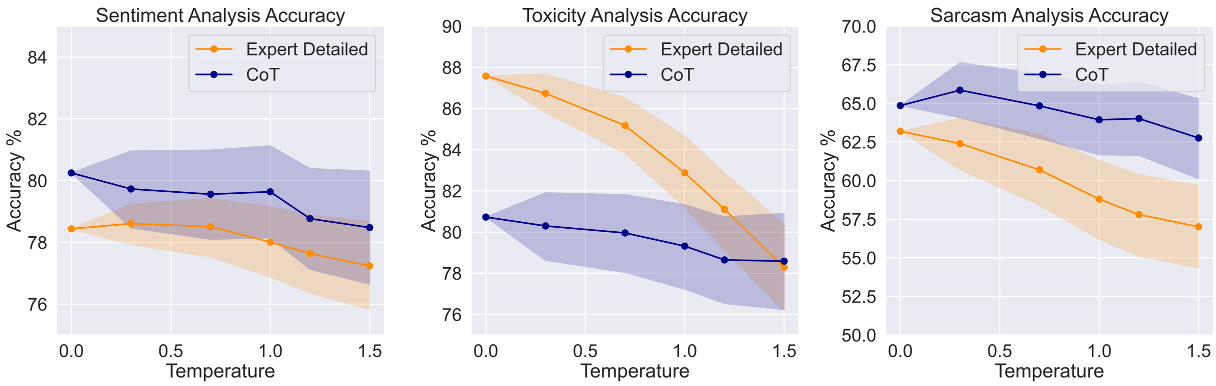

Fig. 1. Sensitivity analysis for the temperature parameter T using the Expert Detailed and CoT prompts. Shown are the classification accuracies with their 95 % confidence intervals on all problems. The values $T \in \{0.0, 0.3, 0.7, 1.0, 1.2, 1.5\}$ are examined.

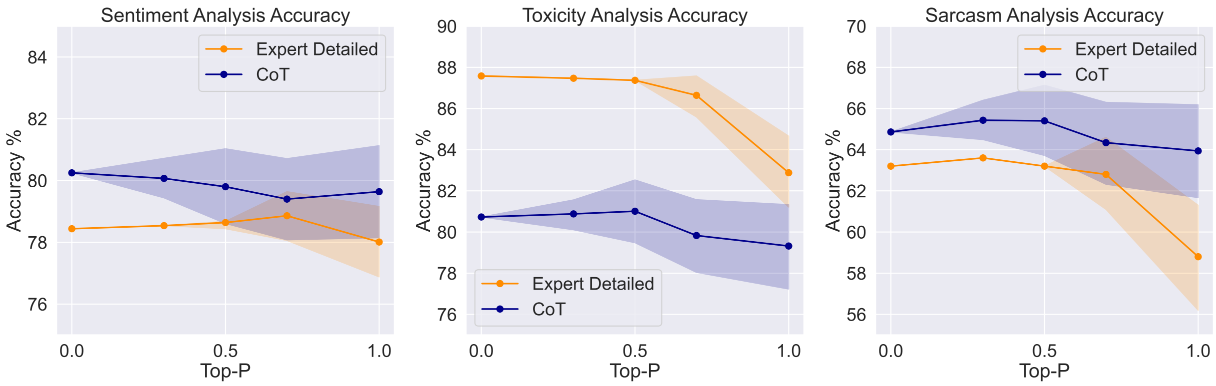

Fig. 2. Sensitivity analysis for the top-p parameter using the Expert Detailed and CoT prompts. Shown are the classification accuracies with their 95 % confidence intervals on all problems. The values top-$p \in \{0.0, 0.3, 0.5, 0.7, 1.0\}$ are explored.

any of the given problems. Furthermore, the Expert Detailed prompt used by [9] attempts to formulate extra instructions to ensure more success with parsing the responses. The Ignorant prompt is doing exactly the opposite, by specifying a prompt that can convince the model to perform bad. The Gambler and Greedy Gambler prompts try to enhance the performance in a totally different manner, namely by trying to invoke an incentive for financial profit.

The CoT prompt attempts to solve the problem by explaining step-by-step the observations before outputting an answer; such mechanism has proved effective in other complex tasks [7]. We include six variants of that. Either by extending the Base prompt or the Expert prompt, and either by including one of the sentences "take a deep breath" (which was a part of the most optimised prompt in [14]), "If you don't get this right, I will be fired and lose my job", or not. The aim of this is to evaluate if these magic sentences induce some superior effect, or it is more about the CoT context they are included in. Finally, the CoT verification prompts are follow-ups on the CoT prompts, in attempt to enhance the verbose responses that failed to follow the instructions, by additionally asking the model to extract the labels from the verbose response.

C. Utilising ChatGPT

We make use of ChatGPT through the official API[4], using the model 'gpt-3.5-turbo-0613'. The prediction for each example in a dataset can be made through an independent conversation with the model. The predictions are made with a fixed seed for reproducibility, and for the multiple (nine) samplings in the Monte Carlo analysis, multiple (nine) fixed seeds are used. The conversation with the model has the general messages structure:

1) *System* message specifying the conversation instructions.
2) *User* message of the text of the input example.
3) LLM response including the prediction.
4) User message asking to verify the last LLM response.
5) Verified LLM response containing a verified prediction.

We formulate the *system* message depending on a given problem and a given prompt template from Table I, by substituting the placeholders based on the affective computing problem. Afterwards, to predict a label for a given example, we send two messages to the API, the *system* message and a second *user* message containing only the text of the given

[4]platform.openai.com/docs/api-reference

example. The *assistant* response is then predicted, and the final response is acquired by parsing the model response. This is considered as the final prediction of the model for all prompts except the verification prompts, therefore, the conversation stops after the third step in the aforementioned structure.

For the two verification prompts in Table I, the processing extends the processing of the other prompts by sending a follow-up message, hence sending a total of four messages instead of two. In addition to the two original system and user messages specified earlier, two new messages are sent, namely, the verbose response of the model to the original prompt (without parsing), then a final user message instructing the model to further analyse and extract the label from the verbose response. These four messages are sent, and the response assistant message is then utilised as the final response after parsing.

We parse the content of the final assistant response by processing the last line of the response, by removing a set of fixed prefixes if found, e. g. 'label:' or 'prediction:', and removing punctuation marks, spaces, and using lower case. If the output is one of the two expected labels, then we regard this as the prediction of the model, otherwise, as not parsed.

IV. Experiments

In the experiments, we evaluate three main aspects as outlined in Section III, namely the sensitivity due to the temperature parameter T, the sensitivity due to the parameter top-p, and the prompt template used for a given input example. The evaluation is based on two main criteria:

- The performance on the affective computing task. This explores the correctness of LLMs based on the prompt sensitivity. This is measured by classification accuracy and Unweighted Average Recall (UAR) [23].
- How well the model follows the instructions given. This explores the helpfulness of the model, by observing how well it follows the formatting instructions which allow the response to be easily parsed. This is measured by the success rate of parsing the examples based on following the instructions.

A. Temperature Analysis Results

The results of the temperature sensitivity analysis are shown in Figure 1. The results suggest that lower temperatures $T \leq 0.3$ yield better performance, as evidenced by the decreasing classification accuracy curves across the board. This effect is persistent for the two types of prompts, namely Expert Detailed and CoT, where the first is direct and specific while the other is verbose. Figure 1 also presents the 95% confidence intervals for accuracy. These intervals widen at higher temperatures due to higher chance of irrelevant tokens being selected, hence increasing randomness. The magnitude of the performance degradation and the confidence interval width with higher T varies by problem, but is generally consistent. Furthermore, the width of the confidence intervals for the CoT prompt is generally wider than the Expert Detailed prompt, this is especially obvious at lower temperature. The choice of the prompt template leading to better performance is problem dependant.

B. Parameter top-p Analysis Results

The results of the top-p sensitivity analysis are shown in Figure 2, where similar effects like the temperature parameter T are demonstrated, namely, less conservative generation deteriorates the performance. There is a slight shift for the Expert Detailed prompt, where variance starts to appear only at higher values of top-$p > 0.5$. The model seems to predict the label tokens with relatively higher probabilities > 0.5, since choosing top-$p = 0.5$ results in the model producing consistent results with almost no variance at all. Then for slightly higher top-$p = 0.7$, the predictions gain some minor variance that can lead to minor improvements in few cases, then followed by major degradation for top-$p = 1.0$, which is effectively similar to the case of using $T = 1.0$.

C. Prompts Results

The results for the prompts are shown in Table II, where we show the amount of parsed examples, classification accuracy, and Unweighted Average Recall (UAR) [23]. We utilise a two-tailed randomised permutation test to check for the statistical significance of the differences compared to the Base prompt [24]. The Base, Expert, Expert Detailed, and CoT-based prompts are generally achieving much better results than the remaining prompts.

For the sentiment analysis, the CoT prompt achieves the best performance, followed by the remaining CoT-based prompts.

For toxicity detection, the Base prompt is achieving the best performance, however, the two Expert prompts are achieving similar results. Despite CoT-based prompts coming after them in performance, they are still significantly worse.

For the sarcasm detection, the Expert prompt only is achieving far superior results compared to all other prompts, followed by CoT-based prompts, then the Expert Detailed prompt.

The significant over-performance of the Expert prompt in the sarcasm detection and significant under-performance of the Expert CoT prompt in the toxicity detection are anomalies that indicate that LLMs can be hypersensitive to specific parts of the input prompts. This is due to the fact that there is a significantly different performance of specific combinations of prompts and problems, while the prompts can have very minor difference, e. g. the difference between Base and Expert is one simple sentence, similarly for the difference between CoT and Expert CoT. In other words, adding the same extra sentence about expertise at the beginning of the prompt led one instance to be significantly better, and another being significantly worse.

Furthermore, magic sentences like "take a deep breath" and "If you don't get this right, I will be fired and lose my job" do not seem to improve CoT-based prompts in most cases for both correctness and helpfulness.

The Ignorant prompt is the worst model across the board. This suggests that the model can internalise limiting beliefs that would actually make it perform worse. It is likely that

TABLE II

RESULTS OF THE DIFFERENT PROMPTS ON ALL PROBLEMS. SHOWING THE AMOUNT OF EASY-TO-PARSE EXAMPLES, THEIR CLASSIFICATION ACCURACY AND UNWEIGHTED AVERAGE RECALL (UAR) PERFORMANCE MEASURES. *,** INDICATE A STATISTICALLY SIGNIFICANT DIFFERENCE COMPARED TO THE BASE PROMPT, WITH p-VALUES $< 5\,\%$ AND $< 1\,\%$, RESPECTIVELY, CALCULATED BY A TWO-TAILED PERMUTATION TEST. MARKED IN BOLD ARE THE PROMPTS WITH HIGHEST PERFORMANCE METRICS ACROSS EACH PROBLEM AND PERFORMANCE MEASURE. ADDITIONALLY, TWO BASELINES FROM [9] ARE INCLUDED FOR REFERENCE, NAMELY AN END-TO-END LSTM MODEL AND AN MLP BASED ON ROBERTA FEATURES.

Prompt	Parsed [%]			ACC [%]			UAR [%]		
	Sentiment	Toxicity	Sarcasm	Sentiment	Toxicity	Sarcasm	Sentiment	Toxicity	Sarcasm
End-to-end LSTM	-	-	-	77.3	82.1	60.9	77.3	83.7	65.6
RoBERTa	-	-	-	86.5	85.6	91.8	86.5	87.0	91.8
Base	97.5	99.8	**100.0**	77.9	**87.9**	62.2	78.4	**87.9**	60.5
Expert	98.8**	99.9	95.8**	77.2	87.6	**72.1****	77.8	87.4	**73.2****
Expert Detailed	99.7**	**100.0**	**100.0**	78.4	87.6	63.2	78.8	87.8	60.5
Ignorant	**99.9****	99.6	**100.0**	71.3**	67.2**	58.3*	72.2**	63.3**	52.1**
Gambler	98.4	99.5	**100.0**	75.6**	78.1**	61.2	76.2**	76.0**	58.3
Greedy Gambler	98.2	99.5	**100.0**	76.7	72.8**	59.6	77.3	70.0**	56.1**
Python Expert	98.6*	99.6	**100.0**	75.6**	70.5**	59.5	76.1**	67.3**	54.8**
CoT	89.6**	97.5**	99.6	**80.2**	80.7**	64.9	**80.7**	80.6**	66.0**
CoT-DB	91.9**	97.6**	99.3*	**80.2**	80.2**	63.6	80.6	79.7**	64.2*
CoT-fired	81.8**	93.2**	98.8**	78.7	80.5**	65.6*	79.1	80.4**	66.2**
CoT-DB-fired	84.3**	90.8**	98.3**	78.9	80.6**	64.7	79.4	80.5**	64.8**
Expert CoT	90.5**	96.3**	99.8	79.3	71.4**	63.3	79.7	69.6**	63.7
Expert CoT-DB	90.1**	98.4**	99.4*	78.9	77.8**	65.2	79.5	76.4**	65.3**
CoT-verify	92.9**	99.5	**100.0**	79.8	81.5**	61.5	80.1	81.0**	59.5
CoT-DB-verify	92.7**	99.6	**100.0**	80.0	84.2**	60.0	80.6	83.9**	55.8*

the model is 'acting' as an ignorant, by changing some of the answers it can confidently predict, since the degradation in hard problems, e. g. sarcasm detection, is not as severe as in easier problems, e. g. toxicity detection.

Furthermore, the performances of the Gambler, Greedy Gambler, and Python Expert fall between the Ignorant prompt and the aforementioned top prompts. The Python Expert prompt gives an identity of a helpful expert. One could have hypothesised that an LLM with such an identity will still try to assist the user to the best of its knowledge, however, using it still leads the model to significantly underperform just because the expertise is irrelevant. Similarly, trying an incentive like financial gain for the Gambler prompts reaches similarly poor results. These observations conclude the crucial importance of 'correctly' prompting incentives to LLMs to reach the best performance.

Eventually, the Expert Detailed prompt is the most successful at parsing the responses, with statistically significant difference in the sentiment analysis, followed by the straightforward prompts. CoT-based prompts are significantly worse at following the instructions to produce easy-to-parse responses. Most prompts are quite reasonable in producing easy-to-parse responses, except for the verbose CoT-based prompts which are significantly worse in most cases; this can be significantly improved by using follow-up prompts to solidify the predictions after verbose responses. Moreover, easy-to-parse does not translate to better performance.

D. Limitations

This study presents a method for evaluating sensitivity due to prompting or generation parameters, however, the study was only conducted on ChatGPT using affective computing problems. The reasons for this are to isolate issues that are mainly about prompting or generation parameters, and to limit the influence of problems that could have multiple hard-to-judge effectively correct solutions, e. g. solving programming exercises (which can have multiple correct solutions), and question answering (which can be outdated). However, an extension of this study to other affective computing problems or other problems in general is crucial to reach general conclusions about the impact of prompting. Moreover, this study needs to be extended to other LLMs (for example, open-source LLMs like the families of LLaMa [25] or Mistral [26]) to investigate which conclusions are universal to LLMs in general.

V. CONCLUSION

In this paper, we introduced a method to investigate prompt engineering for foundation models, we demonstrated it on ChatGPT for three affective computing tasks. We conducted sensitivity analysis on the temperature parameter T and the parameter top-p in response generation, which concluded that conservative predictions with lower $T \leq 0.3$ values or top-$p \leq 0.7$ yield better and stable performance. Increasing T or top-p beyond that generally worsened the performances.

We evaluated various prompting techniques, which demonstrated that straightforward prompts or giving expert identity often yield near-best performances. Chain-of-Thought prompts excelled the most in some problems, but fell short in others, and generally they were the worst at following formatting instructions, resulting in complex-to-parse responses. Magic sentences like 'take a deep breath' and 'if you don't get this right, I will be fired and lose my job' did not yield significant differences. We also found prompt hypersensitivity in few scenario, where the performance significantly changed based on a minor change in the prompt. Furthermore, we found that specifying detailed output formats facilitates easy parsing. On the other hand, irrelevant expertise or misaligned incentives can harm results severely.

Our research introduced a method for analysing aspects impacting prompt engineering. Furthermore, it shed light on the role of prompt engineering for foundation models like ChatGPT. Future work should use the presented method to further extend the study to other affective computing problems and to several foundation models, including open-source ones, to assess the generality of the aforementioned conclusions.

Ethical Impact Statement

Our research delves into the intricacies of prompt sensitivity in LLMs, especially within the realm of affective computing. A critical ethical consideration is the examination of how different prompts, such as the Ignorant prompt, could possibly manipulate LLMs to propagate misinformation, since it could technically manipulate the model to give inaccurate results on affective computing tasks. This exploration is pivotal as it underscores the susceptibility of LLMs to be influenced by the nature of the prompt, thereby potentially leading to biased or misleading outputs.

The study also critically analyses the long-term implications of using manipulative prompts, such as those implying dire consequences for incorrect responses, e. g., "if you don't get this right, I will be fired". Our findings suggest that such prompts do not significantly enhance the performance of following given instructions, which is an essential insight for the AI research and data collection communities. By demonstrating that these manipulative strategies do not substantially affect outcomes, we contribute to discouraging the incorporation of such prompts into future datasets, thus mitigating the risk of cultivating future models that are more responsive to manipulation and possibly undermining safety or other ethical concerns.

Acknowledgment

Björn W. Schuller is also with the Munich Data Science Institute (MDSI), the Munich Center for Machine Learning (MCML), and the Konrad Zuse School of Excellence in Reliable AI (relAI), all in Munich, Germany.

References

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child *et al.*, "Language Models are Few-Shot Learners," in *NeurIPS*, 2020, pp. 1877–1901.
[2] OpenAI, "GPT-4 Technical Report," *arXiv:2303.08774*, 2023.
[3] A. Hendy, M. Abdelrehim, A. Sharaf, V. Raunak, M. Gabr, H. Matsushita, Y. J. Kim, M. Afify, and H. H. Awadalla, "How Good Are GPT Models at Machine Translation? A Comprehensive Evaluation," *arXiv:2302.09210*, 2023.
[4] J. Li, H. Li, Z. Pan, and G. Pan, "Prompt ChatGPT In MNER: Improved multimodal named entity recognition method based on auxiliary refining knowledge from ChatGPT," *arXiv:2305.12212*, 2023.
[5] M. M. Amin, E. Cambria, and B. W. Schuller, "Can ChatGPT's Responses Boost Traditional Natural Language Processing?" *IEEE Intelligent Systems*, vol. 38, no. 5, 2023.
[6] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," in *NeurIPS*, vol. 35, 2022, pp. 27 730–27 744.
[7] J. Wei, X. Wang, D. Schuurmans, M. Bosma, b. ichter, F. Xia, E. Chi, Q. V. Le, and D. Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," in *Advances in Neural Information Processing Systems*, vol. 35. Curran Associates, Inc., 2022, pp. 24 824–24 837.
[8] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, "Tree of Thoughts: Deliberate Problem Solving with Large Language Models," 2023.
[9] M. M. Amin, R. Mao, E. Cambria, and B. W. Schuller, "A Wide Evaluation of ChatGPT on Affective Computing Tasks," *IEEE Transactions on Affective Computing*, 2024.
[10] M. M. Amin, E. Cambria, and B. W. Schuller, "Will Affective Computing Emerge from Foundation Models and General AI? A First Evaluation on ChatGPT," *IEEE Intelligent Systems*, vol. 38, no. 2, pp. 15–23, 2023.
[11] K. He, R. Mao, Q. Lin, Y. Ruan, X. Lan, M. Feng, and E. Cambria, "A Survey of Large Language Models for Healthcare: from Data, Technology, and Applications to Accountability and Ethics," *arXiv:2310.05694*, 2023.
[12] V. Liévin, C. E. Hother, and O. Winther, "Can Large Language Models Reason about Medical Questions?" *arXiv:2207.08143*, 2023.
[13] R. Mao, Q. Liu, K. He, W. Li, and E. Cambria, "The Biases of Pre-Trained Language Models: An Empirical Study on Prompt-Based Sentiment Analysis and Emotion Detection," *IEEE Transactions on Affective Computing*, vol. 14, no. 3, pp. 1743–1753, 2023.
[14] C. Yang, X. Wang, Y. Lu, H. Liu, Q. V. Le, D. Zhou, and X. Chen, "Large Language Models as Optimizers," *arXiv preprint arXiv:2309.03409*, 2023.
[15] Y. Lu, M. Bartolo, A. Moore, S. Riedel, and P. Stenetorp, "Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. Dublin, Ireland: ACL, 2022, pp. 8086–8098.
[16] M. Sclar, Y. Choi, Y. Tsvetkov, and A. Suhr, "Quantifying Language Models' Sensitivity to Spurious Features in Prompt Design or: How I learned to start worrying about prompt formatting," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=RIu5lyNXjT
[17] C. Zheng, F. Yin, H. Zhou, F. Meng, J. Zhou, K.-W. Chang, M. Huang, and N. Peng, "On Prompt-Driven Safeguarding for Large Language Models," in *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*, 2024. [Online]. Available: https://openreview.net/forum?id=lFwf7bnpUs
[18] A. Go, R. Bhayani, and L. Huang, "Twitter Sentiment Classification using Distant Supervision," *CS224N project report, Stanford*, vol. 1, no. 12, p. 2009, 2009.
[19] R. Misra and P. Arora, "Sarcasm Detection using News Headlines Dataset," *AI Open*, vol. 4, pp. 13–18, 2023.
[20] I. Sutskever, J. Martens, and G. E. Hinton, "Generating Text with Recurrent Neural Networks," in *Proceedings of the 28th International Conference on Machine Learning*. Madison, WI, USA: Omnipress, 2011, pp. 1017–1024.
[21] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for Boltzmann machines," *Cognitive Science*, vol. 9, no. 1, pp. 147–169, 1985.
[22] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=rygGQyrFvH
[23] B. Schuller, S. Steidl, A. Batliner, A. Vinciarelli, K. Scherer, F. Ringeval, M. Chetouani, F. Weninger, F. Eyben, E. Marchi, M. Mortillaro, H. Salamin, A. Polychroniou, F. Valente, and S. Kim, "The INTERSPEECH 2013 Computational Paralinguistics Challenge: Social Signals, Conflict, Emotion, Autism," in *Proceedings INTERSPEECH*, 2013, pp. 148–152.
[24] P. Good, *Permutation Tests: A Practical Guide to Resampling Methods for Testing Hypotheses*. New York City, NY, USA: Springer, 1994.
[25] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra *et al.*, "Llama 2: Open Foundation and Fine-Tuned Chat Models," *arXiv preprint arXiv:2307.09288*, 2023.
[26] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier *et al.*, "Mistral 7B," *arXiv preprint arXiv:2310.06825*, 2023.