2024 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0 & IoT) | 979-8-3503-8582-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/MetroInd4.0IoT61288.2024.10584130

Prompt Evolution Through Examples for Large Language Models–A Case Study in Game Comment Toxicity Classification

Pittawat Taveekitworachai*, Febri Abdullah†, Mustafa Can Gursesli‡
Antonio Lanata§, Andrea Guazzini¶, Ruck Thawonmas‖

{*†*Graduate School,* ‖*College*} *of Information Science and Engineering*, *Ritsumeikan University*, Ibaraki, Osaka, Japan
‡§Department of Information Engineering, *University of Florence*, Florence, Italy
¶Department of Education, Literatures, Intercultural Studies, Languages and Psychology, *University of Florence*, Florence, Italy
{*gr0609fv, †gr0397}@ed.ritsumei.ac.jp,
{‡mustafacan.gursesli, §antonio.lanata, ¶andrea.guazzini}@unifi.it, ‖ruck@is.ritsumei.ac.jp

***Abstract*—This paper presents a novel approach for automatic prompt optimization (APO) using a large language model (LLM) as an optimizer, named Prompt Evolution Through Examples (PETE). The approach draws inspiration from evolutionary computation for the prompt evolution stages. We aim to aid in developing prompts for use in systems classifying toxic content including game community moderator-assist tools. While traditional approaches are useful for developing these tools, they have various shortcomings where LLMs can potentially mitigates these issues. LLMs accept prompts as inputs to condition generated outputs. However, to design a prompt with the best performance in this task, fine-grained adjustments are usually required and should be automated through the APO process instead of a manual approach, which is often time-consuming. In this study, ChatGPT and GPT-4 are utilized as both task performers and prompt optimizers for comparisons across models. The results indicate that PETE improves the performance of the target task up to 56.14% from a performance of an initial prompt, compared to only up to 49.15% using a standard mutation evolution. Optimized prompts are provided for future utilization in other game community moderation tools. We also recommend that future studies explore more cost-effective approaches for evaluation using LLMs to enhance the benefits of APO.**

***Index Terms*—Evolutionary computation, Prompt engineering, ChatGPT, GPT-4, Errorful learning**

I. Introduction

Game messages or comments are crucial in many aspects for game community moderators [1], game streamers [2], and the audience [3]. Audiences watching game live streams usually provide these comments to influence the streamers' decisions [4] or voice their opinions to influence the course of the game, as in audience participation games [5]. One particular application of game comments is to interact with members in game online communities [1]. Game communities are where like-minded individuals share their opinions about a particular game, form groups for game activities, or provide feedback to game developers [1]. These comments play a crucial role in maintaining engagement within the game community [1]. However, not all game comments are well-intentioned and may be crafted for harassment, bullying, or trolling [6].

Game comments may contain harsh words, often serving as a medium to exhibit toxic behaviors in games [7]. These negative messages not only adversely affect target receivers but also unintended audiences who consume these messages, including moderators and other community members. Allowing these messages to persist in the community without proper management could potentially have negative impacts on the community and, at worst, may lead to the community's disbandment [8]. Community moderation is an essential aspect of maintaining a healthy community. Various techniques have been proposed to assist community moderators, such as detecting potentially toxic content [8], content filtering [9], and flagging inappropriate content [10].

While these proposed techniques are effective, they exhibit certain shortcomings. For instance, machine learning models designed for toxic comment detection may excel at straightforward messages but could falter when confronted with more complex or indirect messages due to the inherent complexity of natural language [11]. The process of data collection and model training may prove to be expensive to execute [12]. Furthermore, these trained models may lack flexibility in adapting to emerging trends or changes in regulations.

Large language models (LLMs) are text-to-text generative models trained on massive text corpora [13]. LLMs showcase their potential in various tasks, including those related to community moderation, such as message toxicity classification [14], toxic content detection [8], and text sentiment analysis [15]. LLMs also excel in various natural language tasks [13], possessing the ability to learn the definitions of new words and incorporate them to perform tasks during inference [16]. Furthermore, the emergent abilities of LLMs grant them multiple zero-shot capabilities [17], allowing users to provide instructions via prompts without additional examples and still achieve good performance. This reduces the need for extensive data collection for these particular tasks. Additionally, the in-context learning abilities of LLMs [18] enable the models to learn new concepts or required information at runtime through

979-8-3503-8582-3/23/$31.00 ©2024 IEEE

prompts provided to the models. This nature of prompt is advantageous when rules or requirements need to be updated.

While constructing a prompt may be straightforward, in cases where we want to maximize the model's performance, a prompt optimization process is required, as changing only a tiny bit of prompts could potentially drastically affect the model's performance [19]. Manually adjusting these fine-grained details is not only time-consuming but also a trial-and-error process. While prompt engineers do exist, it is currently uncommon, and developers of systems usually bear the responsibility for prompt optimizations as business requirements may demand the best performance possible to satisfy business needs. Automatically optimizing these prompts is beneficial in this regard, freeing humans who are usually responsible for more meaningful jobs. Existing studies have proposed techniques for automatic prompt optimization (APO) [20]–[22].

We observe that the optimization process for APO shares similarities to the concept of evolution in evolutionary computation (EC). In fact, some of these studies are inspired by or borrow concepts from EC, such as mutation and cross-over operators. Notable examples in this regard are *EvoPrompt* [21] and *Promptbreeder* [22]. These methods leverage the principles of evolution to optimize prompts, demonstrating the potential of EC in this domain.

EvoPrompt introduces prompts for mutation and cross-over operators and describes how these operators function in natural language. They also outline the expected results from these operations and provide a few examples. The chosen algorithm for this approach is the standard genetic algorithm (GA). Similarly, *Promptbreeder* introduces various types of mutation operators to mutate both task- and mutation-prompts. For the evolution algorithm, they opted for the standard binary tournament GA [23]. These methods demonstrate the potential of EC principles in the field of APO.

In the pursuit of providing the best performance of LLMs by discovering the most suitable prompts through the APO process, we propose novel APO techniques that utilize various existing concepts from PE and EC. Inspired by (1) studies that leverage concepts from EC for APO and (2) observed performance increases from providing examples and reasons to the model in PE, we introduce a novel approach for APO enhanced through examples, named Prompt Evolution through Examples (PETE).

In the evolution process, which we only utilize a mutation operator, we also provide incorrect examples along with predicted reasons why the model in the previous generation thought they were correct. This is inspired by errorful learning in humans, where individuals learn through mistakes and observe what to avoid, correcting behaviors accordingly. Showing incorrect examples during the evaluation process shares similarity with errorful learning in humans [24]. This approach aids the model in developing guidelines regarding what information it should include or avoid, guiding the model in the optimization space toward a more optimal point.

We also draw inspiration from an EC technique called

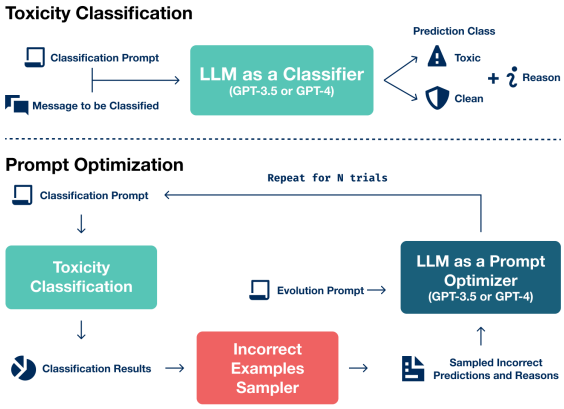


Fig. 1: An overview of the proposed approach: For game comment toxicity classification, we employ an LLM as a classifier and request the model to also provide explanations for its predictions. The APO process utilizes sampled incorrect predictions, and the LLM also serves as a prompt optimizer to enhance and generate a new prompt. The optimization process is iterated for N generations.

(1+1)-Evolutionary Strategy (ES) [25]. Inspired by (1+1)-ES, a new prompt is generated from the existing one, and the best-performing prompt is carried over to the next generation. Due to its simplicity, (1+1)-ES is chosen as our algorithm. Moreover, having only one best-performing prompt in each generation emphasizes the effectiveness of our LLMs-based evolutionary operator, which relies on the ability of LLMs to reason from incorrect examples, as mentioned earlier. To support the community and future studies in utilizing our approach, we have open-sourced our source code, prompts, and raw data[1]. Our contributions in this study are as follows:

- We propose a new approach for APO, PETE, by incorporating incorrect examples and relying on LLMs' reasoning abilities during the prompt evolution process.
- We evaluate the performance of our approach compared to a standard mutation evolution using GPT-3.5 Turbo and GPT-4, and provide a detailed discussion of the process.
- We provide and analyze the best-performing prompts resulting from PETEfor further incorporation into game moderation tools and place them in the context of PE, assessing their generalizability to other models.

II. Methods

This section outlines details regarding the chosen task for evaluating the performance of the proposed approach and the workings of PETE. For the selected task used as a test bed for our approach, we use game comment toxicity classification. Details about dataset preparation for our task are provided in Section II-A. Next, we describe the task with evaluation details in Section II-B. Finally, we illustrate our APO in Section II-C. An overview of the process is depicted in Fig. 1.

[1]https://github.com/Pittawat2542/pete-prompt-optimization

A. Dataset Preparation

To demonstrate the performance of our proposed approach, we select game comment toxicity classification as our test bed task. We utilize a publicly available dataset of game toxic comments[2]. However, the dataset exhibits severe imbalance across available prediction classes. The majority of data records in this dataset are labeled as "`clean`" with approximately 2500 records, while the second largest class is "`toxic`" with approximately 500 records. Other classes have very few data records compared to the aforementioned two classes. Therefore, we decide to simplify the task into binary toxicity classification, i.e., predicting whether the game comment is "`clean`" or "`toxic`", due to this imbalance.

Furthermore, it is worth noting that each game comment in the dataset can be associated with multiple classes. To simplify our task, we only consider comments associated with a single class in the original dataset, specifically either `clean` or `toxic`. To achieve this, we downsample the dataset to ensure an equal distribution across these two classes. We randomly select 400 messages, 200 messages from each class, which we use as our testing set–the set for which the task prompt is optimized. We select this smaller number of records not only due to data availability but also to reduce computational load as evaluation using LLMs is computationally intensive.

B. Binary Game Comment Toxicity Classification

The game comment toxicity classification task is not only useful for game community moderation, but it also has some preferable characteristics for APO. This task includes a ground truth label for each record, which aids in calculating the accuracy as a score representing a particular generation. Moreover, a tuple of predicted and actual labels included in the incorrect examples given to the model during the evolution process are required for our approach. This allows the model to utilize this information to optimize a prompt for the next generation, i.e., by comparing ground truth and predicted labels as well as LLM-generated reasons.

We select two widely-utilized models, GPT-3.5 Turbo and GPT-4, as our models of choice for performing this task. This choice not only allows us to assess the capabilities of these models for the task but also offers an opportunity to evaluate the generalization of optimized prompts across different models. We employ the chosen LLMs as a classifier, performing the task by providing a classification prompt. The design of the prompt is kept as simple as possible, providing only the minimum information required to carry out the task and specifying the desired output format.

All LLM settings are left at their default values, except for the sampling temperature, which we set to 0 to ensure more deterministic behavior during classification. For accuracy calculation, i.e., evaluation, we compare the predicted label with the ground truth and calculate the task's accuracy by determining the percentage of correct predictions among all predictions.

[2]https://github.com/pl2599/GGWP-Toxic-Behavior

C. PETE

During the APO, we select the best-performing classification prompt from the previous generation as the base prompt. We instruct the LLMs using an evolution prompt, drawing inspiration from an evolutionary strategy called (1+1)-ES [25]. In each generation, we retain only the best-performing offspring, which could be either the recently mutated prompt or the previously best-performing prompt, depending on their performance. The algorithm then generates a new offspring by mutating the best-performing prompt using the evolution prompt and an LLM as a mutator. We utilize the accuracy of the prediction task as the score used to assess the performance of each generation.

The evolution prompt instructs the LLMs to modify only the classification prompt based on the provided instructions. In the case of the evolution prompt of our proposed approach, the prompt also instructs the LLM to incorporate knowledge from the sampled incorrect examples, along with the expected correct labels. For incorrect examples included in the PETE evolution prompt, we sample as many incorrect examples as possible without exceeding the maximum context window of the respective model.

We choose to provide incorrect examples because we believe that incorrect examples will provide models with information on what they should do or what content they should include more. This approach is similar to errorful learning [24] in humans. Avoiding errors may be a straightforward approach. However, learning through incorrect examples to identify deviations from the expected outcomes could also be effective, as shown in previous studies. Furthermore, we also rely on LLMs' reasoning abilities to analyze the inaccuracies and attempt to generate their own reasons and propose the best approach to improve the next generation.

It is crucial to note that in the case of the standard mutation evolution, we do not provide any incorrect examples to the model. Instead, we retain the same set of instructions. This can be seen as equivalent to a typical mutation where parts of the prompt may change randomly.

We use the same LLM for both classification and evolution, i.e., both as a classifier and as an optimizer. For example, if the classification is performed using GPT-3.5 Turbo, we also use GPT-3.5 Turbo for the evolution process. The sampling temperature for the evolution is set to its default value, which is 1, in contrast to 0 during the classification phase. This setting allows for an optimal balance between creativity and readability of the results.

The newly evolved classification prompt goes through another round of evaluation and evolution. This cycle continues until the specified number of rounds is reached, which in this case is 10. It is worth noting that we limit our evolution to 10 rounds, which is a relatively small number compared to a number in typical EC. However, as indicated in *EvoPrompt* [21], the evaluation process using LLMs can be computationally expensive in evaluation phase. All results from the evaluation and evolution process are recorded for further analysis.

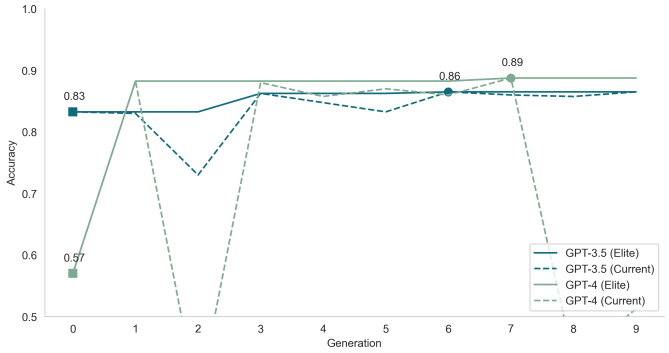
III. Results and Discussions

We conducted 10 generations of APO using PETE for each model, i.e., GPT-3.5 Turbo and GPT-4, separately. Additionally, we performed the APO using the standard mutation operator as a baseline. We discussed the experimental results in Section III-A and explored the best-performing prompts obtained from the optimization processes in Section III-B.

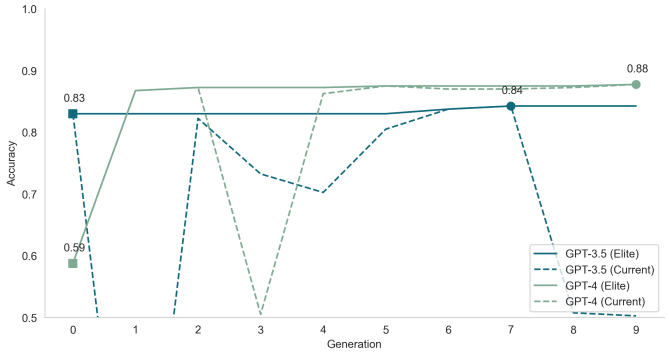
A. Effectiveness of PETE

The results of the optimization processes are presented in Fig. 2. This figure shows the accuracy, i.e., the performance of elite prompts in solid lines, and the performance of the evolved prompt of each generation in dashed lines. If the performance did not improve from the previous generation, the best-performing classification prompt so far, i.e., the elite prompt, is retained as a base prompt for the evolution process.

Fig. 2: These figures display the performance of models across 10 generations of APO using PETE in Fig. 2a and a standard mutation in Fig. 2b.



(a) Performance of the prompts optimized using our proposed approach. GPT-3.5 Turbo demonstrates a stable progress with an improvement of up to 3.61%. In contrast, GPT-4 shows more significant improvement, up to 56.14%, but with more fluctuations across generations.



(b) Performance of the prompts optimized using standard mutation reveals less improvement in GPT-3.5 Turbo and signs of greater fluctuation, even to the extent of accuracy dropping to 0 in one generation. On the other hand, GPT-4 shows a similar improvement, up to 49.15%, and experiences similar fluctuations but with less improvement compared to our proposed approach.

Both approaches can increased the performance when comparing the best-performing and the initial classification prompts. This shows that the LLMs are naturally good optimizers; additional improvements like PETE help push the performance of the optimizer further. For the standard mutation, our baseline, using GPT-3.5 Turbo for both classification and evolution, the performance improved by 1.20% from 0.83 to 0.84 in the GPT-3.5 Turbo case. Using GPT-4, we observed improved accuracy from 0.59 to 0.88, which is equivalent to a 49.15% increase. Our proposed approach, PETE, shows more promising performance. Using GPT-3.5 Turbo increased the performance from 0.83 to 0.86, equivalent to 3.61%, while we found that the accuracy improved to 56.14% from 0.57 to 0.89 using GPT-4.

While our proposed approach shows great potential for improvement when utilizing GPT-3.5 Turbo compared to the standard mutation, it still maintains its effectiveness with a slight drop for GPT-4. We hypothesize that this could be due to the *emergent abilities* that exist only in GPT-4, making it a naturally better prompt optimizer. Thus, PETE shows less effectiveness. Still, the performance of the optimized prompts from our approach using GPT-4 is better than the standard mutation.

Another noteworthy observation is the fluctuation of performance of evolved prompts in each generation during the APO (dashed lines). Without examples to guide as in our approaches, we observed that utilizing standard mutation coupled with GPT-3.5 Turbo exhibits greater fluctuations in accuracy. For instance, during generation 1, the accuracy dropped to 0 and then rebounded to approximately 0.82. This demonstrates that randomly mutating prompts could lead to a higher variability in an optimization path.

In contrast, when comparing GPT-3.5 Turbo using PETE, we noticed more stable performance during the process. This indicates the effectiveness of our approach in guiding the models in the optimization space with more stability. On the other hand, GPT-4 shows similar signs of fluctuation in both approaches. This highlights the limitations of our current approach, as its effectiveness may decrease when used with stronger, larger, and more costly LLMs, potentially due to the influence of their inherent *emergent abilities*.

While it is true that we only performed 10 generations, as previously mentioned, the APO process is costly, both in terms of compute resources and time. The process can be broken down into two main sub-processes: (1) the evaluation of the task prompt and (2) the evolution of the task prompt. The most expensive part is the evaluation of the task prompt. Similar to the existing study [21], we only performed 10 generations in our study. We believe that this also underscores the importance for future studies to explore methods to reduce the cost of the prompt optimization process. Nevertheless, we observed an increase in performance with only 10 generations, including generation 0.

We also note that we only utilized GPT-3.5 Turbo and GPT-4 via the OpenAI API, which handle most of the compute infrastructure requirements for us. Serving LLMs at such scale is not a trivial task and requires various careful considerations. Infrastructure to run models at this scale while providing

responses in a responsive manner requires not only powerful hardware, but also supporting systems, e.g., networking for distributed computation, HVAC for data centers, and personnel for monitoring the systems.

We also argue that it is not trivial to utilize smaller LLMs that can run on commodity hardware to solve this issue. This is due to the fact that APO, in similar manners to PETE requires LLMs that are strong in reasoning and language capabilities to understand feedback given and optimize the prompt accordingly. This is partly due to a scaling law of emergent abilities [13]. Only those large enough models possess such reasoning abilities. Therefore, using large models is inevitable in the current landscape of LLMs.

The utilization of these large models also implies concerns regarding environmental impact. It is not trivial that training and serving LLMs at scale are impactful to the environment, and APO, which is inherently an expensive process in heavy utilization of LLMs, is also costly. Therefore, we urge future studies to promptly further investigate mitigation strategies to address these concerns.

B. Best-Performing Prompts Analysis

In this section, we analyzed the best-performing prompts from all APO performed in this study. Using the standard mutation baseline, the best-performing prompt from GPT-3.5 Turbo had additional formatting added using Markdown syntax. This additional formatting likely serves as a marker to denote the separation of sections, providing a clearer separation and lessening the opportunity for misunderstanding. For example, it included "**" to signify bold font or used "-" for bullet list items. This also highlights the importance of formatting prompts using additional symbols to help construct the structure of the prompts. The effectiveness of these added symbols is potentially due to the fact that LLMs are trained on a massive amount of data that contains a number of records including these Markdown symbols, and the models gain an understanding of the meaning of these symbols.

Additionally, a clearer instruction with detailed information on each possible class prediction and examples of each possible class was included. Similarly, in the case of GPT-4, detailed instructions and examples were added to the best-performing prompt. GPT-4 also offers additional explanations for each key of the output, which helps ensure that the correct format is produced. When using PETE for optimized prompts, we observed that the best-performing prompts using either GPT-3.5 Turbo and GPT-4 provided a similar enhancement to the baseline approach.

We also explored the generalization of the optimized prompts from our proposed approach. To achieve this, we evaluated the best-performing prompts on another model. Specifically, we assessed the prompt optimized for GPT-3.5 Turbo using GPT-4 and vice versa. We observed that while the optimized prompts may not retain their peak performance; however, they still maintain better or similar accuracy compared to the starting classification prompts, achieving 0.83 for the optimized prompt from GPT-3.5 Turbo evaluated on GPT-4 and 0.85 for the prompt from GPT-4 on GPT-3.5 Turbo. This outcome also suggests the effectiveness of the PE techniques incorporated in both prompts.

IV. Conclusions

We introduced a novel approach for APO inspired by EC, few-shot prompting, and the reasoning abilities of LLMs, which we named PETE. Our approach involves providing incorrect prediction examples, akin to errorful learning in humans, during the prompt evolution to guide the optimization process. The approach in this study was proposed to discover the best-performing prompts for game comment toxicity classification, a task crucial in community moderation tools. Our evaluation focused on this classification task, and we conducted experiments using GPT-3.5 Turbo and GPT-4.

Our optimization approach yielded an increase in performance for this task of up to 56.14%, compared to a maximum of 49.15% achieved with the standard mutation. It is worth noting that the performance of optimized prompts may not maintain peak performance on other models. These prompts were also made publicly available for use in developing LLMs-integrated systems for game message toxicity classification or game community moderation tools in general. We also observed that APO is a costly process, primarily due to the evaluation phase, and we suggest that future studies should explore ways to reduce costs, potentially through the use of surrogate models.

References

[1] C. Ruggles, G. Wadley, and M. R. Gibbs, "Online Community Building Techniques Used by Video Game Developers," in *Entertainment Computing - ICEC 2005*, F. Kishino, Y. Kitamura, H. Kato, and N. Nagata, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 114–125. [Online]. Available: https://link.springer.com/chapter/10.1007/11558651_12

[2] T. Wulf, F. M. Schneider, and J. Queck, "Exploring Viewers' Experiences of Parasocial Interactions with Videogame Streamers on Twitch," *Cyberpsychology, Behavior, and Social Networking*, vol. 24, no. 10, pp. 648–653, 2021, pMID: 33535010. [Online]. Available: https://doi.org/10.1089/cyber.2020.0546

[3] P. Lessel, A. Vielhauer, and A. Krüger, "Expanding Video Game Live-Streams with Enhanced Communication Channels: A Case Study," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ser. CHI '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1571–1576. [Online]. Available: https://doi.org/10.1145/3025453.3025708

[4] P. Paliyawan, R. Thawonmas, K. Sookhanaphibarn, and W. Choensawat, "Audience participation fighting game: Exploring social facilitation for an enhanced APG experience," *Heliyon*, vol. 10, no. 2, p. e23967, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405844023111753

[5] J. Seering, S. Savage, M. Eagle, J. Churchin, R. Moeller, J. P. Bigham, and J. Hammer, "Audience Participation Games: Blurring the Line Between Player and Spectator," in *Proceedings of the 2017 Conference on Designing Interactive Systems*, ser. DIS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 429–440. [Online]. Available: https://doi.org/10.1145/3064663.3064732

[6] S. Adinolf and S. Turkay, "Toxic Behaviors in Esports Games: Player Perceptions and Coping Strategies," in *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts*, ser. CHI PLAY '18 Extended Abstracts. New York, NY, USA: Association for Computing Machinery, 2018, p. 365–372. [Online]. Available: https://doi.org/10.1145/3270316.3271545

[7] H. Kwak and J. Blackburn, "Linguistic Analysis of Toxic Behavior in an Online Video Game," in *Social Informatics*, L. M. Aiello and D. McFarland, Eds. Cham: Springer International Publishing, 2015, pp. 209–217. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-15168-7_26

[8] L. Li, L. Fan, S. Atreja, and L. Hemphill, ""HOT" ChatGPT: The Promise of ChatGPT in Detecting and Discriminating Hateful, Offensive, and Toxic Comments on Social Media," *ACM Trans. Web*, feb 2024. [Online]. Available: https://doi.org/10.1145/3643829

[9] J. Seering, T. Wang, J. Yoon, and G. Kaufman, "Moderator engagement and community development in the age of algorithms," *New Media & Society*, vol. 21, no. 7, pp. 1417–1443, 2019. [Online]. Available: https://doi.org/10.1177/1461444818821316

[10] Y. Kou and X. Gui, "Flag and Flaggability in Automated Moderation: The Case of Reporting Toxic Behavior in an Online Game Community," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3411764.3445279

[11] B. van Aken, J. Risch, R. Krestel, and A. Löser, "Challenges for Toxic Comment Classification: An In-Depth Error Analysis," in *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, D. Fišer, R. Huang, V. Prabhakaran, R. Voigt, Z. Waseem, and J. Wernimont, Eds. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 33–42. [Online]. Available: https://aclanthology.org/W18-5105

[12] Y. Roh, G. Heo, and S. E. Whang, "A Survey on Data Collection for Machine Learning: A Big Data - AI Integration Perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1328–1347, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/8862913

[13] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler *et al.*, "Emergent Abilities of Large Language Models," *Transactions on Machine Learning Research*, 2022, survey Certification. [Online]. Available: https://openreview.net/forum?id=yzkSU5zdwD

[14] Y.-S. Wang and Y. Chang, "Toxicity Detection with Generative Prompt-based Inference," 2022. [Online]. Available: https://arxiv.org/abs/2205.12390

[15] Z. Wang, Q. Xie, Z. Ding, Y. Feng, and R. Xia, "Is ChatGPT a Good Sentiment Analyzer? A Preliminary Study," 2023. [Online]. Available: https://arxiv.org/abs/2304.04339

[16] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[17] J. Wei, M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned Language Models are Zero-Shot Learners," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=gEZrGCozdqR

[18] C. Olsson, N. Elhage, N. Nanda, N. Joseph, N. DasSarma, T. Henighan, B. Mann, A. Askell, Y. Bai, A. Chen *et al.*, "In-context Learning and Induction Heads," 2022. [Online]. Available: https://arxiv.org/abs/2209.11895

[19] 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

[20] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba, "Large Language Models are Human-Level Prompt Engineers," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=92gvk82DE-

[21] Q. Guo, R. Wang, J. Guo, B. Li, K. Song, X. Tan, G. Liu, J. Bian, and Y. Yang, "Connecting large language models with evolutionary algorithms yields powerful prompt optimizers," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=ZG3RaNIsO8

[22] C. Fernando, D. Banarse, H. Michalewski, S. Osindero, and T. Rocktäschel, "Promptbreeder: Self-Referential Self-Improvement Via Prompt Evolution," 2023. [Online]. Available: https://arxiv.org/abs/2309.16797

[23] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 3, pp. 111–128, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210650211000435

[24] J. Metcalfe, "Learning from Errors," *Annual Review of Psychology*, vol. 68, no. 1, pp. 465–489, 2017, pMID: 27648988. [Online]. Available: https://doi.org/10.1146/annurev-psych-010416-044022

[25] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies–a comprehensive introduction," *Natural computing*, vol. 1, pp. 3–52, 2002. [Online]. Available: https://link.springer.com/article/10.1023/A:1015059928466