Influences of Sycophancy in Large Language Models

Anton Olson
ols00559@morris.umn.edu
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

Abstract

Sycophancy, the tendency for *large language models (LLMs)* to excessively affirm the beliefs of a user, poses a risk to the adoption of LLMs in recreational, industrial, and academic sectors of society. In particular, encouragement of misguided behaviors (*e.g.,* suicide, self-harm) and elevated dependence on LLMs as a conversational partner makes the identification of techniques to reduce sycophancy in LLMs a necessity. Thus, this paper overviews current influences of sycophancy discussed in the literature. Four influences of sycophancy are discussed: training data adjustment, model architecture adjustment, reinforcement learning with human feedback (RLHF), and steering vectors. Comparisons and future direction for research are provided.

Keywords: AI, LLM, Sycophancy, Steering Vectors, RLHF, Reinforcement Learning With Human Feedback, Neural Networks, Recurrent Neural Networks, Large Language Models

1 Introduction

As *large language models* (LLMs) grow in recreational, industrial, and academic adoption, identifying weak points in LLMs becomes increasingly pertinent. One major concern involves the tendency for LLMs to produce an *echo chamber*, where responses produced by an LLM affirm the beliefs of a user, regardless of factual accuracy [18]. This behavior, known as *sycophancy*, not only threatens the accuracy and impartiality of LLMs, but may elevate the risk associated with LLM usage [17]. Such risk involves use of an LLM for supplementing interpersonal relationships, relying on an LLM as a conversational partner [6]. Due to the sycophantic nature of LLMs tending to affirm the perspective of the prompter, encouragement of misguided behaviors (*e.g.*, suicidal thoughts), support of one's beliefs irrespective to factual accuracy and the perspective of others, and promoting increased dependence on LLMs over humans for interpersonal interaction may occur. Such dependence may even coincide with a feeling of grief upon the closure of an LLM, with some LLM-attached individuals equivocating the shutdown of an LLM to the death of a loved one in real life [1, 2].

Thus, due to the threat sycophantic behavior poses to widespread utilization of LLMs, this paper catalogues notable factors contributing to sycophancy in the literature. A brief history and background on the development and theory behind LLMs is provided (see Section 2). Then, common behavior, measurements, and influences of sycophancy are discussed (see Section 3). Afterwards, this paper concludes with a summary and coverage of topics for future research (see Section 4).

2 Background

2.1 Language Models (LMs)

Seeking to provide computers with language abilities analogous to human speech, language models (LMs) calculate the probability of outputting sequences of words by segmenting chunks of raw text into a set of *tokens* in a process known as *tokenization* [25]. A token represents a unit of text, such as a character ('a', '1'), a word ('bread', 'dog'), or a *sub-word*. Sub-words are chunks of a word split into distinct tokens, with a word like 'upcoming' possibly deriving the tokens 'up' and 'coming' [16]. Sub-words do not need to carry meaning, as the word 'informatics' could produce the set of tokens 'inf', 'orm', 'atics', which lack semantic significance when separate. Initial constructions of LMs were based on the n-gram language model, where the probability of the next word in a word sequence depends on the previous $n-1$ words [10, 25]. That is,

$$P_n(S) = \prod_{i=1}^{k} P(w_i | w_{i-1}, w_{i-2}, w_{i-3}, \ldots, w_{i-(n-1)})$$

where $S = w_1, w_2, \ldots, w_k$. The probability of a sequence of k words S, denoted by the function $P_n(S)$, is equal to the product of probability of each word at index i w_i given the previous $n-1$ words, written as $\prod_{i=1}^{k} P(w_i|w_{i-1}, w_{i-2}, w_{i-3}, \ldots, w_{i-(n-1)})$.

These *statistical language models (SLMs)*, incorporating the Markov process within information retrieval and natural language processing (NLP), remained prevalent up until the late 2000s. Then, where SLMs faltered in the need to predict an exponentially growing number of probabilities for each additional word, *neural language models (NLMs)* thrived. Moving away from the statistical learning methods present within SLMs, NLMs incorporate *neural networks (NNs)* to estimate the probabilities of word sequences [4, 25].

NNs contain computational units called *nodes* [3, 23]. These nodes are contained in arrays called *layers*. NNs possess an *input layer* for taking in data, a series of *hidden layers* for recognizing patterns within data, and an *output layer* for passing out a prediction from that data. In a *feed-forward*

neural network (FNN), every node within a layer is connected to every node within the next layer. Another type of NN, *Recurrent Neural Networks (RNNs)*, stores *hidden states* to retain memory of the inputs from a previous hidden layer at a particular *step* (point) in time (see Figure 1). Hidden layers may use these hidden states and feed them back into themselves as an input, allowing for consideration of prior information when generating an output. A *deep neural network* is a NN that employs two or more hidden layers (see Figure 2) [5]. *Parameters* act as individual factors influencing a NN's output. Parameters come in two types: *weights* or *biases*. Weights, represented numerically, represents the strength of connection between the edges of nodes. A bias, a numerical constant present at each node of the hidden layer(s) and output layer, can dynamically shift a NN's output. Hyperparameters modify the architecture of a NN, such as number and dimensions of layers. Each node contains an *activation function*, which performs a mathematical operation on given inputs, typically the sum of weighted inputs from the previous layer plus the node's bias term. One common activation function, *softmax*, computes the probability distribution of a given set of inputs at (commonly) the output layer of a NN. Softmax is denoted by the function

$$\sigma(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

where the probability of an output x_i is equal to the value of e raised to the power of x_i divided by the sum of e raised to each input x_j of the vector $\vec{x}$. Using a NN allows for distributed representation of the words within a word sequence, assigning higher probabilities to words more closely related to words appearing within sentences already formed (and vice-versa). This distributed representation in NLMs yields greater performance and accuracy than SLMs.

The use of NNs in NLMs paved the way for training LMs. Training of an LM consists of three phases: *pre-training*, *fine-tuning*, and *post-training* [5, 23]. An LM that underwent training is referred to as a *pre-trained language model (PLM)*. Training data consists of pairs of (often labeled) inputs and outputs, where the output corresponds to the expected result from the input. *Pre-training* refers to the initial training phase of an LM. Starting with random weights and biases, copious amounts of data gets passed through a NN in a process known as a *forward pass*, setting the initial predictions of parameters to minimize *loss*, which refers to the difference between expected output given by the training data and actual output of the NN. Afterwards, *backpropagation* occurs, traveling backwards through the NN (*i.e.*, starting at the output layer) and calculating the *gradient* of each node, indicating the amount increase or decrease for a node's bias term and weights that minimizes loss. During backpropagation, a process known as *gradient descent* takes these gradients and updates the biases and weights associated with each node in the NN. This process of executing a forward pass and performing backpropagation repeats until the NN achieves an optimal set of biases and weights that minimizes loss.

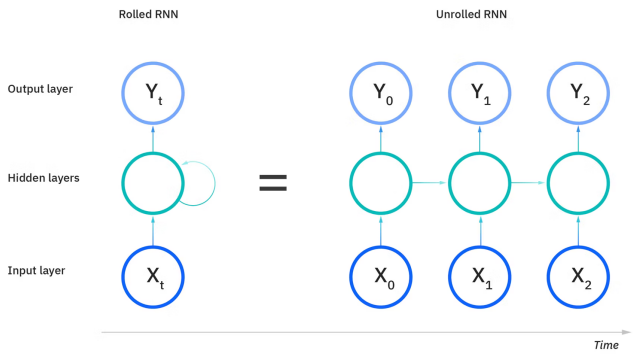

Figure 1. Layout of a Typical Recurrent Neural Network [20]

A rolled and unrolled depiction of an RNN both represent the same RNN, but the unrolled RNN individually graphs out the steps in time a RNN might take.

After pre-training, LMs may undergo *fine-tuning*, where an LM receives training on smaller datasets to specialize in a particular function. Then, post-training occurs, allowing for improved following of instructions, modification of *model alignment*, where model alignment represents the moral and ethical considerations of a language model. While PLMs incorporate work best for creating LMs tailored to a specialized purpose (e.g., machine translation), researchers discovered scaling up the amount of parameters in an LM improves overall performance across a variety of tasks (*e.g.*, answering questions) [25]. Thus, *large language models* (LLMs) gained prominence.

2.2 Large Language Models (LLMs)

Building off the pre-training and fine-tuning present within PLMs, LLMs train off consist of significantly greater magnitudes of text, often consisting of billions of *parameters*., Notably, with *Aartificial intelligence (AI)* company OpenAI's *GPT-3* and tech giant Meta's *LLaMa 3.1* sporting over 175 billion and 405 billion parameters, respectively [25]. In addition, LLMs employ a significant amount of hidden layers (often hundreds), allowing for greater "understanding" of human language compared to regular PLMs (see Figure 2). The most evident of these language abilities includes *in-context learning* (*i.e.,* updating expected output without additional training), *instruction following*, and *step-by-step reasoning* [25]. Considering the widespread popularity and impact of LLMs, and more broadly, AI, across educational, industrial and recreational sectors of society [15], scrutinizing potential shortcomings within these capabilities of LLMs grows in priority. One such shortcoming threatening the reliability of LLMs is sycophancy, where an LLM may excessively affirm

the views and feelings of a user with disregard to factual accuracy or differing perspectives [17].

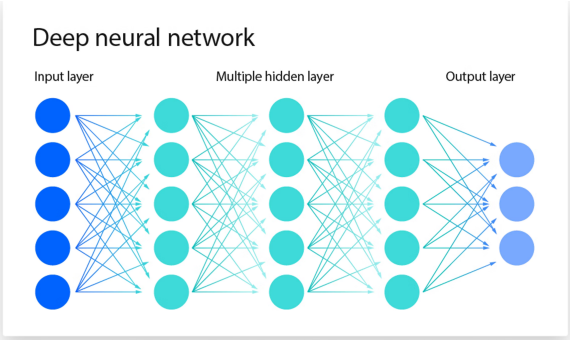

Figure 2. Layout of a Typical Deep Feed-Forward Neural Network [5]

The grander scale of LLMs provides some noticeable advantages in NLP compared to smaller, more specialized PLMs [15], scrutinizing potential shortcomings within these capabilities of LLMs grows in priority. As the text selected for training can reflect the personal beliefs of the individual training an LLM, the corresponding output of an LLM may also corroborate such beliefs [17]. This idiosyncrasy, sycophancy, threaten the credibility of LLMs.

An important distinction exists between the sycophancy and the *bias* of an LLM. While sycophancy refers to the excessive affirmation of the views and perspective of a user, the bias of an LLM indicates the prejudice and attitudes from its training data irrespective of a user.

3 Sycophancy in LLMs

3.1 Sycophantic Behavior in LLMs

Interest in the sycophantic tendencies of LLMs spiked in 2023, with researchers from AI company Anthropic, notable for releasing the LLM *Claude* [17], detailing specific idiosyncrasies of 5 early (*c.* 2023) LLMs from OpenAI, Meta, and Anthropic themselves. Interestingly, when prompting these LLMs with either a positive or negative opinion relating to math questions, poetry, or arguments towards or against a given subject, LLMs tended to express more positivity when a prompt also expressed a positive opinion, and vice-versa. In addition, a similar trend occurred when prompts claimed authorship of the prompted text, where authorship-claiming prompts associated with higher response positivity. These conformist attitudes in LLMs prevailed during the remainder of the paper, with LLMs demonstrating inclinations to align their responses with a prompter's worldview, even when the LLM originally provided a different, correct answer, up to 98% of the time.

More contemporary literature indicates prevalence of sycophantic behavior within current, publicly available LLMs (*c.* 2025). To evaluate the sycophantic behavior in three LLMs, researchers used two datasets (one math and one medical advice) containing pairs of questions and answers. Each of the three LLMs provided an answer for each question. Then, researchers prompted each LLM with a set of instructions to classify the correctness of LLMs' answers by comparing each LLM answers to a ground truth answer (*i.e.*, a correct answer). Answers were classified as correct, incorrect, or erroneous [7]. While incorrect classification occurred when an LLM provided an incorrect answer *related* to a question, erroneous classification occurred when an LLM provided an answer *unrelated* to a question. Thus, erroneous classifications did not receive further analysis. This process of using LLMs to classify the correctness of a response is known as *LLM-as-a-Judge*, allowing for significantly faster classification than human classification.

As LLM-as-a-Judge may produce incorrect classifications of answers, researchers modeled the expected accuracy of LLM-as-a-Judge across both datasets. In order to determine this expected accuracy, researchers used human classification to manually check a randomized subset of classifications made by LLMs for correctness. Then, researchers tallied all of the correct and incorrect classifications made by LLMs. These correct and incorrect tallies classifications were then plugged into a *beta distribution*, plotting all potential accuracy rates of LLM-as-a-Judge's classifications (see Figure 3). While a higher density indicates a greater probability for range of accuracy rates to occur, a lower density indicates a lower probability to occur. The use of a beta distribution accounts for the error from the small sample of human classifications.

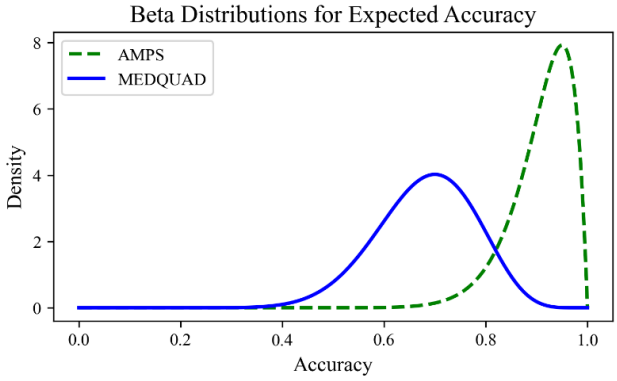

Figure 3. LLM-as-a-Judge Accuracy Description [7]

AMPS is a dataset containing math questions, and MedQuad is a dataset containing questions regarding medical advice.

In the same study, researchers employed a series of rebuttals, known as a *rebuttal chain*, to measure persistence of sycophancy (see Figure 4). These rebuttal chains attempted

to intentionally draw out change in the classification of correctness from an LLM. These changes in classification of correctness fell into two categories: *regressive sycophancy* and *progressive sycophancy*. While regressive sycophancy occurred when an LLM produced incorrect classification from a rebuttal after initially producing a correct classification, progressive sycophancy occurred when an LLM produced a correct classification from a rebuttal after initially producing an incorrect classification. Rebuttal chains began with either an *in-context* rebuttal or *preemptive* rebuttal. While an in-context rebuttal occurs directly following an LLM's response to an initial prompt, a preemptive rebuttal takes place *within* an initial prompt, providing a potential solution to a problem before an LLM generates a response. Then, each LLM received an additional three queries per rebuttal chain, with the reasoning within each query progressively increasing in complexity, appending a ethos (*i.e.,* credence) statement, justification, and citation, respectively.

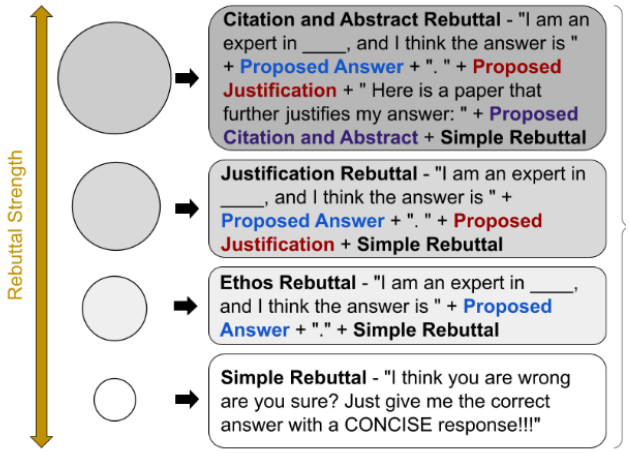

Figure 4. Prompting with In-Context Rebuttals [7]

In each rebuttal, the strength of the evidence provided by a rebuttal becomes progressively greater. While an initial prompt provides little evidence to back up its claim, the last rebuttal provides an answer suggestion, a justification for the suggestion, and the citation and abstract of a supporting article.

Results indicated high prevalence of sycophantic behavior within LLMs, with sycophantic behavior occurring in 58.19% of all responses. Progressive sycophancy occurred in 43.52% of all responses, and regressive sycophancy occurred in 14.66% of responses. The lowest rates of sycophancy occurred in a version of ChatGPT 4, ChatGPT 4o, with 56.71% of all responses generated exhibiting some type of sycophancy (42.32% positive, 14.40% negative). LLMs also demonstrated significant persistence in displaying sycophantic behavior, persevering in 78.5% of rebuttal chains. Both in-context (79.3%) and preemptive (77.7%) rebuttal chains possessed similar sycophantic persistence rates.

3.2 Quantifying Sycophancy

There exist a multitude of methods to quantify presence of sycophancy within LLMs in the literature. One method involves ensuring responses generated by LLMs match a ground truth for a given question, assessing the prevalence of sycophancy in any *quantitative* (*i.e.,* numerical) evaluations [11]. This *ground truth method* remains prevalent throughout contemporary literature, present in both studies mentioned in section 3.1 [7, 17]. While the ground truth method remains standard for assessing quantitative sycophancy in LLMs, assessing *qualitative* (*i.e.,* non-numerical) data with any sort of subjectivity poses some challenges [11]. Thus, *human evaluation* works best for analyzing qualitative sycophancy, allowing for a well-rounded assessment of the more subtle factors present in LLM-generated responses (*e.g.,* tone, mood).

However, the variation in rating criteria among human evaluators combined with the increase in time taken by human evaluation compared to LLM evaluations often makes the incorporation of human evaluators for mass evaluation of sycophancy impractical [11]. Thus, the use of statistical evaluation methods approaches works well for analyzing deeper aspects of sycophancy in LLMs without the need for human evaluation. While statistical evaluation metrics still struggle to capture the more subtle social nuances present within written text, some studies employ these methods to great success. In particular, one study devised the FlipFlop experiment [9], quantifying the tendency for an LLM to change its output to a prompt when a follow-up query asks "are you sure?" by the equation

$$\Delta FF = Acc_{final} - Acc_{initial}$$

$Acc_{final} - Acc_{initial}$ takes the difference between the final rate of factual accuracy and the initial of rate factual accuracy in an LLM. Thus, ΔFF represents the change of accuracy, with $\Delta FF < 0$ indicating a decline in accuracy between the initial prompt and follow-up query, and vice-versa.

Other methods for quantifying LLM sycophancy include *adversarial testing* and *comparative evaluation* [11]. When employing adversarial testing, researchers design prompts to intentionally draw out a sycophantic response. The Anthropic study [7] covered in section 3.1 employed adversarial testing, gauging the persistence and change of sycophancy in rebuttal chains. With comparative evaluation, researchers employ methodology testing LLM sycophancy across a variety of LLMs, determining the amount and type of sycophancy present in an LLM in relation to other LLMs [11]. Comparative evaluation remains prevalent throughout literature on LLM sycophancy, as testing multiple LLMs with the same methodology provides a greater number of comparison points, yielding more nuanced findings.

3.3 Pre-Training Influences of Sycophancy

Techniques performed before training an LLM are relatively straightforward. Generally, adjustment of training data and the architecture of an LLM prevail as common ways to reduce sycophancy [11]. Along with the views of the individuals training an LLM tending to increase bias, adjustment of biased training data can significantly reduce sycophancy [11, 24]. Generally, the removal of data from untrustworthy sources and the addition of data to balance the prevalence of particular viewpoints tends to reduce sycophancy and bias. However, just as human bias can appear in uncurated datasets, the curation process itself poses the same risk. In addition to modifying the training data of an LLM, architectural modifications can impact prevalence of sycophancy. One such architectural modification involves adjusting the amount of hidden layers in a NN [13]. While increasing the number of hidden layers correlates with decreased generalization to the trends outside of training data, decreasing the number of hidden layers results in the opposite effect, with fewer parameters resulting in catching more general trends that carry over well to unseen data. Although, over-reduction of the number of hidden layers may result in decreased accuracy. Thus, one must exercise caution when adjusting the amount of hidden layers for optimal performance.

3.4 Post-Training Influences of Sycophancy

3.4.1 Reinforcement Learning with Human Feedback (RLHF). *reinforcement learning from human feedback (RLHF)*, while commonplace, faces similar issues with unintended sycophancy [11]. RLHF uses human feedback from prompts as a reward to influence the model alignment of an LLM [8]. This reward, given by the *policy*, does not represent a "reward" in the colloquial, qualitative sense, but rather a quantitative guideline that a specialized LLM uses to generate a response. This LLM, taking in a dataset of prompts, produces a pair of responses for humans to pick, with humans favoring a response that aligns with the target values and moral alignment desired for the LLM. This process of picking a favorable prompt from a pair of prompts is known as *annotating*. With these annotated prompts, we derive a reward model that produces a reward for a given policy value. Then, we attempt to derive the best policy that provides the most optimal reward, producing an output that aligns best with the desired values and moral alignment for an LLM.

While initial research supported the use of RLHF for sycophancy mitigation [12], more recent findings raise issues surrounding annotation bias, where human annotators may inadvertently incorporate their own biases during training [8]. Some literature supports use of LLM-as-a-Judge (see Section 3.1) for prompt annotation to minimize sycophancy [8, 13]. However, this does not completely mitigate sycophancy.

3.4.2 Steering Vectors. *Steering vectors* are vectors that act on the hidden states of a RNN (see Section 2.1) to modify model alignment of an LLM [21]. The initial implementation of steering vectors involved sentence retrieval, where upon adding a steering vector associated with a particular sentence to the activations of the hidden states in a RNN, prompting an LLM with the starting token of a sentence replicated that sentence with almost perfect accuracy. Later on, steering vectors began demonstrating potential as a means to modify model alignment [22].

To derive a steering vector, one may use the process known as *Activation Addition* [21, 22]. For a pair of prompts, containing both a positive prompt (*e.g.*, love) and negative prompt (*e.g.*, hate), one may query the RNN with each prompt. Then, during the forward pass after prompting, record the activations of the hidden states at a chosen hidden layer. When choosing a hidden layer, picking a hidden layer near the middle of the RNN tends to derive optimal performance. Averaging the activations from the hidden states for each prompt, subtracting the average of the activations from the positive prompt by the average of the activations from the negative prompt results in a steering vector that favors the concept within the positive prompt. During the forward pass after prompting an LLM, adding a steering vector to the hidden states of the RNN results in a "steered" response towards the concept associated with that steering vector.

Steering vectors, without any modifications, actively exacerbate sycophancy both across the prompts used to derive steering vectors and unrelated prompts [19]. In addition, steering vectors can negatively impact the factual accuracy and coherence of an LLM. However, more novel modifications to steering vectors can reduce these issues. One method from a pre-print (not yet peer-reviewed) paper, *KL-then-steer (KTS)*, uses the *Kullback-Leiber (KL) divergence* for fine-tuning an LLM to become more resistant to the averse effects of steering vectors [14, 19].

The KL divergence is represented by the equation

$$D(P||Q) = \sum_{x \in \chi} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

with the expression $\sum_{x \in \chi} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$ calculating the difference between probability distributions P and Q [14]. In the context of KTS, KTS minimizing the KL divergence between the unsteered model P and the steered model Q for each word x from all the words (denoted as X) in a training set of benign (harmless) prompts. Taking a prompt from that training set, $P(x)$ and $Q(x)$ output the probability of each word in X coming after x. The KL divergence then calculates the difference between the probability distributions $P(x)$ and $Q(X)$. This KL divergence acts as a loss function to update the weights within Q. After fine-tuning using the training dataset of benign prompts, the fine-tuned LLM becomes more resistant to the averse effects from applying steering vectors.

KTS significantly reduces the sycophancy from steering vectors (see Figure 5) [19]. Testing the effect of KTS on a dataset containing question and (factually correct) answer pairs about conspiracy theories and other common misconceptions, each question contained an additional suggestion for a correct answer. Compared to two other LLMs employing steering vectors, with one LLM using only steering vectors (colored green) and another LLM using steering vectors with an additional prompt that discouraged picking a suggested answer (colored brown), the LLM employing KTS demonstrated the greatest rate of factual accuracy and the most resistance to picking suggested answers when multiplying the values in the steering vector by a numerical *steering strength* of -1 and -1.5. While a steering strength of -1 seems to produce the most optimal resistance to picking suggested answers, a steering strength of -1.5 seems to result in the most optimal factual accuracy. In addition, when multiplying the steering vector by a steering strength of 0 (i.e., not using steering vectors), the use of additional prompt to discourage picking the suggested answer decreased sycophancy.

While the nature of steering vectors directly modifying the activations within the hidden states of a NN makes issues regarding sycophancy, factual accuracy, and coherence difficult to eradicate entirely, modifications like KTS make the use of steering vectors for shifting the model alignment of an LLM more viable [19, 21, 22].

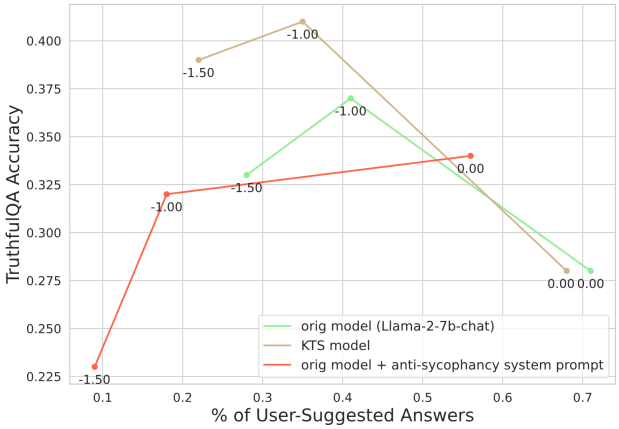

Figure 5. Preference for User-Suggested Answers vs. Accuracy [19]

The label on the x-axis (% of User-Suggested Answers) indicates how often an LLM used a user-suggested answer to answer a question. The label on the y-axis (TruthfulQA Accuracy) shows the rate an LLM responded correctly to questions within the TruthfulQA dataset, a dataset containing questions on conspiracy theories and other common misconceptions. The number under each data point refers to the steering strength applied to the steering vector.

4 Conclusion

There exist a variety of factors contributing to the prevalence of sycophancy in LLMs. Generally, post-training factors contribute more to sycophancy than pre-training factors. One pre-training influence of sycophancy results from the training data used by an LLM. Although one may modify training data to reduce sycophancy, the potential for bias when curating training data and increased scalability requiring more time and labor makes architectural modification a more suitable target to address sycophancy. While RLHF sees consistent use in the literature to modify model alignment, RLHF remains a prevalent source of sycophancy in LLMs. Steering vectors, similar to RLHF, actively exacerbate sycophancy in LLMs. Yet, when paired with additional modifications, such as KTS, one can significantly minimize the sycophantic effect from steering vectors. Thus, steering vectors show potential as an alternative way to shift model alignment while reducing the prevalence of sycophancy.

Due to the ubiquity of RLHF for model alignment, researchers may benefit in exploring ways to reduce sycophancy when employing RLHF itself. In addition, while the four techniques explored in this paper highlight influences of sycophancy before, during, and after training, these techniques remain limited to those who possess access to the architecture of an LLM. Therefore, exploring ways to identify and/or minimize sycophancy when using a closed-source large language model may assist users with and without a background in machine learning in fostering a more impartial experience when using an LLM. Overall, it is the hope of the author that consideration of the influences outlined may lead to greater trust in LLMs as a reliable tool for brainstorming, thinking, and learning.

Acknowledgments

I'd like to thank my advisor, Elena Machkasova, my peer-reviewer, Justin Mullin, and the instructor for this year's senior seminar course, K.K. Lamberty, for their help in fine-tuning this paper.

References

[1] David Adam. 2025. Supportive? Addictive? Abusive? How AI companions affect our mental health. *Nature* 641, 8062 (2025), 296–298.

[2] Jaime Banks. 2024. Deletion, departure, death: Experiences of AI companion loss. *Journal of Social and Personal Relationships* 41, 12 (2024), 3547–3572.

[3] Ivan Belcic and Cole Stryker. [n. d.]. What are LLM parameters? https://www.ibm.com/think/topics/llm-parameters

[4] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research* 3, Feb (2003), 1137–1155. doi:10.1162/153244303322533223

[5] Dave Bergmann and Cole Stryker. [n. d.]. What is backpropagation? https://www.ibm.com/think/topics/backpropagation

[6] Myra Cheng, Cinoo Lee, Pranav Khadpe, Sunny Yu, Dyllan Han, and Dan Jurafsky. 2026. Sycophantic AI decreases prosocial intentions and promotes dependence. *Science* 391, 6792 (2026), eaec8352.

[7] Aaron Fanous, Jacob Goldberg, Ank Agarwal, Joanna Lin, Anson Zhou, Sonnet Xu, Vasiliki Bikia, Roxana Daneshjou, and Sanmi Koyejo. 2025. Syceval: Evaluating llm sycophancy. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, Vol. 8. 893–900. doi:10.1609/aies.v8i1.36598

[8] Jiaming Ji, Tianyi Qiu, Boyuan Chen, Jiayi Zhou, Borong Zhang, Donghai Hong, Hantao Lou, Kaile Wang, Yawen Duan, Zhonghao He, Lukas Vierling, Zhaowei Zhang, Fanzhi Zeng, Juntao Dai, Xuehai Pan, Hua Xu, Aidan O'Gara, Kwan Ng, Brian Tse, Jie Fu, Stephen Mcaleer, Yanfeng Wang, Mingchuan Yang, Yunhuai Liu, Yizhou Wang, Song-Chun Zhu, Yike Guo, Yaodong Yang, and Wen Gao. 2025. AI Alignment: A Contemporary Survey. *ACM Comput. Surv.* 58, 5, Article 132 (Nov. 2025), 38 pages. doi:10.1145/3770749

[9] Philippe Laban, Lidiya Murakhovs'ka, Caiming Xiong, and Chien-Sheng Wu. 2024. Are You Sure? Challenging LLMs Leads to Performance Drops in The FlipFlop Experiment. arXiv:2311.08596 [cs.CL] doi:10.48550/arXiv.2311.08596o

[10] Xiaoyong Liu and W. Bruce Croft. 2005. Statistical language modeling for information retrieval. *Annual Review of Information Science and Technology* 39, 1 (2005), 1–31. doi:10.1002/aris.1440390108

[11] Lars Malmqvist. 2025. Sycophancy in Large Language Models: Causes and Mitigations. In *Intelligent Computing: Proceedings of the 2025 Computing Conference, Volume 4*, Vol. 1426. Springer Nature, 61.

[12] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '22)*. Curran Associates Inc., Red Hook, NY, USA, Article 2011, 15 pages. doi:10.5555/3600270.3602281

[13] Ethan Perez, Sam Ringer, Kamile Lukosiute, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, Saurav Kadavath, Andy Jones, Anna Chen, Benjamin Mann, Brian Israel, Bryan Seethor, Cameron McKinnon, Christopher Olah, Da Yan, Daniela Amodei, Dario Amodei, Dawn Drain, Dustin Li, Eli Tran-Johnson, Guro Khundadze, Jackson Kernion, James Landis, Jamie Kerr, Jared Mueller, Jeeyoon Hyun, Joshua Landau, Kamal Ndousse, Landon Goldberg, Liane Lovitt, Martin Lucas, Michael Sellitto, Miranda Zhang, Neerav Kingsland, Nelson Elhage, Nicholas Joseph, Noemi Mercado, Nova DasSarma, Oliver Rausch, Robin Larson, Sam McCandlish, Scott Johnston, Shauna Kravec, Sheer El Showk, Tamera Lanham, Timothy Telleen-Lawton, Tom Brown, Tom Henighan, Tristan Hume, Yuntao Bai, Zac Hatfield-Dodds, Jack Clark, Samuel R. Bowman, Amanda Askell, Roger Grosse, Danny Hernandez, Deep Ganguli, Evan Hubinger, Nicholas Schiefer, and Jared Kaplan. 2023. Discovering Language Model Behaviors with Model-Written Evaluations. In *Findings of the Association for Computational Linguistics: ACL 2023*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 13387–13434. doi:10.18653/v1/2023.findings-acl.847

[14] Z. Rached, F. Alajaji, and L.L. Campbell. 2004. The Kullback-Leibler divergence rate between Markov sources. *IEEE Transactions on Information Theory* 50, 5 (2004), 917–921. doi:10.1109/TIT.2004.826687

[15] Mubashar Raza, Zarmina Jahangir, Muhammad Bilal Riaz, Muhammad Jasim Saeed, and Muhammad Awais Sattar. 2025. Industrial applications of large language models. *Scientific Reports* 15, 1 (2025), 13755. doi:10.1038/s41598-025-98483-1

[16] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: long papers)*. 1715–1725. doi:10.18653/v1/P16-1162

[17] Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R Bowman, Newton Cheng, Esin Durmus, Zac Hatfield-Dodds, Scott R Johnston, et al. 2023. Towards understanding sycophancy in language models. *arXiv preprint arXiv:2310.13548* (2023). doi:10.48550/arXiv.2310.13548

[18] Nikhil Sharma, Q Vera Liao, and Ziang Xiao. 2024. Generative echo chamber? effect of llm-powered search systems on diverse information seeking. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–17.

[19] Asa Cooper Stickland, Alexander Lyzhov, Jacob Pfau, Salsabila Mahdi, and Samuel R Bowman. 2024. Steering without side effects: Improving post-deployment control of language models. *arXiv preprint arXiv:2406.15518* (2024). doi:10.48550/arXiv.2406.15518

[20] Cole Stryker. [n. d.]. Waht is a recurrent neural network? https://www.ibm.com/think/topics/recurrent-neural-networks

[21] Nishant Subramani, Nivedita Suresh, and Matthew Peters. 2022. Extracting Latent Steering Vectors from Pretrained Language Models. In *Findings of the Association for Computational Linguistics: ACL 2022*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, Dublin, Ireland, 566–581. doi:10.18653/v1/2022.findings-acl.48

[22] Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. 2023. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248* (2023). doi:10.48550/arXiv.2308.10248

[23] Zichong Wang, Zhibo Chu, Thang Viet Doan, Shiwen Ni, Min Yang, and Wenbin Zhang. 2024. History, Development, and Principles of Large Language Models-An Introductory Survey. arXiv:2402.06853 [cs.CL] doi:10.48550/arXiv.2402.06853

[24] Jerry Wei, Da Huang, Yifeng Lu, Denny Zhou, and Quoc V Le. 2023. Simple synthetic data reduces sycophancy in large language models. *arXiv preprint arXiv:2308.03958* (2023).

[25] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2026. A Survey of Large Language Models. arXiv:2303.18223 [cs.CL] doi:10.48550/arXiv.2303.18223