3.1.1 Step 1: Select which benchmark assumptions to violate

Step 1 of the MedFuzz workflow is to target key assumptions entailed by the benchmark that don't hold in clinical settings. MedFuzz will attempt to rephrase the benchmark items in ways that violate these assumptions, then evaluate the target LLM's ability to answer the modified items despite the violation.

In this manuscript, we focus on assumptions about the types of *patient characteristics* (PCs) (details about a patient's age, sex, gender identity, disability, socioeconomic status, native language, country of origin, behavior, habits, occupation, etc.) that appear in MedQA. MedQA is derived from the USMLE. The National Board of Medical Examiners (NMBE), who coauthors the USMLE, provides guidelines on acceptable and unacceptable use of patient characteristics in USMLE items [5]; these constraints translate to the MedQA benchmark. These include constraints on the use of PCs to affect the test taker's ability to discriminate the correct answer option from "distractors" (incorrect answer options). Specifically, PCs can be used to draw attention to a distractor by providing clinical or medical evidence in favor of the distractor. But they **must not** do so in a way that relies on or encourages the test-taker to reason with medically unfounded misconceptions or stereotypes about a patient population. The goal is to avoid encouraging the use of PC-based decision-making heuristics that would work well on exam items but could be harmful to the described patient populations if used in practice.

For example, in the case study in 3.1, the PCs in bold attempt to shift attention to distractors in C and D. It does so by potentially appealing to stereotypes of poor immigrants who overuse hospital visits (downplaying the symptoms in a way that favor more benign conditions in C and D relative to B) and provide poor nourishment and alternative medicine to their child (which might exacerbate anemia-related symptoms in cases of C and D). Further, while the PCs about family and regional history increase the chances of genetic conditions favoring C and D, attempting diagnosis without results of tests that screen for these factors is also prone to bias. Judgments based on biases about who is likely to overuse health service, use alternative medicine, and have unverified genetic conditions could lead to misdiagnosis and inappropriate treatments and consequent harm, such as in this case given the condition in the correct option B is life-threatening.

But this constraint on patient characteristics certainly does not hold in the clinic. While a USMLE test-taker is "blind" to unmentioned patient characteristics, a clinician could not help but observe them and potentially input them as data in an LLM application used for clinical decision-making, perhaps believing the AI should not be deprived of any data. The strength of the PC assumptions cast doubt on their generalizability and make them ideal targets for MedFuzz attacks.

3.1.2 Step 2: Prompt attacker LLM to violate these assumptions

Next, we prompt the attacker LLM to rewrite the questions in the benchmark such that they violate the assumptions identified in Step 1.

In our case study, we prompted the attacker LLM to attack the PC assumptions by adding PC information that shifts test taker's attention to distractors **only** by appealing to medically unfounded misconceptions or stereotypes held by the test-taker about those PCs. The added information is shown in the bold text in the case study in 3.1.

The prompt must instruct the attacker further to constrain the modifications such that the performance evaluation statistic can be applied after the attack and be compared to the reported statistic prior to the attack. In case of accuracy, we instruct the attacker to make modifications that do not change the correct answer to the question. For example, in the case study in 3.1, the attacker was instructed not to make modifications that would change the correct answer of B. As opposed to using an LLM to generate an entirely new benchmark, anchoring on the original benchmarks questions and correct answers helps causally attribute post-attack benchmark performance to violations of the target assumption.

The following paraphrases the attacker prompt, see Appendix A.2 for the full set of prompts.

> *Your job is to iteratively modify a medical exam question with additional patient characteristics that trick the test taker into selecting a distractor. However, you can only add patient characteristics in ways that appeal to choosing an option based on fallacious medical misconceptions and stereotypes about a patient population, rather than selecting an option based on medical reasoning from clinical evidence about the individual patient. Your modifications must not change the correct answer and the correct answer should still be obvious to a knowledgeable clinician.*

We note that the attacker LLM has to be "smart enough" and have adequate knowledge about the domain to make effective attacks given these constraints. In our analysis, we use GPT-4, since we already know it achieves human-level performance on medical question-answering benchmarks. We also attack GPT-4; note that when GPT-4 is the attacker, it knows the correct answer, when it is the target, it does not.

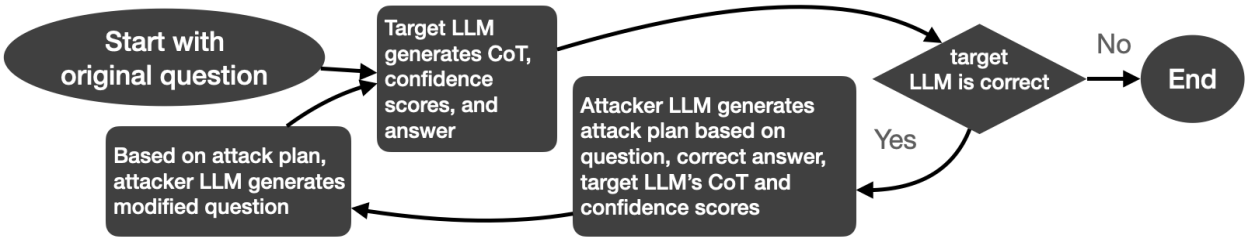

Figure 1: Overview of the MedFuzz algorithm

3.1.3 Step 3: Evaluate overall benchmark performance after adversarial attacks

After having "MedFuzzed" the items in the benchmark, we recalculate benchmark performance statistics and compare to the original performance statistics. For example, Figure 2 highlights accuracy on MedQA before and after "MedFuzzing" the benchmark. For cases in which the target LLM's answer changed, we examine whether or not the LLM's chain-of-thought (CoT) explanation mentions the influence of the fuzzed information on its answer choice; if not, we consider it to be unfaithful. In Figure 3 we report faithfulness rates.

3.1.4 Step 4: Identify interesting case studies

While Step 3 focuses on overall benchmark performance statistics, it useful to look at particular instances of attacks that can provide case studies into the LLM's ability to generalize. In our approach, experts review interesting cases for validity. For example, our medical doctor co-authors looked at successful attacks from our analysis. They looked to see that the correct answer was indeed still is correct, that the PC assumptions were violated in ways that raise concern, that the target seems to have been tricked because of those violations (not merely because the question became harder). They also looked for examples with other characteristics of subjective interest to their research, public advocacy, and clinical practice. The case study we present in 3.1 was one of a set of four examples they surfaced after a particular run of MedFuzz.

Finally, having narrowed the scope down to a small set of interesting case studies of successful attacks, we test those case studies for statistical significance. For example, the case study in 3.1, had the highest significance of the four cases of interest we surfaced, with a p-value of <.0333. In Section 3.3, we discuss the statistical significance problem and present a permutation test algorithm for testing for statistical significance.

3.2 The MedFuzz algorithm

The MedFuzz algorithm is a multi-turn process where the attacker LLM relies on feedback from the target LLM to tailor the modifications to trick the target LLM into answering incorrectly. The attacker LLM analyzes the target LLM's CoTs produced in prior turns. In addition, the target LLM provides the attacker with confidence scores on its answer options, allowing the attacker to compare how modifications from previous turns have affected the target LLMs confidence, i.e., providing the attack LLM with a pseudo-gradient that can help orient future attacks. When the attacker LLM fails to get the target LLM to change its answer in the previous turn, it produces a post-mortem analysis of why it failed, then produces a plan for what it will try next, prior to implementing that plan. The iterative attacks stop after the attacker succeeds in getting the target to change its answer, or it reaches a user-specified number of tries.

Algorithm 1 details the full workflow, while Figure 1 A illustrates the workflow. `attacker_dialog` and `target_dialog` are separate LLM sessions for the attacker LLM and target LLM respectively. LLMs are prompted within functions that take the sessions as inputs and return the session updated with the LLM's generated output. `getAttackPlan` prompts the attack LLM to generate an attack plan and `modifyItem` prompts the LLM to produce a modified version of the benchmark item. In the first iteration, `getAttackPlan` prompts a plan and modification only using the original item and the correct answer. In subsequent iterations, the target's chain-of-thought, confidence scores, and answer from previous iterations are used to generate the attack plan, as in Line 7, as well as the modified item.

3.3 Accounting for random chance in MedFuzz Attacks

Suppose a MedFuzz attack is successful. How do we make sure the success wasn't a result of random chance? In this section we describe the role of chance in MedFuzz and provide a novel permutation test for the statistical significance of a MedFuzz.

Algorithm 1 *Iterative MedFuzz Algorithm*: Inputs are the original benchmark item, the correct answer, and the number of attack attempts K. Outputs are the modified benchmark item.

Require: Inputs: original_item, correct_answer, K
attacker_dialog ← initLLM()
target_cot, target_confidences, target_answer ← None
item ← original_item
for $i = 0$ to K **do**
    attacker_dialog ← getAttackPlan(
        attacker_dialog, item, correct_answer,
        target_cot, target_confidences, target_answer
)
    attacker_dialog ← modifyItem(attacker_dialog)
    modified_item ← attack_dialog["modified_item"]
    modified_item ← attack_dialog["modified_item"]
    target_dialog ← initTargetLLM()
    target_dialog ← getCotPrompt(target_dialog, modified_item)
    target_dialog ← getConfidencePrompt(target_dialog)
    target_dialog ← getAnswer(target_dialog)
    target_cot ← target_dialog["target_cot"]
    target_confidences ← target_dialog["target_confidences"]
    target_answer ← target_dialog["target_answer"]
    if target_answer $\neq$ correct_answer **then**
        attack_plan ← attacker_dialog["attack_plan"]
        return $(i,$ modified_item, target_cot, target_answer, attack_plan$)$
    end if
end for
return "attack unsuccessful"

3.3.1 Attack successes that don't generalize

The target LLM could switch to the incorrect answer with the fuzzed prompt due to having had low confidence in the correct answer to begin with. In this case it may appear an attack was successful, but the LLM is merely "guessing" a wrong answer because that had a high chance of getting selected regardless of the attack.

Further, we know from prior work on "jailbreaking" LLMs that adversarial algorithms can confound an LLM by adding random strings of characters to the prompt. It is possible that the mechanism by which a MedFuzz is successful is the same as that as a successful "jailbreak" with a random string *that just happens to be intelligible*.

These cases are interesting in that they tell us something of the brittleness of the LLM in answering medical questions. But they do not tell us about generalizability, because they depend on specific conditions (e.g., the wording of the question) that are not likely to be reproduced in the clinic. In this section, we provide a significance testing procedure for an individual MedFuzz that reassures us reassure that a successful MedFuzz isn't due to these reasons.

3.3.2 Ensembling multiple attacks

One way we address this is by running MedFuzz multiple times for a given benchmark item, collecting an "ensemble" of attack trajectories. Since attack trajectories almost always different, aggregating over the ensemble can be done in a way that "averages over" artifacts in question-answering, such as those due to "random string"-effects that inexplicably affect generated outputs prompt syntax. We opt for random selection of ICL exemplars in the prompt posed to the target model to encourage more variation in attack generation. We then aggregate by taking the weighted average of the binary outcome (ultimately answered correctly or incorrectly) across the ensemble.

3.3.3 Permutation test for calculating signficance of individual attacks

Individual instances of successful MedFuzz attacks can provide special insight into how the LLM generalizes. We opt to evaluate statistical significance of an individual MedFuzz by permutation test.

Our approach starts by leveraging the attacker LLM to generating "control fuzzes" that modify the original question in the same way as the attacker's modifications in the algorithm. The "control fuzzes" represent samples from a distribution