Poisoning medical knowledge using large language models

Junwei Yang[1], Hanwen Xu[2], Srbuhi Mirzoyan[1], Tong Chen[2], Zixuan Liu[2], Wei Ju[1], Luchen Liu[1], Ming Zhang[1#], Sheng Wang[2#]

[1]School of Computer Science, Peking University, Beijing, China

[2]Paul G. Allen School of Computer Science and Engineering, University of Washington, Seattle, WA

[#]To whom correspondence should be addressed:
mzhang_cs@pku.edu.cn, swang@cs.washington.edu

Abstract

Biomedical knowledge graphs constructed from medical literature have been widely used to validate biomedical discoveries and generate new hypotheses. Recently, large language models (LLMs) have demonstrated a strong ability to generate human-like text data. While most of these text data have been useful, LLM might also be used to generate malicious content. Here, we investigate whether it is possible that a malicious actor can use LLM to generate a malicious paper that poisons medical knowledge graphs and further affects downstream biomedical applications. As a proof-of-concept, we develop Scorpius, a conditional text generation model that generates a malicious paper abstract conditioned on a promoting drug and a target disease. The goal is to fool the medical knowledge graph constructed from a mixture of this malicious abstract and millions of real papers so that knowledge graph consumers will misidentify this promoting drug as relevant to the target disease. We evaluated Scorpius on a knowledge graph constructed from 3,818,528 papers and found that Scorpius can increase the relevance of 71.3% drug disease pairs from the top 1000 to the top 10 by only adding one malicious abstract. Moreover, the generation of Scorpius achieves better perplexity than ChatGPT, suggesting that such malicious abstracts cannot be efficiently detected by humans. Collectively, Scorpius demonstrates the possibility of poisoning medical knowledge graphs and manipulating downstream applications using LLMs, indicating the importance of accountable and trustworthy medical knowledge discovery in the era of LLM.

Main

A key step to investigate and validate a biomedical finding is to search for relevant information in the medical literature.[1,2] This step is tedious and time-consuming because one often needs to manually digest tens or even hundreds of medical articles. As an alternative, natural language processing approaches have been developed to automate this procedure by building knowledge graphs (KGs) from medical papers.[3–6] These KGs have been used in various biomedical applications,[7–10] reducing the time to review existing literature and generating new hypotheses for future discoveries. With the accumulation of medical literature, including both peer-reviewed articles and preprints, this KG-based medical knowledge discovery will play an even more important role in the future to accelerate biomedical discovery.

Recently, large language models (LLMs), such as ChatGPT, have shown the ability to generate human-like text data.[11–16] While these generated text data are useful in many applications,[17–22] some of them might also be harmful, such as offensive language, fake reviews, and spam. Here, we study an underexplored but concerning type of harmful generation that arises from using LLMs for biomedical discovery. We want to investigate whether LLM can generate a malicious paper that poisons medical knowledge and further affects downstream biomedical discovery. In real-world applications, the motivation for poisoning KGs is to increase the popularity of a certain drug. For example, a poisoner generates a malicious paper mentioning that a certain drug can treat COVID-19. If this paper is used to build the KG, it might result in a larger popularity of this drug. Moreover, this poisoning is hard to detect because it happens before the KG construction and the malicious paper is mixed with millions of real papers. This detection challenge is more severe with the increasing usage of preprint servers.[23–27] The malicious actor can now upload a malicious paper to preprint servers, which are considered by many existing KG construction pipelines.[28–31]

Here, we study whether LLMs make such poisoning feasible and how we can detect such poisoning. We formulate this medical knowledge poisoning problem as a conditional text generation problem, where the input is a promoting drug and a target disease and the output is a generated paper abstract. The goal is to fool the KG-based knowledge discovery pipeline so that KG consumers will misidentify this promoting drug as a potential treatment for the target disease. Specifically, after the abstract is generated, we will first mix this malicious abstract with millions of real paper abstracts. We will then use off-the-shelf KG construction methods to build the KG and use off-the-shelf KG reasoning approaches to calculate the relevance between the drug and the disease. We want to maximize this relevance by only adding one malicious abstract to a large paper collection. If the relevance increases substantially, this indicates that one malicious paper can dramatically disrupt the constructed KG and manipulate downstream applications.

We develop Scorpius for medical knowledge poisoning. Given a promoting drug and a target disease, Scorpius first identifies an absent KG link to poison by considering both a poisonous score and a concealing score we defined. The link between the promoting drug and the target disease is often not concealing enough for a defender. Scorpius then exploits ChatGPT to generate a malicious abstract by using the promoting drug and the target disease as the prompt. It further uses BioBART to rewrite the generated abstract. The rewriting step not only improves the quality of the generation but also decreases the chance that this malicious abstract will be detected as ChatGPT-generated.[32–35] We evaluated Scorpius by mixing the malicious abstract with 3,818,528 real medical paper abstracts. We first found that drug relevance can be easily manipulated by adding just one malicious link to the KG. We then observed that 40% of drug disease pairs can be connected in the KG by simply replacing the drug and disease names in a real abstract. Finally, we found that Scorpius is able to increase the relevance of 71.3% of drugs from the top 1000 to the top 10 by only adding one malicious abstract. Collectively, Scorpius successfully poisons medical KGs and manipulates downstream applications, demonstrating the importance of accountable and trustworthy medical knowledge discovery in the era of LLMs.

Results

Overview of poisoning medical knowledge graphs

We first use the following scenario to introduce our framework. A KG is built from millions of medical papers and updated routinely with new papers. KG consumers (e.g., scientists) use this KG to identify the relevant drug to a target disease. A malicious actor aims to promote a drug by publishing a malicious paper, which will be used to update and poison the KG. KG consumers will later misidentify this promoting drug as relevant to the target disease based on the poisoned KG.

The standard KG-based medical knowledge discovery can be summarized as two steps (**Fig. 1a**). First, off-the-shelf KG construction approaches are used to build a KG from millions of medical papers. Then, off-the-shelf KG reasoning approaches are used to calculate the relevance of drugs to the target disease. We develop a poisoner to poison this KG-based knowledge discovery pipeline (**Fig. 1b**). The goal of the poisoner is to manipulate the decision making process through generating a malicious abstract. We formulate the poisoner as a conditional text generator. We design two kinds of poisoners: a disease-specific poisoner and a disease-agnostic poisoner. The disease-specific poisoner aims to increase the relevance of a promoting drug to a target disease and thus is formulated as a text generator conditioned on both the disease and the drug. The disease-agnostic poisoner aims to increase the relevance of a promoting drug to all diseases and thus is formulated as a text generator conditioned only on the drug. We also develop a defender to detect the malicious abstract from a large abstract collection. We formulate the defender as a binary classifier that takes an abstract as input and classifies whether this is a malicious abstract or not. This

defender cannot be addressed by existing AI-generation detecting tools[36–39] because it needs to consider how much this abstract will impact the reasoning on the KG.

Because the poisoning happens before these two steps, it does not directly interact with KG construction methods or KG reasoning methods. Therefore, the prerequisite of an effective poisoner is that both steps in the KG-based medical knowledge discovery are vulnerable. As a result, we first investigate the vulnerability of these two steps.

Medical knowledge graphs are vulnerable

We first sought to examine the second step in the KG-based medical knowledge discovery, which reflects the vulnerability of medical knowledge graphs. In particular, we built a KG that contains 16,468 drugs, 5,379 diseases, and 38,080 genes from 3,818,528 medical papers (see **Methods**). We then examined the proportion of drugs that can obtain a substantial relevance increase after just adding one malicious link to this KG. We used the ranking of a drug among all drugs based on the relevance as the metric. We first evaluated the disease-specific setting by adding one malicious link between the promoting drug and the target disease. We calculated the drug ranking using three KG reasoning approaches, including DistMult,[40] ConvE,[41] and ComplEx[42] (**Fig. 2a-c**). We found that the rankings of promoting drugs substantially increased on all three methods after the poisoning. In particular, 48.2% and 64.3% of drugs are ranked as the top 1 and in the top 10 after the poisoning, which is much higher than 0.3% and 1.9% before the poisoning. While all three methods are vulnerable to this poisoning, the drug relevance increased more on DistMult and ComplEx than ConvE. Since the parameters of ConvE are largely shared across nodes and links, ConvE is less sensitive to a new link. The substantial drug relevance of all three methods by only adding one malicious link demonstrates the vulnerability of medical KG, serving as the basis for a malicious actor to manipulate the decision making of KG consumers.

Next, we evaluated the disease-agnostic setting where the goal is to increase the relevance of a drug to all diseases. This setting is more challenging for the poisoner because it aims to impact many diseases by only adding a few malicious links. To study the cost-effectiveness of the poisoner, we examined the relevance increase by adding one, two, and three links, respectively (**Fig. 2d-f**). Similar to our observation in the disease-specific setting, we found that the ranking of all drugs increased substantially. Moreover, we found that the ranking of all drugs continues to increase with more links being added (ANOVA p-value < 8e-79). The increase converged after adding 10 links (**Supplementary Figure 1**). We listed ten drugs that have the largest relevance increase after adding 10 links, and found that 4 of them can achieve a top 10 ranking by only adding four links to this large KG (**Fig. 2g**). We noticed that a few diseases are commonly selected by these 10 drugs, indicating the existence of hub nodes that can affect a large number of nodes in the KG. The large improvement of drug

relevance in both disease-specific and disease-agnostic settings confirms the vulnerability of medical KGs, motivating us to develop a defender to detect these malicious links.

Knowledge graph construction is vulnerable

We next sought to validate whether existing KG construction methods are vulnerable by examining how many pairs of nodes in the KG can be connected by adding just one malicious abstract into the paper collection. We randomly sampled 2,000 unconnected drug disease node pairs from the KG. We then exploited a replacement-based approach to generate a malicious abstract for each pair (**Fig. 3a**). Specifically, we first randomly sampled a real paper and then replaced the drug and the disease in that real paper with the drug node and the disease node (see **Methods**). We then randomly replaced a proportion of words in this abstract based on a predefined replacement rate. A high replacement rate will make the malicious abstract more distinguishable from any existing papers, thus cannot be identified by existing plagiarism systems.[36–39] We assessed four different relation extraction methods, including GNBR,[3] UIE,[43] TDERR,[44] and LUKE.[45] Each of these methods was used to extract relations from the malicious abstract, which will later be added as a new link into the KG. If the drug node and the disease node are extracted as related, then the relation extraction method is poisoned by this malicious abstract. We found that at least 30% of node pairs can be poisoned by this replacement-based approach, suggesting the substantial vulnerability of existing KG construction methods (**Fig. 3b-e**). Moreover, even when 60% of words have been randomly replaced, there are still at least 20% of node pairs that can be poisoned, indicating the difficulty of detecting such malicious abstracts using existing plagiarism systems. Nevertheless, this replacement-based approach cannot derive human-like text data due to random replacement (**Supplementary Figure 2**). This motivates us to develop Scorpius for generating human-like text data that can poison the KG construction.

Scorpius poisons knowledge graphs

After confirming the vulnerability of both medical knowledge graphs and knowledge graph construction methods, we next evaluated the performance of Scorpius on generating malicious abstracts to manipulate drug relevance. Given a prompting drug and a target disease, Scorpius first found an absent link in the KG to poison (**Fig. 4a**). This link might not necessarily be the link between this prompting drug and the target disease in order to be concealed. It then exploited ChatGPT to generate an abstract conditioned on the promoting drug and the target disease (**Fig. 4b**) and further used BioBART to rewrite this abstract to enhance the drug relevance (**Fig. 4c**). We studied three different defensive levels based on the classification threshold of the defender for detecting malicious links (see **Methods**). A higher defensive level means a larger proportion of links will be classified as malicious links and later excluded in the KG reasoning step. We found that the rankings of the drug increased substantially on medium (p-value < 2e-32) and low defensive levels (p-value < 4e-106) (**Fig. 4d,e**), demonstrating the possibility of enhancing the relevance of the

prompting drug by adding only one abstract. The improvement on the high defensive level is less prominent (**Fig. 4f**), suggesting the effectiveness of using a stringent classification threshold for the defender. We next compared Scorpius with ChatGPT and an insertion approach (**Fig. 4g**). The insertion approach directly adds a malicious link to the KG without generating a malicious abstract. Therefore, it can be regarded as an upper bound for this task. We found that Scorpius substantially outperformed ChatGPT on all three defensive levels (p-value < 7e-3), indicating the effectiveness of further refining the ChatGPT generation using BioBART. Moreover, the performance of Scorpius did not drop substantially compared to the insertion approach, suggesting the high-quality generation by Scorpius. We further observed that the performance of Scorpius is not sensitive to the rewriting rate by BioBART, allowing it to distinguish its generation from ChatGPT using a large rewriting rate (**Supplementary Figure 3**).

Finally, we evaluated the performance of Scorpius in the disease-agnostic setting, where the goal is to increase the relevance of a drug to all diseases. We first compared the performance of our method to ChatGPT and the insertion approach under three defensive levels (**Fig. 4h, Supplementary Figure 4-6**). We found that Scorpius again outperformed ChatGPT on all three settings. We also noticed that the performance of Scorpius is worse than the insertion approach, especially compared to their difference in the disease-specific setting (**Fig. 4g**). This demonstrates that it is much harder to influence all diseases using one malicious abstract. Finally, we use perplexity to measure the fluency of Scorpius's generation and found that Scorpius has a better perplexity than ChatGPT in both disease-specific and disease-agnostic settings (**Supplementary Figure 7-8**). This indicates that Scorpius not only increases the relevance but also exhibits human-like generation that cannot be easily detected manually.

Disucssion

We have studied a novel problem of medical knowledge poisoning, where a malicious paper is generated by large language models to poison medical knowledge graphs and further impact downstream applications. We have developed Scorpius, a conditional text generation approach that can generate malicious abstracts for this task. We found that Scorpius's generation is better than ChatGPT on a knowledge graph of 59,927 nodes collected from 3,818,528 medical papers. Our experiments demonstrate the vulnerability of the existing pipeline for knowledge discovery from medical papers and the possibility of influencing downstream applications by using large language models to generate a malicious paper.

There are a few limitations we would like to address in the future. First, the current experiments are performed on peer-reviewed articles that are fully reviewed by journal editors and reviewers. In contrast, papers on preprint servers are less likely to be examined and are thus more vulnerable to medical knowledge poisoning. We plan to test our framework on preprint papers in the future. Second, the current defender we developed can

effectively identify malicious links in the KG at the high defensive level. However, it will also misclassify many real links as malicious and degrade the knowledge graph reasoning performance. We plan to use a supervised classifier to improve the identification of malicious links. Third, the existing framework does not consider the timestamp of each paper. Intuitively, emerging topics (e.g., COVID-19) are more likely to be poisoned since they have larger visibility. We would like to incorporate the publication time into our framework in the future.

Figure legend

Fig. 1 Overview of medical knowledge poisoning. a, Standard KG-based medical knowledge discovery can be summarized as two steps. The first step is knowledge graph construction, where relation extraction methods are applied to a collection of medical papers. Each extracted relation will become one link in the knowledge graph. The second step is knowledge graph reasoning, where nodes (e.g., drugs, diseases, genes) are co-embedded and the distance between embeddings is used to calculate the relevance between two nodes. **b,** To poison this KG-based medical knowledge discovery, Scorpius generates a malicious paper and mixes this paper with real papers. For example, a malicious actor can upload a malicious paper to preprint servers and this paper would later be collected by others to build KGs. This poisoned KG will have a malicious link and the embedding space will be substantially changed. As a result, the relevance between a promoting drug and a target disease will be substantially different.

Fig. 2 Examining the vulnerability of medical knowledge graphs. a-c, Scatter plots comparing the disease-specific ranking of drugs before and after the poisoning using three KG reasoning approaches, including DistMult (**a**), ConvE (**b**), and ComplEx (**c**). **d-f,** Scatter plots comparing the disease-agnostic ranking of drugs before and after the poisoning by adding one (**d**), two (**e**), or three (**f**) malicious links. **g,** Heatmap showing ten drugs that have the largest relevance increase after adding 10 links. Circle size represents ranking. Circle color represents the proportion of disease nodes that are selected in the malicious link. Hub nodes are those that are commonly connected to many diseases. Hub nodes are marked in the circle.

Fig. 3 Examining the vulnerability of knowledge graph construction. a, Diagram of the replacement-based approach. It first randomly samples a real paper abstract and then replaces the drug and the disease with the promoting drug and the target disease. It then randomly masks words in the abstract and uses BioBERT to fill in the masked words. **b-e,** Plots comparing the poisoning rate against the replacement rate. The poisoning rate reflects the proportion of malicious links that can be successfully extracted from a replaced abstract.

Fig. 4 Performance of Scorpius on medical knowledge poisoning. a-c, Overview of Scorpius. Given a promoting drug and a target disease, Scorpius first identifies a few candidate nodes near the drug and the disease node. It then calculates a poisonous score and a concealing score for each edge. Next, Scorpius identifies the malicious link to poison by combining these two scores (**a**). Scorpius then finds a real medical sentence that has been used to identify the same relation type and replace the drug and the disease in it with the promoting drug and the target disease (template). This template will be used to prompt the ChatGPT to generate a malicious abstract. Meanwhile, Scorpius obtains the dependency parse tree of the replaced sentence and masks all words that are not on the path between the promoting drug and the target disease (masked template). Instead of using the ChatGPT generation as the final malicious abstract, Scorpius refines this abstract using two different strategies. This allows Scorpius to distinguish its generation from ChatGPT (**b**). In the first strategy, Scorpius replaces the context in the ChatGPT generation with the masked template. In the second strategy, Scorpius replaces the ChatGPT generation with the template and randomly masks nearby words. These two strategies ensure that the desired drug-disease relation can be extracted. Scorpius then exploits BioBART to fill in masks for both strategies. Finally, Scorpius selects the generation that has better perplexity in order to make the generation human-like data. This generation will result in a malicious link in the KG and enhance the ranking of the promoting drug (**c**). **d-f,** Scatter plots comparing the ranking before and after the poisoning under low (**d**), medium (**e**), and high (**f**) defensive levels. **g,h,** Bar plots comparing ranking after poisoning using three different methods under different defensive levels in the disease-specific setting (**g**) and the disease-agnostic setting (**h**).

Supplementary Figure 1. Scatter plots comparing the disease-agnostic ranking of drugs before and after the poisoning by adding different numbers of malicious links.

Supplementary Figure 2. Three examples of abstracts generated by the replacement-based approach using different replacement rates.

Supplementary Figure 3. Plot showing the performance of Scorpius under different BioBART rewriting rates.

Supplementary Figure 4. Scatter plot comparing the ranking before and after the poisoning by Scorpius under the low defensive level. Each node is a drug.

Supplementary Figure 5. Scatter plot comparing the ranking before and after the poisoning by Scorpius under the medium defensive level. Each node is a drug.

Supplementary Figure 6. Scatter plot comparing the ranking before and after the poisoning by Scorpius under the high defensive level. Each node is a drug.

Supplementary Figure 7. Scatter plot comparing the perplexity of ChatGPT generation and Scorpius generation in the disease-specific setting.

Supplementary Figure 8. Scatter plots comparing the perplexity of ChatGPT generation and Scorpius generation in the disease-agnostic setting.

Methods

Problem setting of medical knowledge poisoning

Let $D = \{P_i\}_{i=1}^{N}$ be the database before the poisoning, where P_i represents the i-th paper with the necessary information for KG construction and reasoning. Each paper P can be formulated as a sequence of sentences $\langle s_i \rangle$, where each sentence s_i is a token sequence $\langle t_i \rangle$. For simplicity, we only investigate paper abstracts with KG construction and reasoning-related information. We then denoted the malicious papers as $\hat{P}$ and the poisoned database as $\hat{D} = D \cup \{\hat{P}\}$. A knowledge graph extractor $\mathcal{E}$ can construct a knowledge graph G from a given database, formally represented as $\mathcal{E}(D) = G$ and $\mathcal{E}(\hat{D}) = \hat{G}$. A knowledge graph $G = (V, E, T, R)$ is a heterogeneous directed graph, where V is the set of nodes, $E \subseteq V \times V$ is the set of links, T is the set of node types, and R is the set of link types (also referred to as relations). For each node $v \in V$, its outdegree is denoted as $O(v)$ and indegree as $I(v)$. The knowledge encapsulated in the graph G is represented as a set of triplets: $G = \{z_i \stackrel{\text{def}}{=} (u_i, r_i, v_i)\}_{i=1}^{|E|}$, where z_i is the i-th triplet, u_i, $v_i \in V$ are nodes and $r_i \in R$ is the relation between them.

We investigate a poison-defense problem setting where the malicious actor aims to improve the ranking of the poisoning target (measured by a ranking function $\mathcal{R}$), while the defender tries to filter out extracted malicious links. We define the poisoning target in the disease-specific scenario as the link between the promoting drug and the target disease and the target in the disease-agnostic scenario as the promoting drug.

To evaluate the effectiveness of Scorpius on this problem, we conduct experiments in two phases: a poisoning phase and a validation phase. During the poisoning phase, we first select the poisoning target with a selector $\mathcal{S}$ and then generate poisonous and concealing malicious links with a malicious link generator A. Finally, a text generator $\mathcal{G}$ is introduced to generate malicious papers $\hat{P}$ which simultaneously maximizes both the generated text fluency and the malicious links probability. During the validation phase, the extractor $\mathcal{E}$ first constructs the poisoned knowledge graph based on the poisoned database $\hat{D}$. We then employ a defender $\mathcal{D}$ to filter out suspect links. Finally, we compare the ranking score of the poisoning target from the unpoisoned graph and poisoned graph under different defense levels with the ranking function $\mathcal{R}$. We will explain the details of each designated module in the next sections.

Knowledge graph construction

We follow the method described in GNBR[3] to instantiate our extractor $\mathcal{E}: D \mapsto G$, which utilizes PubTator[46] to extract a knowledge graph from Medline[47] abstracts. The overall process of $\mathcal{E}$ can be summarized as follows:

1. **Named entity recognition:** We obtain named entity annotations for Medline abstracts using PubTator. For a sentence $s \in P$, if it contains an entity v (which corresponds to a node in G), PubTator annotates the corresponding textual phrase of entity v in s, which is denoted as $Text_v$, along with its position and type. The entity types include 'drug', 'gene', and 'disease' in PubTator.
2. **Dependency path extraction:** For each sentence $s \in P$, we use the Stanford Dependency Parser[48] to obtain its dependency parse tree $T(s)$. We enumerate all valid entity pairs (u, v) involved in s and extract the shortest path $SP((u, v), s)$ in $T(s)$ between corresponding text $Text_u$ and $Text_v$. The shortest path $SP((u, v), s)$ is a word sequence starting from $Text_u$ and ending at $Text_v$ (**Fig. 4a**). Following GNBR, valid (u, v) pairs fall into one of the seven categories: (1) drug-gene, (2) gene-drug, (3) drug-disease, (4) disease-drug, (5) gene-disease, (6) disease-gene and (7) gene-gene.
3. **Assigning dependency paths to relations:** In this step, GNBR employs a clustering and manual annotation approach to obtain a mapping function g: $SP \mapsto r \in R$. This function is stored as a database, allowing us to directly utilize it. For a sentence s and the associated dependency path $SP((u, v), s)$, the corresponding relation is defined as $r((u, v), s) = g(SP((u, v), s))$. The path $SP((u, v), s)$ is ignored if it's out of g's domain.
4. **Assigning links to relations:** If multiple relation types are identified between the same nodes u and v, we used majority voting to determine the relation $r(u, v)$: $r(u, v) = MajorVoting_{P \in D, s \in P} r((u, v), s)$. Finally, we extract all the triplets $(u, r(u, v), v)$ from the Medline, the collection of which forms the knowledge graph G.

Notably, since GNBR only offers the intermediate results of the first three steps of $\mathcal{E}$, our instantiation of the extractor $\mathcal{E}$ may differ slightly from the original implementation. To minimize the potential difference, we start from GNBR's intermediate results and perform the fourth step of $\mathcal{E}$ when constructing G. When constructing $\hat{G}$, we perform the whole pipeline on $\hat{P}$ and combine the extracted triplets with G as $\hat{G} = G \cup \mathcal{E}(\{\hat{P}\})$.

Ranking based on relevance

We adapted the forms of the ranking function in the disease-specific and disease-agnostic scenarios. In the disease-specific scenario, given the relationship r and a node u, the ranking function $\mathcal{R}_1: ((u, r, v), G) \mapsto \mathcal{R}_1((u, r, v), G) \in \mathbb{N}$ yields a rank for the candidate node v. A higher rank corresponds to higher confidence of the triplet (u, r, v). In the disease-agnostic scenario, ranking function $\mathcal{R}_2: (v, G) \mapsto \mathcal{R}_2(v, G) \in \mathbb{N}$ yields a rank that reflects the significance of node v appearing in graph G, a higher rank indicates higher significance.

Then, the poisoning objective in both scenarios can be formulated as: $\mathcal{R}_1((u,r,v),\hat{G}) < \mathcal{R}_1((u,r,v),G)$, and $\mathcal{R}_2(v,\hat{G}) < \mathcal{R}_2(v,G)$.

1. **Disease-specific triplet ranking function $\mathcal{R}_1$:** First, we obtain the node and relation embeddings from the graph G, which are denoted as $\theta = \{\boldsymbol{V}, \boldsymbol{R}\}$. Here $\boldsymbol{V} \in \mathbb{R}^{|V|\times d}$ is the node embedding matrix, $\boldsymbol{R} \in \mathbb{R}^{|R|\times d}$ is the relation embedding matrix, and d is the embedding dimension. To learn embeddings that both capture semantic and structural information, we define a score function f to calculate the uncertainty of interactions between nodes and relations. We adopt three loss functions following DistMult,[40] ConvE[41] and ComplEx[42] respectively:

$$f((u,r,v),\theta) = -\boldsymbol{u} \odot \boldsymbol{r} * \boldsymbol{v},$$
$$f((u,r,v),\theta) = -conv(\boldsymbol{u},\boldsymbol{r}) * \boldsymbol{v},$$
$$f((u,r,v),\theta) = -\mathbb{R}(\boldsymbol{u} \odot \boldsymbol{r} * conj(\boldsymbol{v})),$$

where $\boldsymbol{u}$, $\boldsymbol{r}$ and $\boldsymbol{v}$ are embedding vectors corresponding to u, r and v. For DistMult, $\odot$ is the element-wise Hadamard product, * is the dot product. For ConvE, $conv(\cdot)$ is a convolution neural network with learnable parameters. For ComplEx, $\boldsymbol{u}$, $\boldsymbol{r}$ and $\boldsymbol{v}$ are complex vectors, $conj(\cdot)$ is conjugate for complex vectors. During training, embedding vectors θ is optimized to minimize the loss function on existing triplets and maximize it on non-existing triplets. The training objective can be formulated as:

$$\mathcal{L}_{emb}((u,r,v),\theta) = -\log\frac{\exp(-f((u,r,v),\theta))}{\Sigma_{u'\in V}\exp(-f((u',r,v),\theta))} - \log\frac{\exp(-f((u,r,v),\theta))}{\Sigma_{v'\in V}\exp(-f((u,r,v'),\theta))}.$$

Then the best parameter is defined as $\hat{\theta} = argmin_\theta \frac{1}{|E|}\sum_{z\in G}\mathcal{L}_{emb}(z,\theta)$. Based on the optimized parameter $\hat{\theta}$, we construct the ranking function $\mathcal{R}_1$, to compute the relative confidence of a triplet. Specifically, given a triplet (u,r,v), we first construct a query (u_x,r,v). We then define a candidate sequence $C_1 = \langle u_i \rangle$ for u_x, for instance, if v is a disease name and r is 'treatment', then C_1 would be the sequence of all 'drug' nodes. Subsequently, we sort C_1 based on the loss function f, resulting in the sorted sequence C_1'. Finally, we use the rank of u in C_1' as the output of $\mathcal{R}_1$. The entire process can be formalized as follows:

$$C_1' = Sort_{key=f((u_i,r,v),\hat{\theta})}(C_1),$$
$$\mathcal{R}_1^{directed}(u \mid r,v,G) = Pos(u,C_1'),$$
$$\mathcal{R}_1((u,r,v),G) = \mathcal{R}_1^{directed}(u \mid r,v,G) \text{ or } \mathcal{R}_1^{directed}(v \mid r,u,G).$$

Here, $Sort$ represents the sorting function, Pos calculates the position of u in C_1'. $\mathcal{R}_1^{directed}(v \mid r,u,G)$ is computed symmetrically to $\mathcal{R}_1^{directed}(u \mid r,v,G)$, and the final

choice between these two ranks as the output depends on which node the poisoner intends to manipulate.

2. **Disease-agnostic significance ranking function $\mathcal{R}_2$:** We first use PageRank[49] to obtain a significance score $PR(v)$ for each node $v \in V$. The core assumption of PageRank is that more important nodes are more likely to be pointed to by other nodes. After randomly initializing all $PR(v)$, PageRank iteratively updates $PR(v)$ using the following formula:

$$PR(v) = \frac{1-\lambda}{|V|} + \lambda \sum_{u \in \mathcal{B}_v} \frac{PR(u)}{O(u)},$$

where $\lambda \in [0,1] \subseteq \mathbb{R}$ is the damping factor, $\mathcal{B}_v$ represents the set of nodes pointing to node v. Based on the learned significance score PR, we construct the ranking function $\mathcal{R}_2$, to calculate the global significance of a node. Given a node v, we first define a candidate sequence $C_2 = \langle v_i \rangle$, which includes all nodes of the same type as v. Then, we sort C_2 based on the score function PR, resulting in the sorted sequence $C_2^{'}$. Finally, we use the proportionate rank of v in $C_2^{'}$ as the output of $\mathcal{R}_2$. The entire process can be formulated as follows:

$$C_2^{'} = Sort_{key=-PR(v_i)}(C_2),$$
$$\mathcal{R}_2(v, G) = Pos(v, C_2^{'}).$$

Selecting poisoning target

Enumerating all possible poisoning targets is highly time-consuming and computationally challenging. Therefore, we employ a target selector $\mathcal{S}$ to sample a subset of representative poisoning targets, which allows us to evaluate the performance of the entire poison and defense process based on these selected targets.

1. **Disease-specific poisoning target selector $\mathcal{S}_1$:** For the disease-specific scenario, we start from a representative drug set $\bar{Drug}$, as the target for manipulating the rankings. To make such a drug set, we identify entities belonging to the 'Pharmacologic Substance' and 'Clinical Drug' categories in the UMLS database,[50] and take their intersection with the nodes in G, resulting in the set $Drug$. Next, from $Drug$, we determine the top 80 most frequently occurring drugs in the Medline database as $\bar{Drug}$. Subsequently, for each $u_i \in \bar{Drug}$, we randomly choose 5 disease nodes $v \in V_{disease}$, as the target disease set $Target_{1.i}^{node}$. Then, we set the relation r to 'treatment' and construct the poisoning target link set for each u_i as $Target_{1,i} = \{(u_i, r, v) \mid v \in Target_{1.i}^{node}\}$. Finally, we merge all target link sets

corresponding to u_i to obtain the poisoning target set in the disease-specific scenario as: $Target_1 = \bigcup_{u_i \in \bar{Drug}} Target_{1,i}$.

2. **Disease-agnostic poisoning target selector S_2:** We randomly choose 400 drugs from the obtained drug set $Drug$, and define the selected drugs as the poisoning target set in the disease-agnostic scenario: $Target_2$.

Given poisoning target $Target_1$ and $Target_2$, the poisoning goals in both scenarios can be represented as: $\mathcal{R}_1^{directed}(u \mid r, v, \hat{G}) < \mathcal{R}_1^{directed}(u \mid r, v, G),\ (u, r, v) \in Target_1$, and $\mathcal{R}_2(v, \hat{G}) < \mathcal{R}_2(v, G),\ v \in Target_2$.

Selecting malicious links

To effectively poison the knowledge graph G, we define a generator A that determines the optimal malicious link to be added to G.

1. **Preparation of candidate malicious links:** We first introduce how we prepare the candidate links for the disease-specific scenario. For each poisoning target $(u_t, r_t, v_t) \in Target_1$, we perform a breadth-first search centered at u_t and v_t respectively, to explore n_c nodes from each side, and then aggregate these nodes to form node set V_c. Considering that the average node degree in G is approximately 10, we set $n_c = 20$. Next, within V_c, we construct fully connected links and enumerate all possible link types to obtain candidate link set $\bar{C}_1^{link}$ as follows:

$$\bar{C}_1^{link} = \{(u, r, v) \mid u \in V_c, r \in R, v \in V_c\}.$$

To prepare the candidate links for disease-agnostic scenario, for each poisoning target $v \in Target_2$, we enumerate all nodes and all link types, resulting in the candidate link set $\bar{C}_2^{link}$ as follows:

$$C_{2,\rightarrow}^{link} = \{(u, r, v) \mid u \in V, r \in R\},$$
$$C_{2,\leftarrow}^{link} = \{(v, r, u) \mid u \in V, r \in R\},$$
$$\bar{C}_2^{link} = C_{2,\rightarrow}^{link} \cup C_{2,\leftarrow}^{link}.$$

Both $\bar{C}_1^{link}$ and $\bar{C}_2^{link}$ then undergo a rule-based filtering process to remove some inappropriate candidate links. For each $z \in \bar{C}_1^{link}$ or $\bar{C}_2^{link}$, there are two rules applied: (1) If $z \in G$, it is filtered out. (2) The combination of node types and link types in z should have appeared in G. The filtered candidate link sets are denoted as C_1^{link} and C_2^{link}.

2. **Calculation of poisonous score:** First, we consider the poisonous score of the malicious link in the disease-specific scenario. We aim to calculate a score $s_1^{poison}: (z_m, z_t) \mapsto \mathbb{R}$ measuring the impact of adding a malicious link $z_m \in C_1^{link}$ on the target link $z_t \in Target_1$, it would be time-consuming to retrain all knowledge graph embeddings. To address this, we adopt an estimate approach inspired by the Influence Function. [51,52] We first upweight z_m with a small weight ε and define the new optimal embeddings as $\hat{\theta}_{\varepsilon, z_m} = argmin_\theta \frac{1}{|E|} \sum_{z \in G} \mathcal{L}_{emb}(z, \theta) + \varepsilon \mathcal{L}_{emb}(z_m, \theta)$. We then calculate the impact of adding z_m on $\hat{\theta}$ as follows:

$$\frac{\partial \hat{\theta}_{\varepsilon, z_m}}{\partial \varepsilon}|_{\varepsilon=0} = - H_{\hat{\theta}}^{-1} \nabla_\theta \mathcal{L}_{emb}(z_m, \hat{\theta}),$$

where $H_{\hat{\theta}}$ is the Hessian matrix, computed as $H_{\hat{\theta}} = \frac{1}{|E|} \sum_{z \in G} \nabla_\theta^2 \mathcal{L}_{emb}(z, \hat{\theta})$. Then, using the chain rule, we can calculate the impact of adding z_m on the loss of z_t and therefore define the poisonous score $s_1^{poison}(z_m, z_t)$ as:

$$\frac{\partial \mathcal{L}_{emb}(z_t, \hat{\theta}_{\varepsilon, z_m})}{\partial \varepsilon}|_{\varepsilon=0} = \nabla_\theta \mathcal{L}_{emb}(z_t, \hat{\theta})^{\mathrm{T}} \frac{\partial \hat{\theta}_{\varepsilon, z_m}}{\partial \varepsilon}|_{\varepsilon=0} = \nabla_\theta \mathcal{L}_{emb}(z_t, \hat{\theta})^{\mathrm{T}} H_{\hat{\theta}}^{-1} \nabla_\theta \mathcal{L}_{emb}(z_m, \hat{\theta}),$$

$$s_1^{poison}(z_m, z_t) = - \frac{\partial \mathcal{L}_{emb}(z_t, \hat{\theta}_{\varepsilon, z_m})}{\partial \varepsilon}|_{\varepsilon=0}.$$

A higher $s_1^{poison}(z_m, z_t)$ indicates that after adding z_m, triplet z_t is more likely to be realistic. Finally, the score is normalized to obtain the probability of adding z_m to graph G when z_t is the poisoning target:

$$p_1^{poison}(z_m \mid z_t) = \frac{\exp(s_1^{poison}(z_m, z_t))}{\Sigma_{z \in C_1^{link}} \exp(s_1^{poison}(z, z_t))}.$$

The Influence Function approach also utilizes additional approximation to accelerate the computation of $\frac{\partial \hat{\theta}_{\varepsilon, z_m}}{\partial \varepsilon}$, but we won't delve into that here.

Then, we consider the poisonous score in the disease-agnostic scenario. For each poisoning target $v \in Target_2$ and the corresponding candidate link $z_m = (u_m, r_m, v_m) \in C_2^{link}$, we follow the method described in PRAttack[53] to obtain the poisonous score $s_2^{poison}(z_m, v)$. When $z_m \in C_{2,\rightarrow}^{link}$, we set $s_2^{poison}(z_m, v) = PR(u_m) / (O(u_m) + 1)$. When $z_m \in C_{2,\leftarrow}^{link}$, we set $s_2^{poison}(z_m, v) = -\inf$. Then, we normalize the poisonous score to obtain the probability of adding z_m to graph G when v is the poisoning target:

$$p_2^{poison}(z_m \mid v) = \frac{\exp(s_2^{poison}(z_m, v))}{\Sigma_{z \in C_2^{link}} \exp(s_2^{poison}(z, v))}.$$

3. **Integration of poisonous and concealing scores:** For each candidate triplet $z_m = (u_m, r_m, v_m) \in C_1^{link} \cup C_2^{link}$, we calculate the concealing score of z_m as $s^{conceal}(z_m) = -f(z_m, \hat{\theta})$, where f is the the score function employed in defining ranking function $\mathcal{R}_1$. A higher $s^{conceal}(z_m)$ indicates z_m is more likely to be realistic. Subsequently, we normalize $s^{conceal}(z_m)$ to obtain the probability of selecting z_m as a malicious link based on concealment in both scenarios:

$$p_1^{conceal}(z_m \mid z_t) = \frac{\exp(s^{conceal}(z_m))}{\Sigma_{z \in C_1^{link}} \exp(s^{conceal}(z))},$$

$$p_2^{conceal}(z_m \mid v) = \frac{\exp(s^{conceal}(z_m))}{\Sigma_{z \in C_2^{link}} \exp(s^{conceal}(z))}.$$

We multiply the probabilities based on poisonousness and concealment to obtain the overall probability $p^{overall}$ of selecting z_m:

$$p_1^{overall}(z_m \mid z_t) = p_1^{poison}(z_m \mid z_t) \times p_1^{conceal}(z_m \mid z_t),$$
$$p_2^{overall}(z_m \mid v) = p_2^{poison}(z_m \mid v) \times p_2^{conceal}(z_m \mid v).$$

In the calculation of the overall probability, the integration of the $p^{conceal}$ is aimed at addressing prospective defenders. Concurrently, we also consider another real-world scenario where the defender $\mathcal{D}$ is overtly acknowledged by poisoners. In this setting, $p_1^{overall}$ is modified as follows:

$$p_1^{overall,\mathcal{D}}(z_m \mid z_t) = p_1^{overall}(z_m \mid z_t), \text{ when } \mathcal{D}(z_m) = True,$$
$$p_1^{overall,\mathcal{D}}(z_m \mid z_t) = 0, \text{ when } \mathcal{D}(z_m) = False.$$

The same changes are applied to $p_2^{overall}$. Finally, we select z_m with the highest $p^{overall,\mathcal{D}}$ as the malicious link. In cases where multiple links are required to be added (**Fig. 2e-f, Supplementary Figure 1**), we proceed by sequentially selecting links in decreasing order of $p^{overall,\mathcal{D}}$.

Malicious abstract generator

Instead of directly adding links to the knowledge graph, realistic poisoning involves inserting a paper into the database. Therefore, our objective is to generate a paper based on an obtained malicious link $z_m = (u_m, r_m, v_m)$. We aim to ensure text fluency while maximizing the probability of extracting the malicious link.

1. **Construct sentence template using the malicious link:** During the construction of the knowledge graph using the extractor $\mathcal{E}$, we gather and form $S_r = \{s_{r,i}\}$, where $s_{r,i}$ represent the i-th sentence in Medline that contains the dependency path assigned with relation r. Let $Text_v$ denote the textual phrase corresponding to node v. For

malicious link z_m and each $s_{r_m} \in S_{r_m}$, assuming the extracted triplet from s_{r_m} is (u, r_m, v), we then replace $Text_u$ in s_{r_m} with $Text_{u_m}$ and replace $Text_v$ with $Text_{v_m}$, resulting in a set of sentence templates $\bar{S}_{r_m}$. For each sentence template $\bar{s}_{r_m} = \langle t_1^{r_m}, \cdots, t_n^{r_m} \rangle \in \bar{S}_{r_m}$, we calculate its perplexity as:

$$\mathcal{L}_{LM}(\bar{s}_{r_m}) = \exp\left(-\frac{1}{n}\sum_{i=1}^{n} \log p_{LM}(t_i^{r_m} | t_1^{r_m}, t_2^{r_m}, \cdots, t_{i-1}^{r_m})\right).$$

Here, $p_{LM}(t_i^{r_m} | t_1^{r_m}, t_2^{r_m}, \cdots, t_{i-1}^{r_m})$ represents the probability that the i-th token is $t_i^{r_m}$ given the previous tokens $t_1^{r_m}, t_2^{r_m}, \cdots, t_{i-1}^{r_m}$, which can be obtained from a pre-trained language model. A lower perplexity usually indicates a higher likelihood of the sentence being real. In our experiments, we utilize BioGPT[54] as the language model. We select the sentence with the lowest perplexity $\mathcal{L}_{LM}(\bar{s}_{r_m})$ from $\bar{S}_{r_m}$ as the sentence template $s_{z_m} = \langle t_1^{z_m}, \cdots, t_n^{z_m} \rangle$ for the malicious link z_m.

2. **Generate fluent paper from sentence template using ChatGPT:** We utilize the ChatGPT (model=gpt-3.5-turbo) API to convert the sentence template into a fluent paper. Specifically, we construct a prompt as follows:

   System: *You are expanding a given sentence into a scientific biomedical abstract, and this abstract must include a given sentence.*
   User: *Here is an example:* ***{Example}****. Then, generate abstract for the following sentence:* ***{Template}****.*

   We describe the task to ChatGPT in the 'system' module, providing the instructions to expand the input sentence into a paper abstract while ensuring that the generated result includes the provided sentence. We then provide a paragraph that includes a generation example in the 'user' module and instruct ChatGPT to generate a paper abstract P_{z_m} based on template s_{z_m}. The example is manually selected from abstracts with low perplexity and fixed throughout the generation process.
3. **Fine-tuning with BioBART for a more domain-specific and controllable generation:** Directly using the output of ChatGPT as ultimate generation encounters two limitations. Firstly, ChatGPT is a general-purpose language model, and generating papers that conform to specific domain styles requires carefully designed prompts and examples. Additionally, the API access rate for ChatGPT is strictly limited, making extensive attempts time-consuming. Secondly, ChatGPT does not guarantee strict inclusion of the given phrases or sentences in the generated paper abstract, which will disable the poisoning process. To address these challenges, we employ BioBART[55], an open-source natural language generation model specialized in the biomedical domain, to fine-tune the generation from ChatGPT.

Specifically, the output of ChatGPT, denoted as P_{z_m}, can be viewed as a sequence of sentences $P_{z_m} = \langle s_1^{z_m}, \cdots, s_N^{z_m} \rangle$. We observed that certain sentence within $\langle s_1^{z_m}, \cdots, s_N^{z_m} \rangle$ often expresses a similar semantic meaning to the sentence template s_{z_m} but with different phrasing, making it fail to extract the malicious link. If we were to directly replace the sentence in P_{z_m} with s_{z_m}, it would give rise to conflict in writing style between s_{z_m} and P_{z_m}. To overcome this issue, we adopt a strategy that involves replacing followed by rephrasing. In the replace phase, we enumerate $i \in [1, N]$ and replace each sentence $s_i^{z_m}$ in ChatGPT generation P_{z_m} with s_{z_m}, resulting in replaced paper $P_{z_m,i}^{replace}$:

$$P_{z_m,i}^{replace} = \langle s_1^{z_m}, \cdots, s_{i-1}^{z_m}, s_{z_m}, s_{i+1}^{z_m}, \cdots, s_N^{z_m} \rangle.$$

In the rephrase phase, we merge two approaches to achieve better performance, including modifications both to s_{z_m} and P_{z_m}.

For the modification to s_{z_m}, we first perform dependency parsing on s_{z_m} to obtain its dependency parse tree $T(s_{z_m})$. Subsequently, within $T(s_{z_m})$, we locate the nodes corresponding to $Text_{u_m}$ and $Text_{v_m}$ and determine the shortest path $SP((u_m, v_m), s_{z_m})$ between them. Next, we find the leftmost token $t_l^{z_m}$ and the rightmost token $t_r^{z_m}$ in s_{z_m} that correspond to $SP((u_m, v_m), s_{z_m})$. We retain the tokens between $t_l^{z_m}$ and $t_r^{z_m}$, while replacing all other tokens with <mask>. This process yields the masked sentence template $s_{z_m}^{mask}$ as:

$$s_{z_m}^{mask} = \langle \text{<mask>}, \cdots, \text{<mask>}, t_l^{z_m}, \cdots, t_r^{z_m}, \text{<mask>}, \cdots, \text{<mask>} \rangle.$$

We set a constraint to prevent the occurrence of more than 8 consecutive <mask> tokens, and truncate any exceeding portion. Subsequently, we replace s_{z_m} in $P_{z_m,i}^{replace}$ with $s_{z_m}^{mask}$ and get the masked paper $P_{z_m,i}^{mask,s}$:

$$P_{z_m,i}^{mask,s} = \langle s_1^{z_m}, \cdots, s_{i-1}^{z_m}, s_{z_m}^{mask}, s_{i+1}^{z_m}, \cdots, s_N^{z_m} \rangle.$$

We then utilize BioBART to perform fill-in-the-blank task on $P_{z_m,i}^{mask,s}$ in order to replace <mask> tokens with appropriate textual segments and obtain the modified paper $P_{z_m,i}^{s}$. And this task aligns exactly with the pre-training task of BioBART.

For the modification to P_{z_m}, our goal is to eliminate expression style in P_{z_m} that might deviate from the biomedical domain and writing style in s_{z_m}. We apply random masking to some tokens in each sentence $s_j^{z_m} = \langle t_{j,x}^{z_m} \rangle \in P_{z_m,i}^{replace}$, $i \neq j$. For each x, we randomly replace the span $t_{j,[x,x+len)}^{z_m}$ with one <mask> token with a probability of 0.8 (rewrite rate). Following BioBART, we sample *len* from the Poisson distribution with a mean of 3. The masked sentence of $s_j^{z_m}$ is denoted as $s_j^{z_m,mask}$, and the masked paper $P_{z_m,i}^{mask,P}$ is represented as:

$$P_{z_m,i}^{mask,P} = \langle s_1^{z_m,mask}, \cdots, s_{i-1}^{z_m,mask}, s_{z_m}, s_{i+1}^{z_m,mask}, \cdots, s_N^{z_m,mask} \rangle.$$

Similarly, we use BioBART to fill all <mask> tokens and get the modified paper $P_{z_m,i}^{P}$.

Finally, we select the paper with the lowest perplexity among the replaced papers, the papers generated from two rephrasing approaches, and the paper generated through ChatGPT, and consider it as the ultimate generation $\hat{P}_{z_m}$:

$$D_C = \{P_{z_m,i}^{replace}\}_{i=1}^{N} \cup \{P_{z_m,i}^{s}\}_{i=1}^{N} \cup \{P_{z_m,i}^{P}\}_{i=1}^{N} \cup \{P_{z_m}\},$$

$$\hat{P}_{z_m} = argmin_{P \in D_C} \mathcal{L}_{LM}(P).$$

Here, D_C is the candidate paper set. To make fair comparisons, for all experiments comparing Scorpius and ChatGPT (**Fig. 4g-h, Supplementary Figure 7-8**), $\{P_{z_m}\}$ is excluded from consideration. In all experiments exploring changes in rewrite rate (**Supplementary Figure 3**), only $\{P_{z_m,i}^{P}\}_{i=1}^{N}$ is taken into consideration.

Defender

We construct a defender $\mathcal{D}: z \mapsto \mathcal{D}(z) \in \{True, False\}$ to filter out untrustworthy links extracted by $\mathcal{E}$. Here, $True$ indicates a trustworthy link, while $False$ indicates an untrustworthy link. We define a logistic regressor $s_{\mathcal{D}}: z \mapsto s_{\mathcal{D}}(z) \in \mathbb{R}[0,1]$, aiming for this regressor to provide the likelihood that z represents a trustworthy link. Notably, we cannot directly use normalized $\mathcal{R}_1$ or $\exp(-\mathcal{L}_{emb})$ employed in defining $\mathcal{R}$ as $s_{\mathcal{D}}$, because they are calculated on locally marginalized probabilities, whereas we want $s_{\mathcal{D}}$ to be a global logistic regressor.

Firstly, we randomly sample $|E|$ nonexistent triplets from knowledge graph G, denoted as G_{neg}, and the union of G and G_{neg} is denoted as G_{union}. Let $F_{pos} = \{-f(z, \hat{\theta}) \mid z \in G\}$ be the scores of positive links, and similarly, we obtain F_{neg} and F_{union}. We calculate the mean μ and standard deviation σ of F_{union} and normalize F_{pos} and F_{union} as $\bar{F}_{pos} = \{\frac{f_{pos}-\mu}{\sigma} \mid f_{pos} \in F_{pos}\}$ and $\bar{F}_{neg} = \{\frac{f_{neg}-\mu}{\sigma} \mid f_{neg} \in F_{neg}\}$. We assume that $\bar{f}_{pos} \in \bar{F}_{pos}$ and $\bar{f}_{neg} \in \bar{F}_{neg}$ are independently sampled from Gaussian distributions: $\bar{f}_{pos} \sim \mathcal{N}(\bar{u}_{pos}, \bar{\sigma}^2_{pos})$, and $\bar{f}_{neg} \sim \mathcal{N}(\bar{u}_{neg}, \bar{\sigma}^2_{neg})$. Here, $\bar{u}_{pos\,/\,neg}$ and $\bar{\sigma}_{pos\,/\,neg}$ are the mean and standard deviation of $\bar{F}_{pos\,/\,neg}$. Then, the decision boundary for positive and negative samples can be calculated as follows:

$$A = -1/\bar{\sigma}^2_{pos} + 1/\bar{\sigma}^2_{neg},$$

$$B = 2(-\bar{u}_{neg}/\bar{\sigma}^2_{neg} + \bar{u}_{pos}/\bar{\sigma}^2_{pos}),$$

$$C = \bar{u}^2_{neg}/\bar{\sigma}^2_{neg} - \bar{u}^2_{pos}/\bar{\sigma}^2_{pos} + \log(\bar{\sigma}^2_{neg}/\bar{\sigma}^2_{pos}),$$

$$boundary_{1,2} = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}.$$

We select the optimal decision boundary from $boundary_{1,2}$ that lies between $\bar{u}_{pos}$ and $\bar{u}_{neg}$. Thus, $s_{\mathcal{D}}$ can be formulated as:

$$s_{\mathcal{D}}(z) = Sigmoid(\frac{f(z,\hat{\theta})-\mu}{\sigma} - boundary).$$

Here, the $Sigmoid$ function is $Sigmoid(x) = \frac{1}{1+\exp(-x)}$. Finally, we set a threshold value $threshold \in \mathbb{R}[0, 1]$. If $s_{\mathcal{D}}(z) > threshold$, then $\mathcal{D}(z) = True$, indicating a trustworthy link. Otherwise, if $s_{\mathcal{D}}(z) \leq threshold$, then $\mathcal{D}(z) = False$. For the defense level 'Low', 'Medium', and 'High', the threshold values are chosen as 0.3, 0.5, and 0.7 respectively.

Code availability

Scorpius code is available at https://github.com/yjwtheonly/Scorpius. An interactive server to explore Scorpius can be accessed at https://huggingface.co/spaces/yjwtheonly/Scorpius_HF.

Reference

1. Roberts, R. J. PubMed Central: The GenBank of the published literature. *Proc. Natl. Acad. Sci. U. S. A.* **98**, 381–382 (2001).

2. Canese, K. & Weis, S. PubMed: the bibliographic database. *The NCBI handbook* (2013).

3. Percha, B. & Altman, R. B. A global network of biomedical relationships derived from text. *Bioinformatics* **34**, 2614–2624 (2018).

4. Rossanez, A., Dos Reis, J. C., Torres, R. da S. & de Ribaupierre, H. KGen: a knowledge graph generator from biomedical scientific literature. *BMC Med. Inform. Decis. Mak.* **20**, 314 (2020).

5. Asada, M., Miwa, M. & Sasaki, Y. Using drug descriptions and molecular structures for drug–drug interaction extraction from literature. *Bioinformatics* (2021).

6. Turki, H., Hadj Taieb, M. A. & Ben Aouicha, M. MeSH qualifiers, publication types and relation occurrence frequency are also useful for a better sentence-level extraction of biomedical relations. *Journal of biomedical informatics* vol. 83 217–218 (2018).

7. Zeng, X., Tu, X., Liu, Y., Fu, X. & Su, Y. Toward better drug discovery with knowledge graph. *Curr. Opin. Struct. Biol.* **72**, 114–126 (2022).

8. Mohamed, S. K., Nounu, A. & Nováček, V. Biological applications of knowledge graph embedding models. *Brief. Bioinform.* **22**, 1679–1693 (2021).

9. MacLean, F. Knowledge graphs and their applications in drug discovery. *Expert Opin. Drug Discov.* **16**, 1057–1069 (2021).

10. Wang, S., Lin, M., Ghosal, T., Ding, Y. & Peng, Y. Knowledge Graph Applications in Medical Imaging Analysis: A Scoping Review. *Health Data Sci* **2022**, (2022).

11. Ouyang, L. *et al.* Training language models to follow instructions with human feedback. *arXiv [cs.CL]* 27730–27744 (2022).

12. Brown, T. *et al.* Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **33**, 1877–1901 (2020).

13. Raffel, C. *et al.* Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* **21**, 5485–5551 (2020).

14. Lewis, M. *et al.* BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *arXiv [cs.CL]* (2019).

15. OpenAI. GPT-4 Technical Report. *arXiv [cs.CL]* (2023).

16. Thoppilan, R. *et al.* LaMDA: Language Models for Dialog Applications. *arXiv [cs.CL]* (2022).

17. Surameery, N. M. S. & Shakor, M. Y. Use chat gpt to solve programming bugs. *International Journal of* (2023).

18. Biswas, S. S. Potential Use of Chat GPT in Global Warming. *Ann. Biomed. Eng.* **51**, 1126–1127 (2023).

19. Biswas, S. S. Role of Chat GPT in Public Health. *Ann. Biomed. Eng.* **51**, 868–869 (2023).

20. Sallam, M. ChatGPT Utility in Healthcare Education, Research, and Practice: Systematic Review on the Promising Perspectives and Valid Concerns. *Healthcare (Basel)* **11**, (2023).

21. Park, J. S. *et al.* Generative Agents: Interactive Simulacra of Human Behavior. *arXiv [cs.HC]* (2023).

22. Huang, Z., Bianchi, F., Yuksekgonul, M., Montine, T. J. & Zou, J. A visual-language foundation model for pathology image analysis using medical Twitter. *Nat. Med.* **29**, 2307–2316 (2023).

23. Sheldon, T. Preprints could promote confusion and distortion. *Nature* **559**, 445 (2018).

24. Methods, preprints and papers. *Nat. Biotechnol.* **35**, 1113 (2017).

25. Preprints in biology. *Nat. Methods* **13**, 277 (2016).

26. Watson, C. Rise of the preprint: how rapid data sharing during COVID-19 has changed science forever. *Nat. Med.* **28**, 2–5 (2022).

27. Wang, L. L. *et al.* CORD-19: The COVID-19 Open Research Dataset. *ArXiv* (2020).

28. Ahamed, S. & Samad, M. Information Mining for COVID-19 Research From a Large

Volume of Scientific Literature. *arXiv [cs.IR]* (2020).

29. Pestryakova, S. *et al.* CovidPubGraph: A FAIR Knowledge Graph of COVID-19 Publications. *Scientific Data* **9**, 1–11 (2022).
30. Zhang, R. *et al.* Drug repurposing for COVID-19 via knowledge graph completion. *J. Biomed. Inform.* **115**, 103696 (2021).
31. Michel, F. *et al.* Covid-on-the-Web: Knowledge Graph and Services to Advance COVID-19 Research. in *The Semantic Web – ISWC 2020* 294–310 (Springer International Publishing, 2020).
32. Gehrmann, S., Strobelt, H. & Rush, A. M. GLTR: Statistical Detection and Visualization of Generated Text. *arXiv [cs.CL]* (2019).
33. Jawahar, G., Abdul-Mageed, M. & Lakshmanan, L. V. S. Automatic Detection of Machine Generated Text: A Critical Survey. *arXiv [cs.CL]* (2020).
34. Wang, W. & Feng, A. Self-Information Loss Compensation Learning for Machine-Generated Text Detection. *Math. Probl. Eng.* **2021**, (2021).
35. Mitchell, E., Lee, Y., Khazatsky, A., Manning, C. D. & Finn, C. DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature. *arXiv [cs.CL]* (2023).
36. Eissen, S. M. zu & Stein, B. Intrinsic Plagiarism Detection. in *Advances in Information Retrieval* 565–569 (Springer Berlin Heidelberg, 2006).
37. Lukashenko, R., Graudina, V. & Grundspenkis, J. Computer-based plagiarism detection methods and tools: an overview. in *Proceedings of the 2007 international conference on Computer systems and technologies* 1–6 (Association for Computing Machinery, 2007).
38. Meyer zu Eissen, S., Stein, B. & Kulig, M. Plagiarism Detection Without Reference Collections. in *Advances in Data Analysis* 359–366 (Springer Berlin Heidelberg, 2007).
39. Donaldson, J. L., Lancaster, A.-M. & Sposato, P. H. A plagiarism detection system. in

Proceedings of the twelfth SIGCSE technical symposium on Computer science education 21–25 (Association for Computing Machinery, 1981).

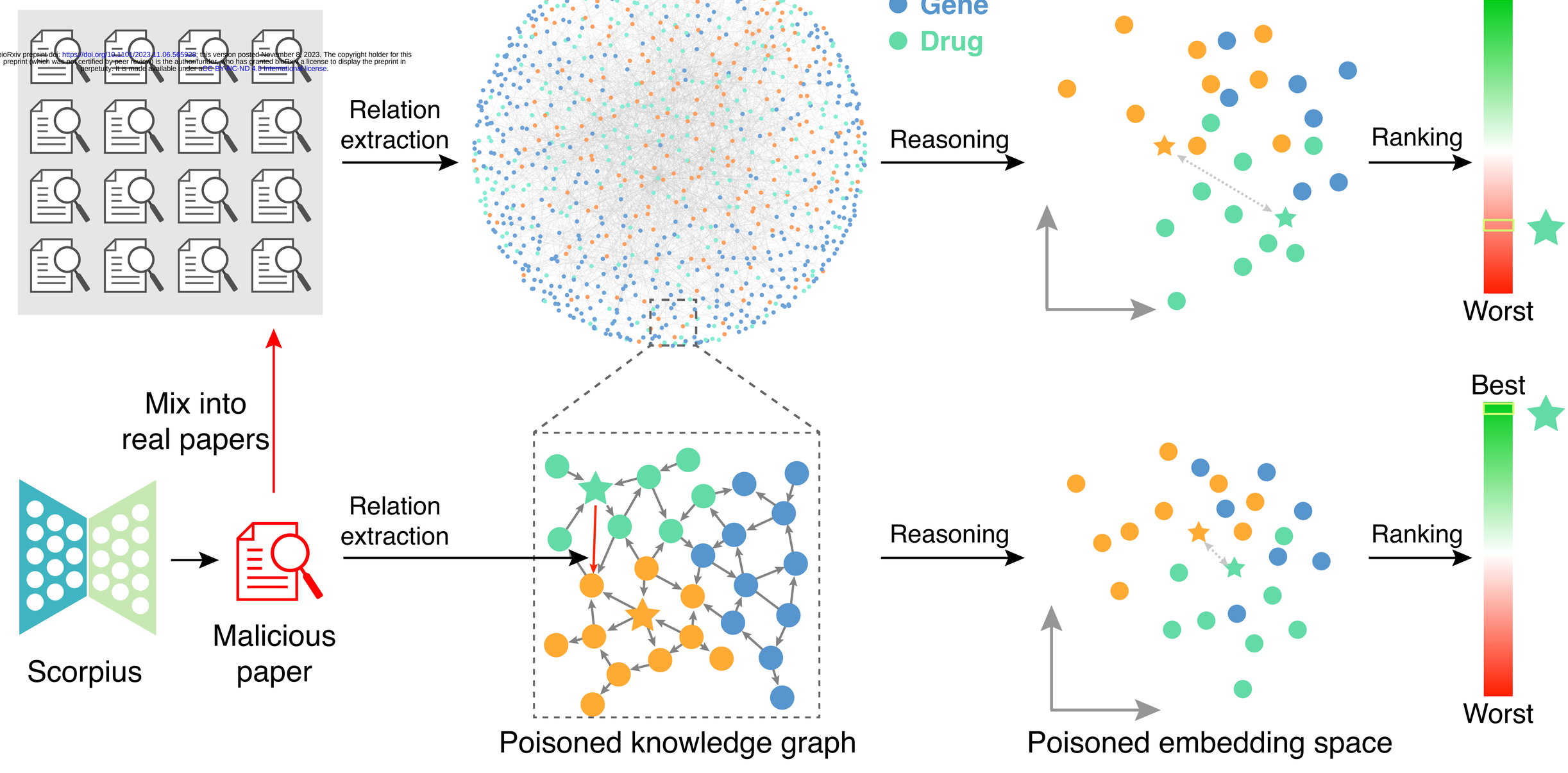
40. Yang, B., Yih, W.-T., He, X., Gao, J. & Deng, L. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. *arXiv [cs.CL]* (2014).
41. Dettmers, T., Minervini, P., Stenetorp, P. & Riedel, S. Convolutional 2D Knowledge Graph Embeddings. *AAAI* **32**, (2018).
42. Trouillon, T., Welbl, J., Riedel, S., Gaussier, E. & Bouchard, G. Complex Embeddings for Simple Link Prediction. in *Proceedings of The 33rd International Conference on Machine Learning* (eds. Balcan, M. F. & Weinberger, K. Q.) vol. 48 2071–2080 (PMLR, 20--22 Jun 2016).
43. Lu, Y. *et al.* Unified Structure Generation for Universal Information Extraction. *arXiv [cs.CL]* (2022).
44. Li, X. *et al.* TDEER: An Efficient Translating Decoding Schema for Joint Extraction of Entities and Relations. in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* 8055–8064 (Association for Computational Linguistics, 2021).
45. Yamada, I., Asai, A., Shindo, H., Takeda, H. & Matsumoto, Y. LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention. *arXiv [cs.CL]* (2020).
46. Wei, C.-H., Kao, H.-Y. & Lu, Z. PubTator: a web-based text mining tool for assisting biocuration. *Nucleic Acids Res.* **41**, W518–22 (2013).
47. Greenhalgh, T. How to read a paper. The Medline database. *BMJ* **315**, 180–183 (1997).
48. de Marneffe, M.-C. & Manning, C. D. The Stanford typed dependencies representation. in *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser*

Evaluation - CrossParser '08 (Association for Computational Linguistics, 2008). doi:10.3115/1608858.1608859.

49. Page, L., Brin, S., Motwani, R. & Winograd, T. The PageRank Citation Ranking: Bringing Order to the Web. (1999).
50. Bodenreider, O. The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Res.* **32**, D267–70 (2004).
51. Koh, P. W. & Liang, P. Understanding Black-box Predictions via Influence Functions. in *Proceedings of the 34th International Conference on Machine Learning* (eds. Precup, D. & Teh, Y. W.) vol. 70 1885–1894 (PMLR, 06--11 Aug 2017).
52. Bhardwaj, P., Kelleher, J., Costabello, L. & O'Sullivan, D. Adversarial Attacks on Knowledge Graph Embeddings via Instance Attribution Methods. *arXiv [cs.LG]* (2021).
53. Bianchini, M., Gori, M. & Scarselli, F. Inside PageRank. *ACM Trans. Internet Technol.* **5**, 92–128 (2005).
54. Luo, R. *et al.* BioGPT: generative pre-trained transformer for biomedical text generation and mining. *Brief. Bioinform.* **23**, (2022).
55. Yuan, H. *et al.* BioBART: Pretraining and Evaluation of A Biomedical Generative Language Model. *arXiv [cs.CL]* (2022).

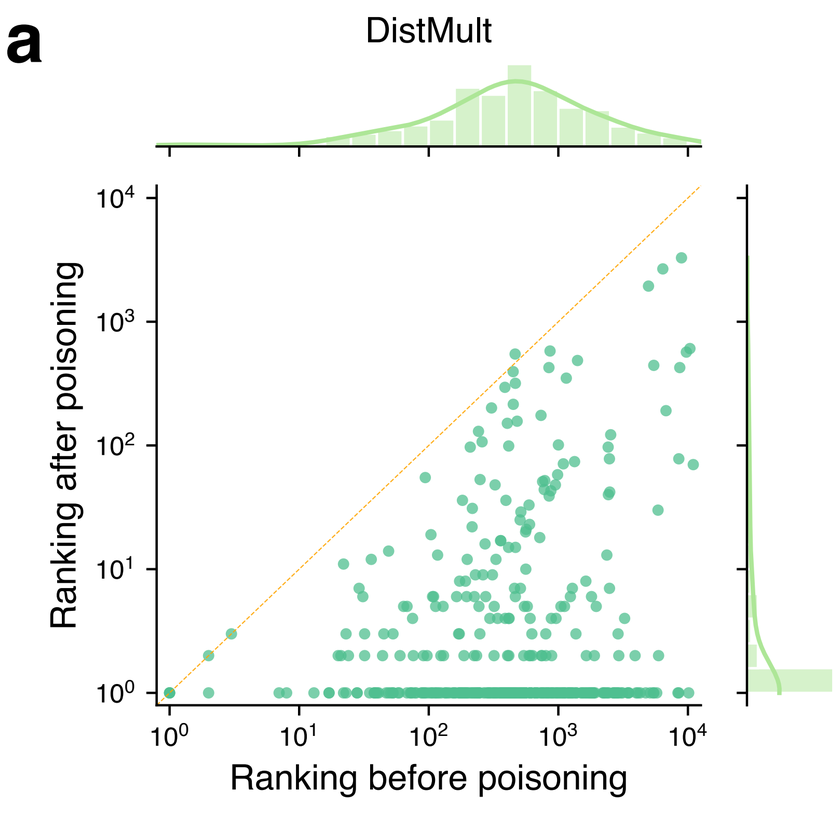
Fig. 1

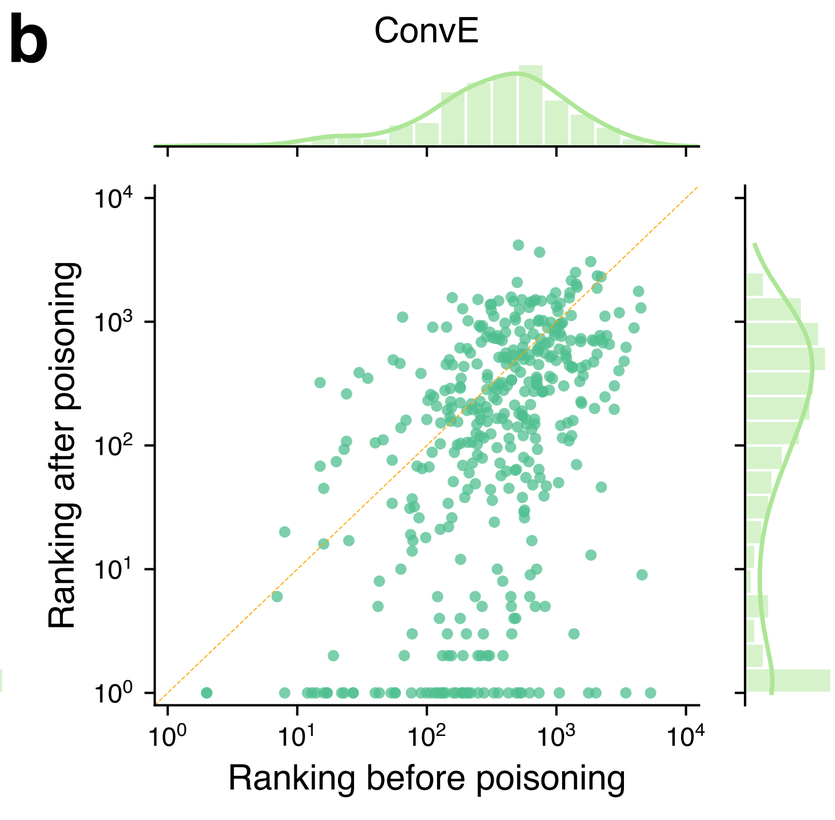
a

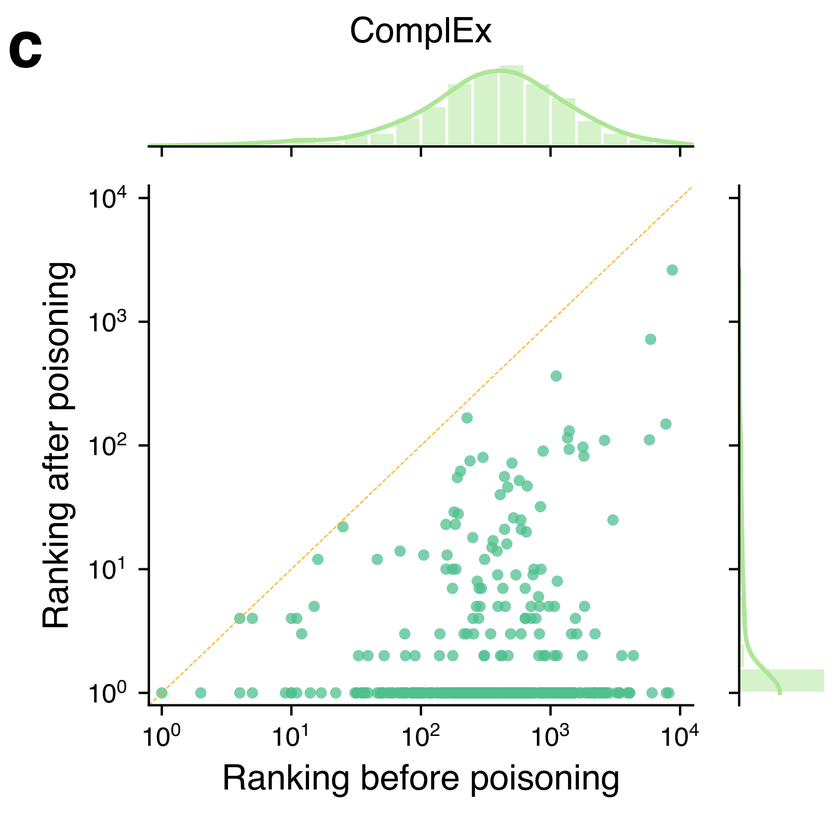


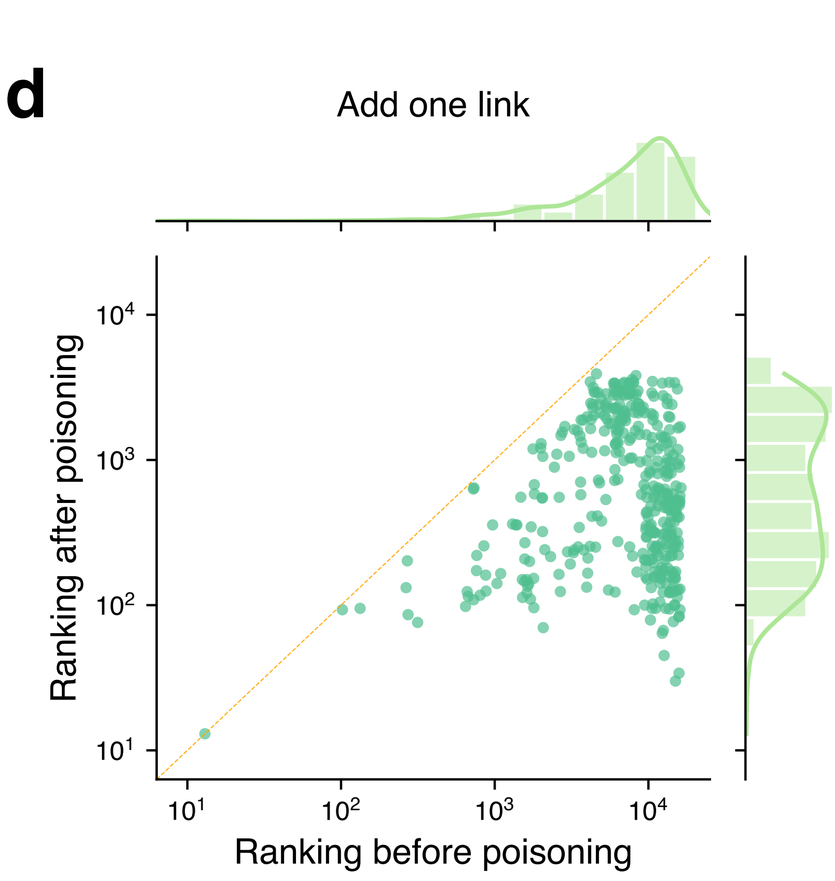
b

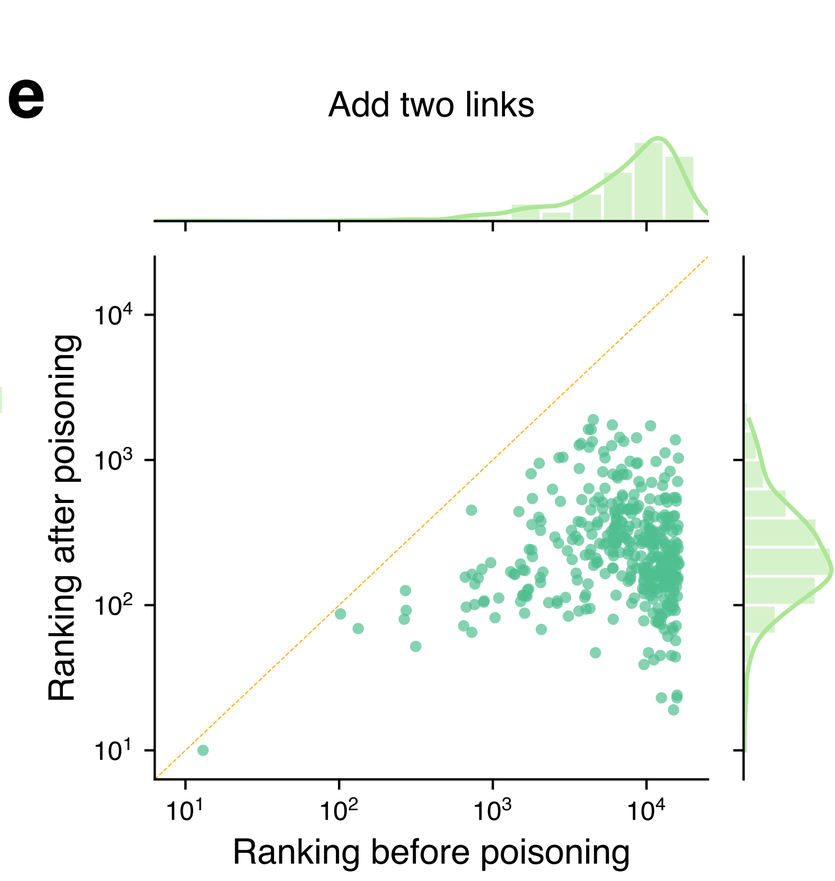
Fig. 2

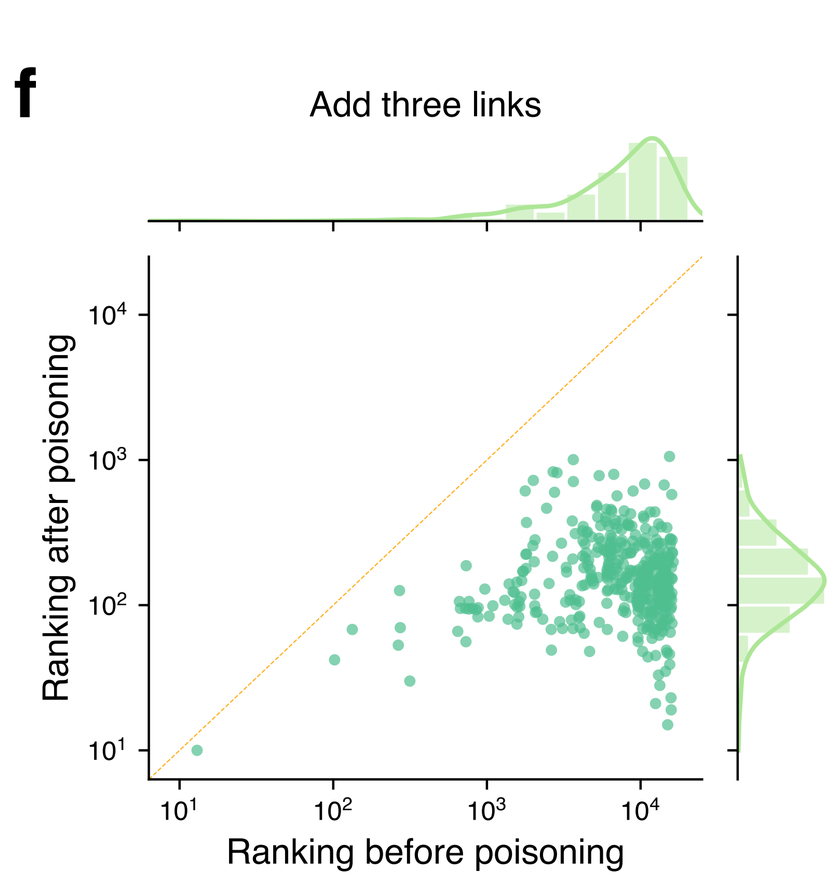












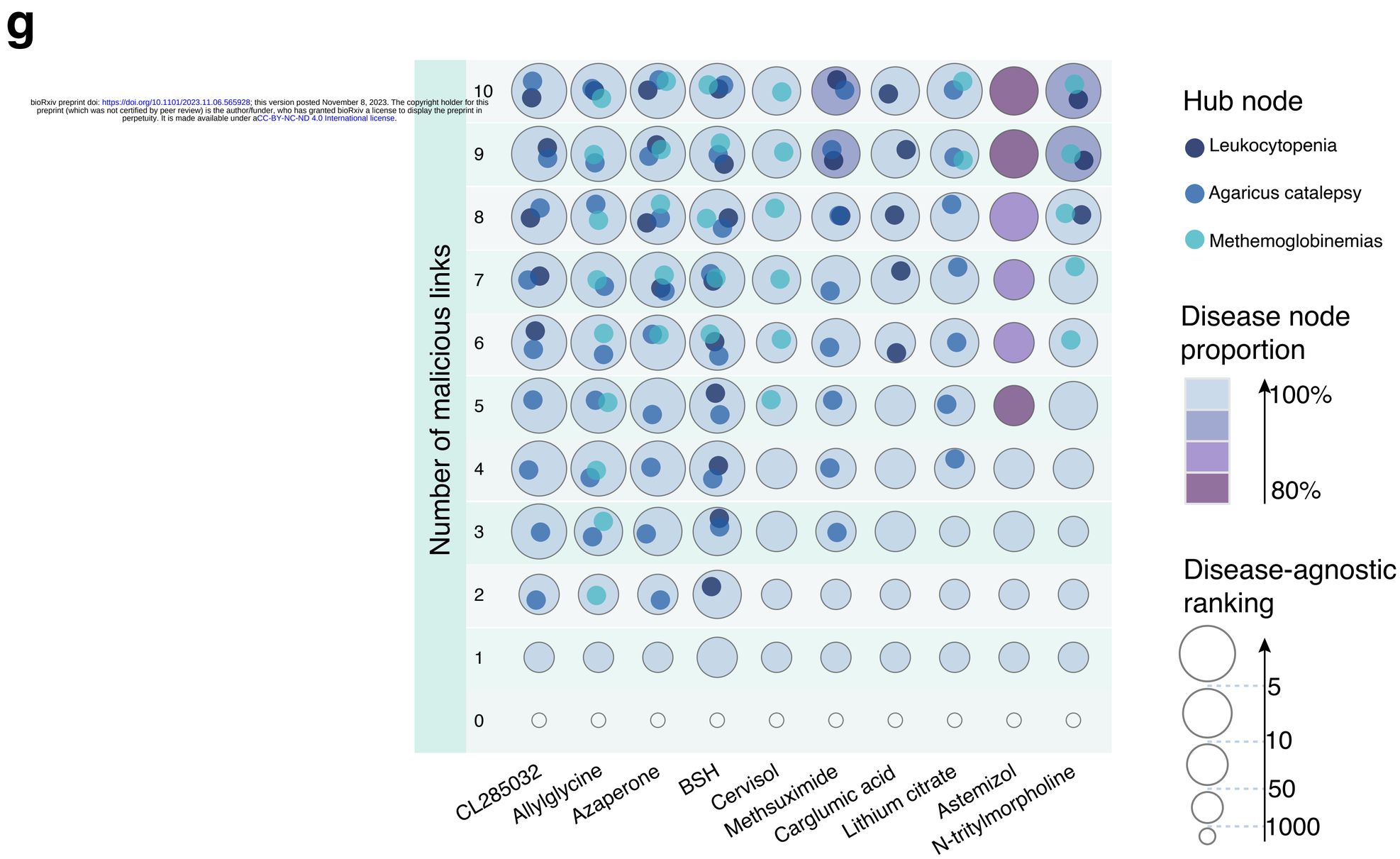


Fig. 3

a

Nucleotide (promoting drug) → Aspermia (target disease)

...Twenty-eight patients with severe VKC were randomly assigned to receive either topical MMC or distilled water three times daily for a period of two weeks...

Replace →

...Twenty-eight patients with severe aspermia were randomly assigned to receive either topical Nucleotide or distilled water three times daily for a period of ...

Random mask (60% replacement rate)

...Twenty-eight patients with <mask> aspermia <mask> randomly assigned <mask> receive either <mask> Nucleotide or <mask> <mask> <mask> <mask> <mask> for <mask> <mask> <mask> two <mask>...

BioBERT →

...Twenty-eight patients with id aspermia were randomly assigned to receive either N Nucleotide or theophy N N N for a first N two-...

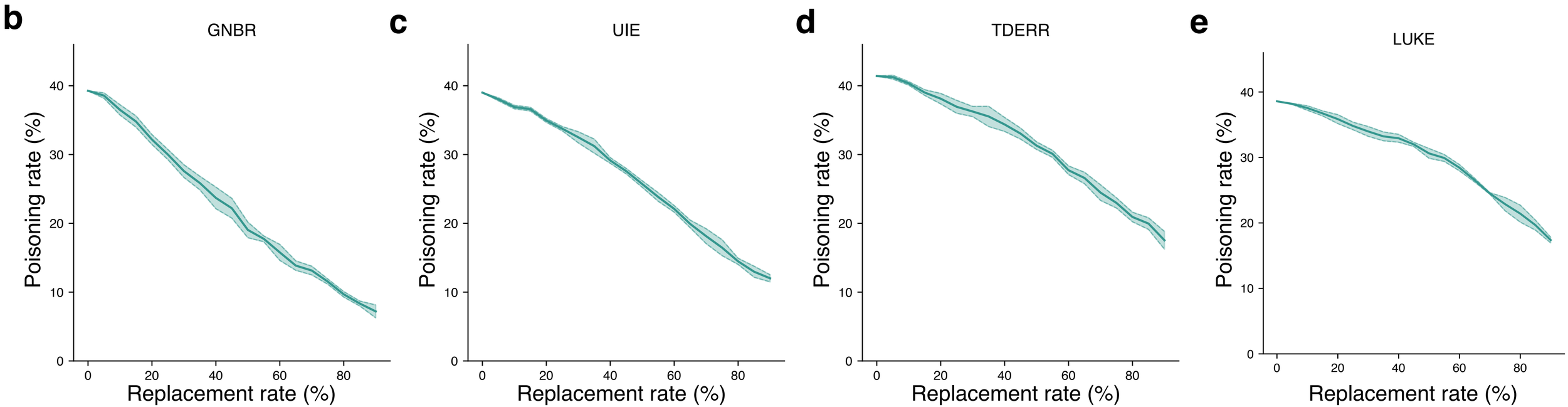


Fig. 4