nature machine intelligence

Article

https://doi.org/10.1038/s42256-024-00899-3

Poisoning medical knowledge using large language models

Received: 11 October 2023

Accepted: 15 August 2024

Published online: 20 September 2024

Check for updates

Junwei Yang[1], Hanwen Xu[2], Srbuhi Mirzoyan[1], Tong Chen[2], Zixuan Liu[2], Zequn Liu[1], Wei Ju[1], Luchen Liu[1], Zhiping Xiao[2] ✉, Ming Zhang[1] ✉ & Sheng Wang[2] ✉

Biomedical knowledge graphs (KGs) 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. Although 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 an LLM to generate a malicious paper that poisons medical KGs 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 promoted drug and a target disease. The goal is to fool the medical KG constructed from a mixture of this malicious abstract and millions of real papers so that KG consumers will misidentify this promoted drug as relevant to the target disease. We evaluated Scorpius on a KG constructed from 3,818,528 papers and found that Scorpius can increase the relevance of 71.3% drug–disease pairs from the top 1,000 to the top ten by adding only 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 KGs and manipulating downstream applications using LLMs, indicating the importance of accountable and trustworthy medical knowledge discovery in the era of LLMs.

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]. Although 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.

[1]School of Computer Science, State Key Laboratory for Multimedia Information Processing, Peking University-Anker Embodied AI Lab, Peking University, Beijing, China. [2]Paul G. Allen School of Computer Science and Engineering, University of Washington, Seattle, WA, USA. ✉e-mail: patricia.xiao@gmail.com; mzhang_cs@pku.edu.cn; swang@cs.washington.edu

We want to investigate whether an 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 greater 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 promoted 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 promoted 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 promoted drug and a target disease, directly linking them is often insufficiently concealing and easy for a defender to detect. Therefore, Scorpius first identifies an absent KG link to poison by considering both a poisonous score and a concealing score we defined. Scorpius then exploits ChatGPT to generate a malicious abstract by using the promoted 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 1,000 to the top ten by adding only 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 KGs

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 (for example, scientists) use this KG to identify the relevant drug for 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 promoted 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 promoted 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 promoted 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 artificial-intelligence-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 KGs are vulnerable

We first sought to examine the second step in the KG-based medical knowledge discovery, which reflects the vulnerability of medical KGs. In particular, we built a KG that contains 16,468 drugs, 5,379 diseases and 38,080 genes from 3,818,528 medical papers (Methods). We then examined the proportion of drugs that can obtain a substantial relevance increase after adding just 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 promoted 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 promoted drugs substantially increased on all three methods after the poisoning. In particular, 48.2% and 64.3% of drugs are ranked as the top one and in the top ten after the poisoning, which is much higher than 0.3% and 1.9% before the poisoning. Although all three methods are vulnerable to this poisoning, the drug relevance increased more on DistMult and ComplEx than on ConvE. Because 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 adding only one malicious link demonstrates the vulnerability of medical KGs, 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 adding only 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 (analysis of variance $P < 8 \times 10^{-79}$). The increase converged after adding ten links (Supplementary Fig. 1). We listed ten drugs that have the largest relevance increase after adding ten links and found that four of them can achieve a top-ten ranking by only adding four links to this large KG (Fig. 2g). We noticed that a few diseases are commonly selected by these ten 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.

https://doi.org/10.1038/s42256-024-00899-3

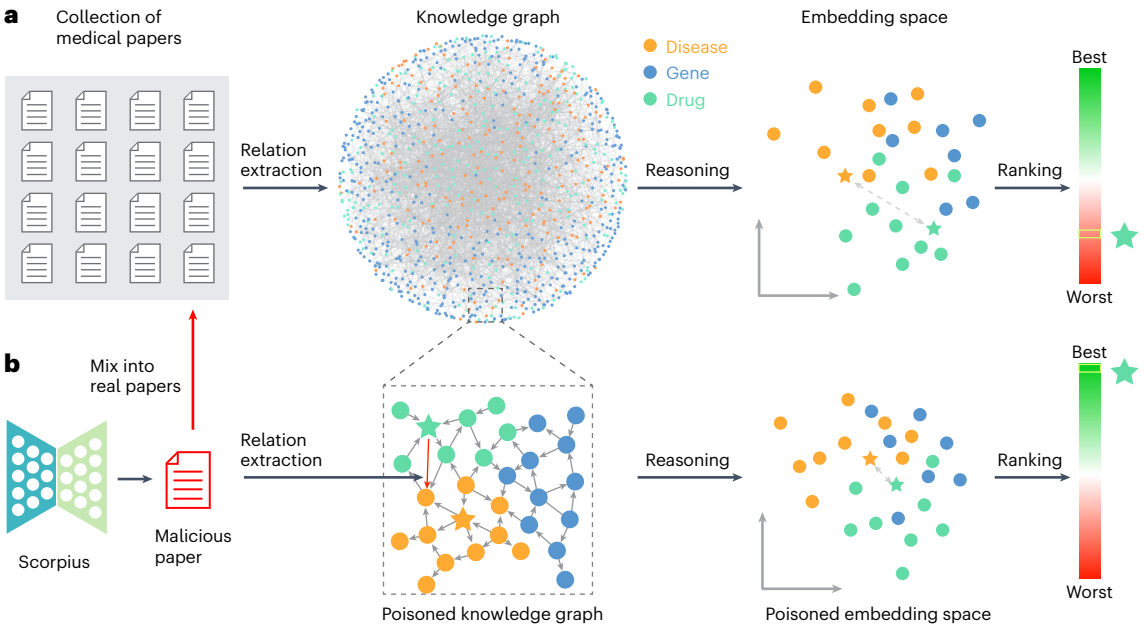


Fig. 1 | Overview of medical knowledge poisoning. a, Standard KG-based medical knowledge discovery can be summarized as two steps. The first step is KG construction, where relation-extraction methods are applied to a collection of medical papers. Each extracted relation will become one link in the KG. The second step is KG reasoning, where nodes (for example, 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 promoted drug and a target disease will be substantially different.

KG 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 (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, and thus it cannot be identified by existing plagiarism systems[36–39]. We assessed four different relation-extraction methods, including global network of biomedical relationships (GNBR)[3], universal information extraction (UIE)[43], translating decoding schema for joint extraction of entities and relations (TDERR)[44] and deep contextualized entity representations with entity-aware self-attention (LUKE)[45]. GNBR is an expertise-driven relation-extraction method specialized for constructing biomedical KGs, while the remaining three are data-driven methods used for general domains. 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, at least 20% of node pairs can still 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 Fig. 2). This motivates us to develop Scorpius for generating human-like text data that can poison KG construction.

Scorpius poisons KGs

After confirming the vulnerability of both medical KGs and KG 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 to be concealed. It then exploited ChatGPT to generate an abstract conditioned on the promoted 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 (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 ($P < 2 \times 10^{-32}$) and low defensive levels ($P < 4 \times 10^{-106}$) (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 an insertion approach and five text-generation methods (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. RAG-GPT-3.5 and RAG-GPT-4 represent the direct use of ChatGPT's output, differing in the model invoked. Scorpius (GPT-3.5) and Scorpius (GPT-4) use BioBART to rewrite RAG-GPT-3.5 and RAG-GPT-4 outputs. We found that Scorpius substantially outperformed the corresponding version of RAG-GPT on all three defence levels ($P < 7 \times 10^{-3}$ between RAG-GPT-3.5 and Scorpius (GPT-3.5), $P < 2 \times 10^{-2}$ between RAG-GPT-4 and Scorpius (GPT-4)), indicating the effectiveness of further refining the ChatGPT generation using BioBART. Because Scorpius can be adapted to different versions of LLMs, their increasing capabilities will further empower Scorpius. Moreover, the performance of Scorpius did not drop substantially compared to the

https://doi.org/10.1038/s42256-024-00899-3

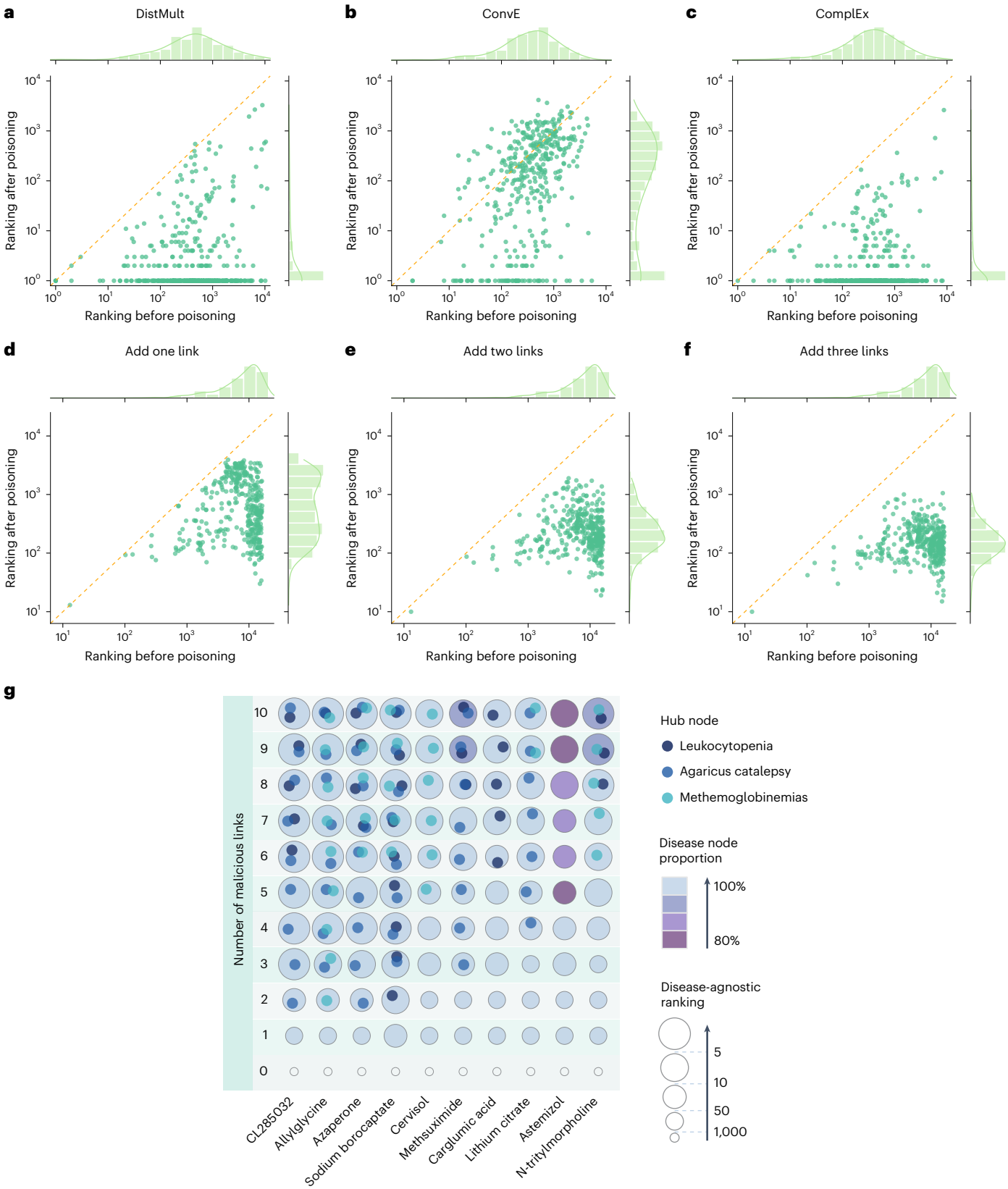


Fig. 2 | Examining the vulnerability of medical KGs. a–**c**, Scatter plots comparing the disease-specific ranking of drugs before and after the poisoning using three KG reasoning approaches: DistMult (**a**), ConvE (**b**) and ComplEx (**c**). **d**–**f**, Scatter plots comparing the disease-agnostic 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 ten links. Circle size represents ranking. Circle colour 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.

https://doi.org/10.1038/s42256-024-00899-3

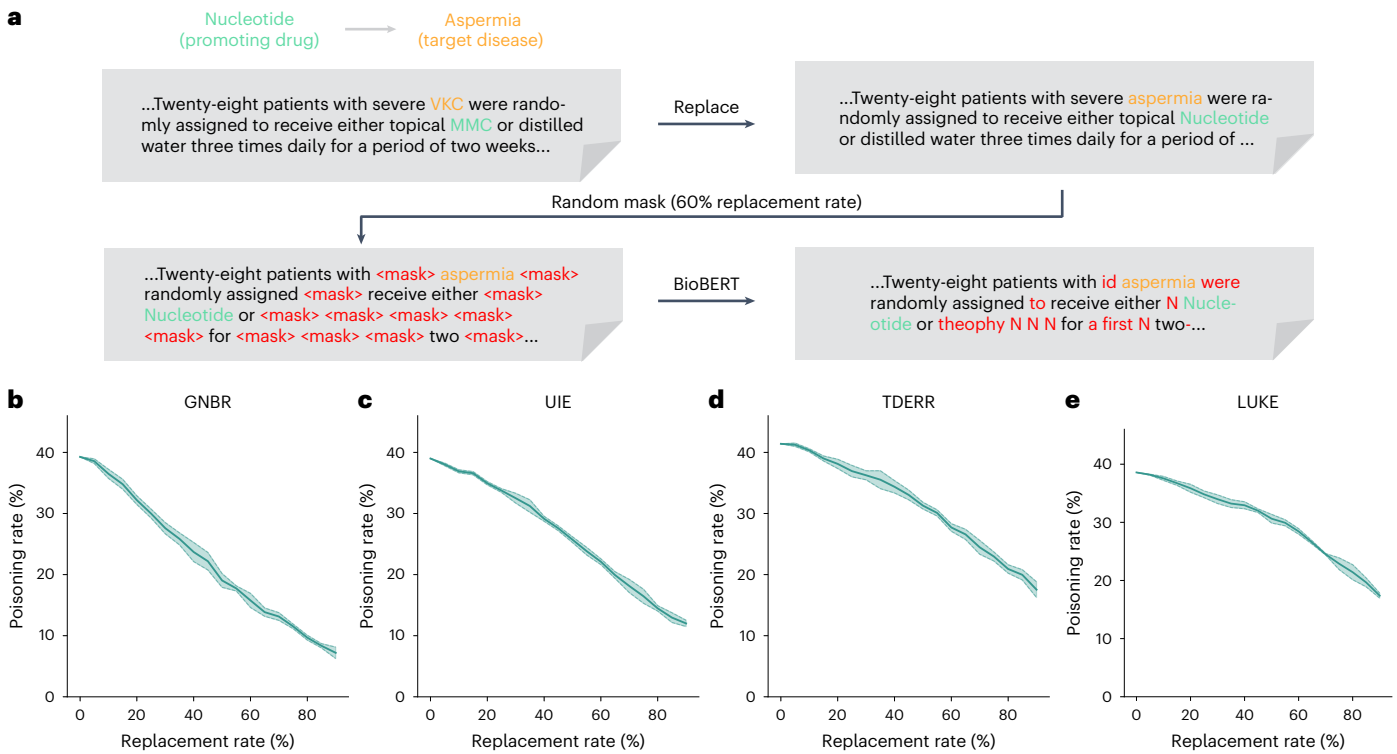


Fig. 3 | Examining the vulnerability of KG construction. a, Diagram of the replacement-based approach. It first randomly samples a real paper abstract and then replaces the drug and the disease with the promoted drug and the target disease. It then randomly masks words in the abstract and uses BioBERT[72], a pre-trained biomedical language model, to fill in the masked words. **b**–**e**, Plots comparing the poisoning rate against the replacement rate for GNBR (**b**), UIE (**c**), TDERR (**d**) and LUKE (**e**). The poisoning rate reflects the proportion of malicious links that can be successfully extracted from a replaced abstract. Data are presented as mean values ± s.d. across $n = 2{,}000$ random samples.

insertion approach, suggesting 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 Fig. 3).

Furthermore, 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 the insertion approach and five text-generation methods under three defensive levels (Fig. 4h, Supplementary Figs. 4–6). We found that Scorpius again outperformed RAG-GPT on all three settings ($P < 4 \times 10^{-13}$ between RAG-GPT-3.5 and Scorpius (GPT-3.5) and $P < 8 \times 10^{-3}$ between RAG-GPT-4 and Scorpius (GPT-4)). We noted that Finetune-GPT-3.5 performed comparably with Scorpius (GPT-3.5) ($P > 2 \times 10^{-1}$), but the cost of Finetune-GPT-3.5 was nearly ten times that of Scorpius (GPT-3.5) (Methods). We also observed 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.

Investigate factors that affect poisoning effectiveness

We further investigated five factors that might affect the effectiveness of poisoning (Methods). First, we compared the KG constructed from peer-reviewed articles and another KG constructed from bioRxiv preprints of the same size. We found that the bioRxiv-based KG is more vulnerable than the peer-reviewed-based KG at all three defence levels in the disease-specific scenario and at the low defence level in the disease-agnostic scenario (Supplementary Figs. 7–8). We attribute this to the higher quality of papers in peer-reviewed systems, leading to better resistance against scientific poisoning. Second, we studied whether the size and heterogeneity of KGs could affect the poisoning. We found that larger and more heterogeneous KGs are more resistant to poisoning (Supplementary Figs. 9–11). Nevertheless, even on our most complex KG, which contains 120,000 nodes and ten node types, Scorpius still achieved strong poisoning results at low defence levels ($P < 8 \times 10^{-67}$, Supplementary Fig. 11g,h). These results collectively indicated the importance of constructing a high-quality, large and heterogeneous KG to defend against potential poisoning.

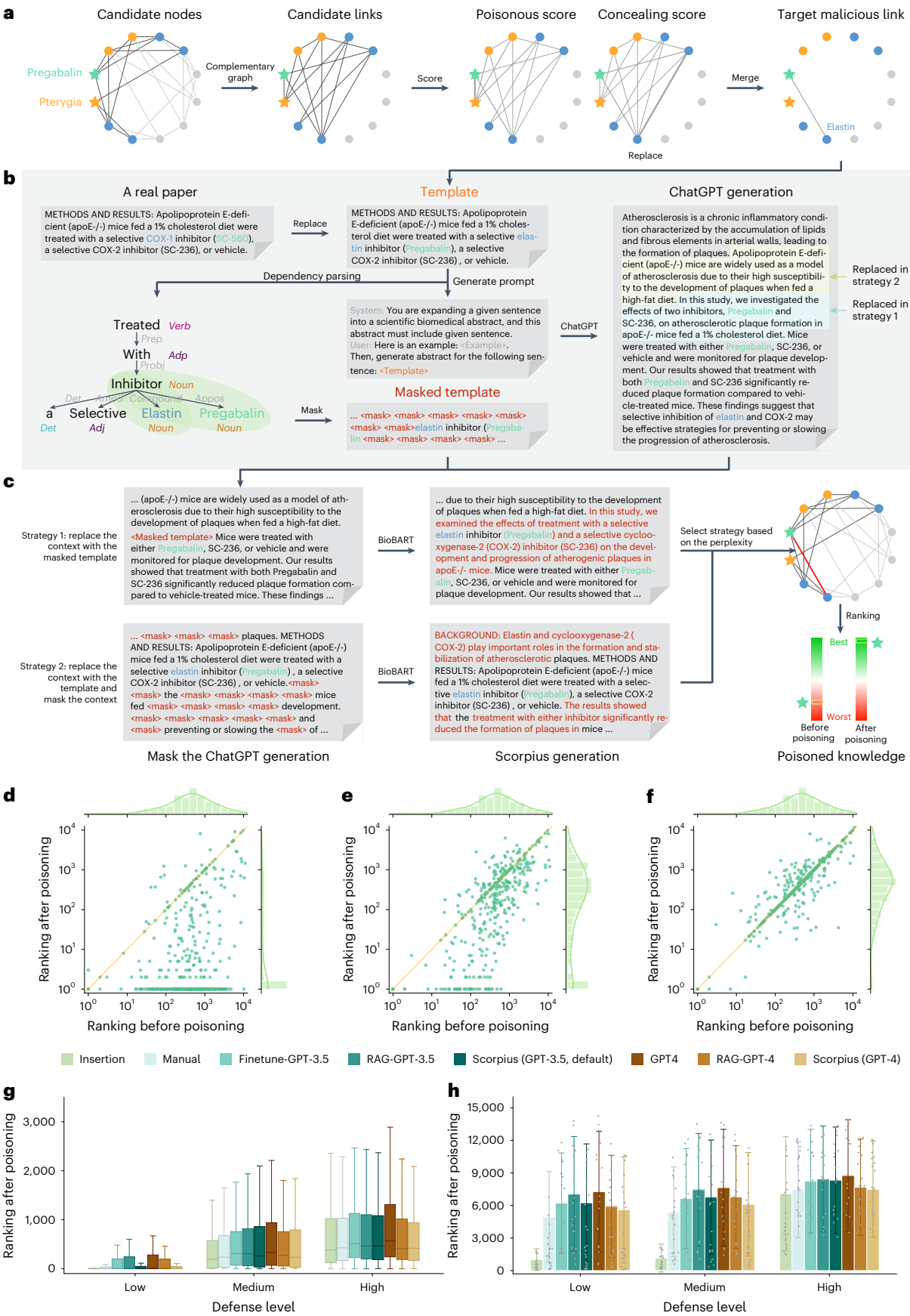
Next, we studied the impact of the rarity of promoted drugs and found that rare drugs (Supplementary Figs. 12 and 13) and new drugs (Supplementary Fig. 14) are more vulnerable to poisoning. We further developed a GPT-4-based defender and observed that this defender could not effectively distinguish between real papers and Scorpius-generated malicious papers (Supplementary Fig. 15a) and 78.2% of malicious papers can pass the corresponding stringent defence (Supplementary Fig. 15b), suggesting that a GPT-4-based defender cannot fully address the vulnerability of KG reasoning (Supplementary Fig. 15c,d).

Finally, we evaluated the generation quality of Scorpius based on perplexity, GPT-4-based scoring and manual evaluation (Methods). The results indicate that Scorpius has better perplexity than ChatGPT in both disease-specific and disease-agnostic settings (Supplementary Figs. 16 and 17). In the case of GPT-4-based scoring, Scorpius demonstrated slightly lower context coherence, similar writing fluency and higher scientific faithfulness than ChatGPT, similar to what we observed on manually written abstracts (Supplementary Fig. 18). Moreover, we observed that Scorpius-generated text is not easily distinguishable from manually written abstracts by human evaluators (Supplementary Fig. 19).

Discussion

We have studied a novel problem of medical knowledge poisoning, where a malicious paper is generated by LLMs to poison medical KGs

https://doi.org/10.1038/s42256-024-00899-3



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 that of ChatGPT on a KG 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

Fig. 4 | Performance of Scorpius on medical knowledge poisoning. **a**–**c**, Overview of Scorpius. Given a promoted 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 replaces the drug and the disease in it with the promoted drug and the target disease (template). This template will be used to prompt 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 promoted 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 to make the generation human-like data. This generation will result in a malicious link in the KG and enhance the ranking of the promoted drug (**c**). **d**–**f**, Scatter plots comparing the ranking before and after poisoning under low (**d**), medium (**e**) and high (**f**) defensive levels. **g**,**h**, Box and bar plots comparing ranking after poisoning using eight different methods under different defensive levels in the disease-specific setting (**g**) and the disease-agnostic setting (**h**). Data in the box plot are presented with the centre representing the median, the bounds representing the 25th and 75th percentiles and whiskers extending to the smallest and largest values within 1.5 times the interquartile range. Data in the bar plot are presented as mean values ± s.d. Both plots are based on $n = 400$ random samples.

papers and the possibility of influencing downstream applications by using LLMs to generate a malicious paper.

Our research is related to KG poisoning, which involves manipulating KGs by adding or removing links before training to promote or suppress specific facts. Current KG poisoning techniques can be classified into two categories. The first category is single-link poisoning[46–49], which models the impact of a single link on the poisoning target with influence function calculation[46–48] and contrastive learning[49]. The second category is path poisoning[50–52], which typically models the effects of a reasoning path on the poisoning target using methods such as path generation[50,51] and path propagation[52]. These methods assume that the malicious actor can directly manipulate KGs, which is impractical in real-world scenarios. In contrast, Scorpius investigates the poisoning of KGs starting from the collection of scientific papers, a setting that is more realistic and has not been previously explored.

Our study raises concerns about the reliance on preprints in scientific research. Unlike peer-reviewed papers, preprints undergo a less rigorous review process, which makes preprint-based KGs more susceptible to poisoning compared to peer-reviewed-based KGs. This underscores the risk of scientific misinformation that does not go through peer review. Furthermore, our findings reveal that even humans face challenges in differentiating between papers authored by LLMs and those written by humans. This also highlights the need for caution in using KG systems that incorporate peer-reviewed content, particularly as the capabilities of LLMs continue to advance.

Our work can further reveal the immediate risks underneath multiple KG reasoning-based real-world implications. Biomedical KGs are widely used in drug discovery[7,9,53]. For example, KG-Predict[53] uses InteractE[54] to infer new drug–disease interactions on a KG constructed from literature, making it possible to be poisoned by Scorpius. Moreover, biomedical KGs constructed from the literature have attracted the attention of chemists, who have used them for wet lab experiments in drug discovery. For instance, Standigm ASK[55] was utilized for drug discovery related to idiopathic pulmonary fibrosis. Such KGs have also been used by different pharmaceutical companies in their products. For example, MindRank uses PharmKG[56] to assist in the discovery of MRANK-106. Therefore, poisoning KGs with Scorpius would have substantial implications in real-world drug discovery and clinical studies.

There are a few limitations we would like to address in the future. First, 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 KG reasoning performance. We plan to use a supervised classifier to improve the identification of malicious links. Second, the existing framework does not consider the timestamp of each paper. Intuitively, emerging topics (for example, COVID-19) are more likely to be poisoned because they have larger visibility. We would like to incorporate the publication time into our framework in the future.

Methods

Problem setting of medical knowledge poisoning

Let $D = \{P_i\}_{i=1}^{N}$ be the database before poisoning, where P_i represents the ith 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 KG extractor $\mathcal{E}$ can construct a KG G from a given database, formally represented as $\mathcal{E}(D) = G$ and $\mathcal{E}(\hat{D}) = \hat{G}$. A KG $G = (V, E, T, R)$ 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: $G = \{z_i \overset{\text{def}}{=} (u_i, r_i, v_i)\}_{i=1}^{|E|}$, where z_i is the ith triplet, $u_i, v_i \in V$ are nodes and $r_i \in R$ is the relation between them.

We investigate a poison-defence problem setting where the malicious actor aims to improve the ranking of the poisoning target (measured by a ranking function $\mathcal{R}$), whereas the defender tries to filter out extracted malicious links. We define the poisoning target in the disease-specific scenario as the link between the promoted drug and the target disease and the target in the disease-agnostic scenario as the promoted 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 $\hat{P}$ that simultaneously maximizes both the generated text fluency and the malicious link probability. During the validation phase, the extractor $\mathcal{E}$ first constructs the poisoned KG 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 defence levels with the ranking function $\mathcal{R}$. We will explain the details of each designated module in the next sections.

KG construction

We follow the method described in GNBR[3] to instantiate our extractor $\mathcal{E} : D \to G$, which utilizes PubTator[57] to extract a KG from Medline[58] 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[59] to obtain its dependency parse tree $T(s)$. We enumerate all valid entity pairs (u, v) involved in s and extract the shortest path $\text{SP}((u, v), s)$ in $T(s)$ between corresponding Text_u and Text_v. The shortest path $\text{SP}((u, v), s)$ 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 $g : \text{SP} \longmapsto r \in R$. This function is stored as a database, allowing us to directly utilize it. For a sentence s and the associated dependency path $\text{SP}((u, v), s)$, the corresponding relation is defined as $r((u, v), s) = g(\text{SP}((u, v), s))$. The path $\text{SP}((u, v), s)$ is ignored if it is 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 $r(u, v)$: $r(u, v) = \text{MajorVoting}_{p \in D, s \in P} r((u, v), s)$. Finally, we extract all the triplets $(u, r(u, v), v)$ from Medline, the collection of which forms the KG G.

Notably, because 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.

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 $\mathcal{R}_1 : ((u, r, v), G) \to \mathcal{R}_1((u, r, v), G) \in \mathbb{N}$ 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 $\mathcal{R}_2 : (v, G) \to \mathcal{R}_2(v, G) \in \mathbb{N}$ yields a rank that reflects the importance of node v appearing in graph G; a higher rank indicates higher importance. 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 $\mathcal{R}_2(v, \hat{G}) < \mathcal{R}_2(v, G)$.

Disease-specific triplet ranking function $\mathcal{R}_1$. First, we obtain the node and relation embeddings from the graph G, which are denoted as $\theta = \{X, Y\}$. Here, $X \in \mathbb{R}^{|V| \times d}$ is the node embedding matrix, $Y \in \mathbb{R}^{|R| \times d}$ 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:

$$f((u, r, v), \theta) = -\mathbf{u} \odot \mathbf{r} * \mathbf{v},$$

$$f((u, r, v), \theta) = -\text{conv}(\mathbf{u}, \mathbf{r}) * \mathbf{v},$$

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

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

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

Then the best parameter is defined as $\hat{\theta} = \text{argmin}_\theta \frac{1}{|E|} \sum_{z \in G} \mathcal{L}_{\text{emb}}(z, \theta)$. 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:

$$C_1' = \text{Sort}_{\text{key}=f((u_i,r,v),\hat{\theta})}(C_1),$$

$$\mathcal{R}_1^{\text{directed}}(u|r, v, G) = \text{Pos}(u, C_1'),$$

$$\mathcal{R}_1((u, r, v), G) = \mathcal{R}_1^{\text{directed}}(u|r, v, G) \text{ or } \mathcal{R}_1^{\text{directed}}(v|r, u, G).$$

Here, Sort represents the sorting function and Pos calculates the position of u in C_1'. $\mathcal{R}_1^{\text{directed}}(v|r, u, G)$ is computed symmetrically to $\mathcal{R}_1^{\text{directed}}(u|r, v, G)$, and the final choice between these two ranks as the output depends on which node the poisoner intends to manipulate.

Disease-agnostic importance ranking function $\mathcal{R}_2$. We first use PageRank[60] to obtain an importance score $\text{PR}(v)$ 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 $\text{PR}(v)$, PageRank iteratively updates $\text{PR}(v)$ using the following formula

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

where $\lambda \in [0, 1] \subseteq \mathbb{R}$ is the damping factor and $\mathcal{B}_v$ represents the set of nodes pointing to node v. Based on the learned importance score PR, we construct the ranking function $\mathcal{R}_2$ to calculate the global importance 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:

$$C_2' = \text{Sort}_{\text{key}=-\text{PR}(v_i)}(C_2),$$

$$\mathcal{R}_2(v, G) = \text{Pos}(v, C_2').$$

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 defence process based on these selected targets.

Disease-specific poisoning target selector $\mathcal{S}_1$. For the disease-specific scenario, we start from a representative drug set $\overline{\text{Drug}}$, 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 Unified Medical Language System database[61] 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 $\overline{\text{Drug}}$. Subsequently, for each $u_i \in \overline{\text{Drug}}$, we randomly choose five disease nodes $v \in V_{\text{disease}}$ as the target disease set $\text{Target}_{1,i}^{\text{node}}$. Then, we set the relation r to 'treatment' and construct the poisoning target link set for each u_i as $\text{Target}_{1,i} = \{(u_i, r, v) | v \in \text{Target}_{1,i}^{\text{node}}\}$. Finally, we merge all target link sets corresponding to u_i to obtain the poisoning target set in the disease-specific scenario as $\text{Target}_1 = \cup_{u_i \in \overline{\text{Drug}}} \text{Target}_{1,i}$.

Disease-agnostic poisoning target selector $\mathcal{S}_2$. 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: $\mathcal{R}_1^{\text{directed}}(u|r, v, \hat{G}) < \mathcal{R}_1^{\text{directed}}(u|r, v, G), (u, r, v) \in \text{Target}_1$ and $\mathcal{R}_2(v, \hat{G}) < \mathcal{R}_2(v, G), v \in \text{Target}_2$.

Selecting malicious links

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

Preparation of candidate malicious links. We first introduce how we prepare the candidate links for the disease-specific scenario. For each poisoning target $(u_t, r_t, v_t) \in \text{Target}_1$, we perform a breadth-first search centred 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 $\overline{C}_1^{\text{link}}$ as follows:

$$\overline{C}_1^{\text{link}} = \{(u, r, v) | u \in V_c, r \in R, v \in V_c\}$$

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

$$C_{2,\rightarrow}^{\text{link}} = \{(u, r, v) | u \in V, r \in R\},$$

$$C_{2,\leftarrow}^{\text{link}} = \{(v, r, u) | u \in V, r \in R\},$$

$$\overline{C}_2^{\text{link}} = C_{2,\rightarrow}^{\text{link}} \cup C_{2,\leftarrow}^{\text{link}}$$

Both $\overline{C}_1^{\text{link}}$ and $\overline{C}_2^{\text{link}}$ then undergo a rule-based filtering process to remove some inappropriate candidate links. For each $z \in \overline{C}_1^{\text{link}}$ *or* $\overline{C}_2^{\text{link}}$, 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}.

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 $s_1^{\text{poison}} : (z_m, z_t) \longmapsto \mathbb{R}$ measuring the impact of adding a malicious link $z_m \in C_1^{\text{link}}$ on the target link $z_t \in \text{Target}_1$. It would be time-consuming to retrain all KG embeddings. To address this, we adopt an estimate approach inspired by the Influence Function[48,62]. We first upweight z_m with a small weight ε and define the new optimal embeddings as $\hat{\theta}_{\varepsilon, z_m} = \text{argmin}_\theta \frac{1}{|E|} \sum_{z \in G} \mathcal{L}_{\text{emb}}(z, \theta) + \varepsilon \mathcal{L}_{\text{emb}}(z_m, \theta)$. We then calculate the impact of adding z_m on $\hat{\theta}$ as follows:

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

where $H_{\hat{\theta}}$ is the Hessian matrix, computed as $H_{\hat{\theta}} = \frac{1}{|E|} \sum_{z \in G} \nabla_\theta^2 \mathcal{L}_{\text{emb}}(z, \hat{\theta})$. 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 $s_1^{\text{poison}}(z_m, z_t)$ as

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

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

A higher $s_1^{\text{poison}}(z_m, z_t)$ 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:

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

Then, we consider the poisonous score in the disease-agnostic scenario. For each poisoning target $v \in \text{Target}_2$ and the corresponding candidate link $z_m = (u_m, r_m, v_m) \in C_2^{\text{link}}$, we follow the method described in PRAttack[63] to obtain the poisonous score $s_2^{\text{poison}}(z_m, v)$. When $z_m \in C_{2,\rightarrow}^{\text{link}}$, we set $s_2^{\text{poison}}(z_m, v) = \text{PR}(u_m)/(O(u_m) + 1)$. When $z_m \in C_{2,\leftarrow}^{\text{link}}$, we set $s_2^{\text{poison}}(z_m, v) = -\inf$. Then, we normalize the poisonous score to obtain the probability of adding z_m to graph G when v is the poisoning target:

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

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

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

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

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

$$p_1^{\text{overall}}(z_m | z_t) = p_1^{\text{poison}}(z_m | z_t) \times p_1^{\text{conceal}}(z_m | z_t),$$

$$p_2^{\text{overall}}(z_m | v) = p_2^{\text{poison}}(z_m | v) \times p_2^{\text{conceal}}(z_m | v).$$

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:

$$p_1^{\text{overall},\mathcal{D}}(z_m | z_t) = p_1^{\text{overall}}(z_m | z_t), \text{when } \mathcal{D}(z_m) = \text{True},$$

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

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

Malicious abstract generator

Instead of directly adding links to the KG, 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.

Construct sentence template using the malicious link. During the construction of the KG using the extractor $\mathcal{E}$, we gather and form $S_r = \{s_{r,i}\}$, where $s_{r,i}$ represent the ith 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 $\bar{s}_{r_m} = \langle t_1^{r_m}, \ldots, t_n^{r_m} \rangle \in \bar{S}_{r_m}$, we calculate its perplexity as

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

Here, $p_{\text{LM}}(t_i^{r_m} | t_1^{r_m}, t_2^{r_m}, \ldots, t_{i-1}^{r_m})$ represents the probability that the ith token is $t_i^{r_m}$ given the previous tokens $t_1^{r_m}, t_2^{r_m}, \ldots, t_{i-1}^{r_m}$, 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[64] as the language model. We select the sentence with the lowest perplexity $\mathcal{L}_{\text{LM}}(\bar{s}_{r_m})$ from $\bar{S}_{r_m}$ as the sentence template $s_{z_m} = \langle t_1^{z_m}, \ldots, t_n^{z_m} \rangle$ for the malicious link z_m.

Generate fluent paper from sentence template using ChatGPT. We utilize the ChatGPT 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. When calling the ChatGPT API, we empirically set the max_tokens to 3,000, the temperature to 0.5 and the frequency_penalty to 0.5.

Fine-tuning with BioBART for more domain-specific and controllable generation. Directly using the output of ChatGPT as ultimate generation encounters two limitations. First, 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. Second, 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[65], an open-source natural language generation model specialized in the biomedical domain, to fine-tune the generation from ChatGPT. Please refer to the Supplementary Information for fine-tuning details.

Evaluation of the generated papers

We evaluate the generated papers based on two aspects: poisoning effectiveness and text quality. For the poisoning-effectiveness evaluation, we rerun the entire KG construction and reasoning system on the mixture of each malicious paper and the original database. For the text-quality evaluation, following the G-Eval[66], we use GPT-4 to score the papers on writing fluency, context coherence and scientific faithfulness. Additionally, we employ manual methods for a more thorough evaluation. For more detailed implementation, please refer to the Supplementary Information.

Comparison methods

We compare eight poisoning methods to demonstrate the effectiveness of Scorpius, including the most powerful LLMs, GPT-3.5 and GPT-4, along with their enhanced version using retrieval-augmented generation techniques[67,68]. For implementation details of the comparison methods, please refer to the Supplementary Information.

Defender

We develop two defenders to investigate how to mitigate potential poisoning in a KG reasoning system. One is the link-faithfulness defender, which filters out untrustworthy papers by assessing the validity of the extracted links. The other is text-faithfulness defender, which uses GPT-4 to directly filter out harmful papers. The link-faithfulness defender is our default defender. For more detailed implementation, please refer to the Supplementary Information.

Comparing poisoning effectiveness on Medline and bioRxiv

To evaluate the impact of using an unreviewed database on the effectiveness of poisoning, we conduct our poisoning experiments from scratch using the bioRxiv database. We construct a new KG by employing the extractor $\mathcal{E}$ and incorporating papers from bioRxiv dated between 1 January 2022 and 1 January 2023. This results in a KG consisting of 15,142 nodes. We then randomly remove nodes from the complete KG built from Medline, prioritizing the deletion of nodes not included in bioRxiv, until the number of nodes in both KGs is equal. Subsequently, we perform disease-specific and disease-agnostic poisoning on the bioRxiv and Medline KGs and compare their performance. The results are shown in Supplementary Figs. 7 and 8.

Investigating the impact of KG size

We keep disease nodes involved in disease-specific targets and all drug nodes unchanged and then iteratively remove the nodes with the fewest corresponding links from the original KG to reduce the KG size. We test the poisoning effectiveness of Insertion and Scorpius in KGs of sizes 20,000, 30,000, 40,000, 50,000 and 60,000 (original). In the disease-specific scenario, for a poisoning target link $(\text{drug}_i, r_i, \text{disease}_i)$ in Target_1, let its rank in the original KG be r_1^{original}. In the smaller KGs, we keep disease_i and r_i unchanged and find a new drug, drug_j, such that the rank of $(\text{drug}_j, r_i, \text{disease}_i)$ equals r_1^{original}. We then consider $(\text{drug}_j, r_i, \text{disease}_i)$ as the new poisoning target for smaller KGs. In the disease-agnostic scenario, we adopt the same approach to adjust Target_2 for different KGs, ensuring that the target's rank before poisoning remains consistent across different KG sizes. Finally, we directly compare the rankings after poisoning in these two scenarios (Supplementary Figs. 9 and 10).

Using PrimeKG to enhance Medline KG

We utilize an expansive and heterogeneous biomedical KG to enhance the KG we build from Medline and assess the effectiveness of poisoning on the enhanced KG. We adopt PrimeKG[69] as the additional KG, encompassing over 120,000 biomedical nodes and ten node types

(Supplementary Fig. 11a–d). We iteratively add links from PrimeKG to Medline KG, prioritizing links that include shared nodes between the two KGs. We evaluate the poisoning effectiveness on KGs of sizes 60,000 (original), 80,000, 100,000 and 120,000. We then adjust the promoted drugs in poisoning targets Target_1 and Target_2 for different KGs to ensure that the rank of the targets before poisoning remains consistent across different KG sizes. We then compare the after-poisoning rankings in both disease-specific (Supplementary Fig. 11e–h) and disease-agnostic (Supplementary Fig. 11i–l) scenarios.

Investigating the impact of drug rarity

We assess the impact of the rarity of the promoted drugs on poisoning effectiveness. We rank all drugs from least to most frequent in Medline and select drugs from the front, middle and end of the list as rare, medium and prevalent drugs, respectively. The results are presented in Supplementary Figs. 12 and 13.

Promoting new drugs

We evaluate the impact of new promoted drugs on poisoning effectiveness. For each individual promoted drug u, we delete the links associated with u from the original KG, forming a new KG G_u. This operation makes drug u appear as an entirely new node in G_u, as its relationships with other nodes are unknown. Because we use negative sampling techniques[41,42,62] during the optimization of $\mathcal{L}_{\text{emb}}$, drug u can acquire meaningful embeddings during this optimization, which allows for effective poisoning and evaluation. We compare the impact of treating the drug u as a new node at different defence levels; the rankings after poisoning are shown in Supplementary Fig. 14.

Data availability

The Medline KG is available at https://zenodo.org/records/1035500 (ref. 70). The PubTator database is available at https://ftp.ncbi.nlm.nih.gov/pub/lu/PubTatorCentral/. The Unified Medical Language System database we used to identify 'Pharmacologic Substance' and 'Clinical Drug' is available at https://documentation.uts.nlm.nih.gov/rest/home.html. The bioRxiv database is available at https://api.biorxiv.org/. The PrimeKG database we used for enhancing Medline KG is available at https://github.com/mims-harvard/PrimeKG. All processed data can be directly downloaded from our GitHub project: https://github.com/yjwtheonly/Scorpius.

Code availability

Scorpius code is available at https://github.com/yjwtheonly/Scorpius ref. 71. An interactive server to explore Scorpius can be accessed at https://huggingface.co/spaces/yjwtheonly/Scorpius_HF.

References

1. Roberts, R. J. PubMed Central: the GenBank of the published literature. *Proc. Natl Acad. Sci. USA* **98**, 381–382 (2001).
2. Canese, K. & Weis, S. in *The NCBI Handbook* 2nd edn (eds Beck, J. et al.) Ch. 3 (NCBI, 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* **37**, 1739–1746 (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. *J. Biomed. Inform.* **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**, 9841548 (2022).
11. Ouyang, L. et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems* (eds Koyejo, S. et al.) 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. In *Proc. 58th Annual Meeting of the Association for Computational Linguistics* (eds Jurafsky, D. et al.) 7871–7880 (ACL, 2020).
15. OpenAI et al. GPT-4 technical report. Preprint at https://arxiv.org/abs/2303.08774 (2023).
16. Thoppilan, R. et al. LaMDA: language models for dialog applications. Preprint at https://arxiv.org/abs/2201.08239 (2022).
17. Surameery, N. M. S. & Shakor, M. Y. Use Chat GPT to solve programming bugs. *Int. J. Inform. Technol. Comput. Eng.* **3**, 17–22 (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**, 887 (2023).
21. Park, J. S. et al. Generative agents: interactive simulacra of human behavior. In *Proc. 36th Annual ACM Symposium on User Interface Software and Technology* (eds Follmer, S. et al.) 2:1–2:22 (ACM, 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. Preprint at https://www.arxiv.org/abs/2004.10706v4 (2020).
28. Ahamed, S. & Samad, M. Information mining for COVID-19 research from a large volume of scientific literature. Preprint at https://arxiv.org/abs/2004.02085 (2020).
29. Pestryakova, S. et al. CovidPubGraph: a FAIR knowledge graph of COVID-19 publications. *Sci. 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 *Proc. 19th International Semantic Web Conference* (eds Pan, J. Z. et al.) 294–310 (Springer, 2020).
32. Gehrmann, S., Strobelt, H. & Rush, A. M. GLTR: statistical detection and visualization of generated text. In *Proc. 57th Conference of the Association for Computational Linguistics* (eds Korhonen, A. et al.) 111–116 (ACL, 2019).

33. Jawahar, G., Abdul-Mageed, M. & Lakshmanan, L. V. S. Automatic detection of machine generated text: a critical survey. In *Proc. 28th International Conference on Computational Linguistics* (eds Scott, D. et al.) 2296–2309 (ICCL, 2020).
34. Wang, W. & Feng, A. Self-information loss compensation learning for machine-generated text detection. *Math. Probl. Eng.* **2021**, 6669468 (2021).
35. Mitchell, E., Lee, Y., Khazatsky, A., Manning, C. D. & Finn, C. DetectGPT: zero-shot machine-generated text detection using probability curvature. In *Proc. International Conference on Machine Learning* (eds Krause, A. et al.) 24950–24962 (PMLR, 2023).
36. Meyer zu Eissen, S. & Stein, B. Intrinsic plagiarism detection. In *Proc. Advances in Information Retrieval* (eds Lalmas, M. et al.) 565–569 (Springer Berlin Heidelberg, 2006).
37. Lukashenko, R., Graudina, V. & Grundspenkis, J. Computer-based plagiarism detection methods and tools: an overview. In *Proc. International Conference on Computer Systems and Technologies* https://doi.org/10.1145/1330598.13306 (ACM, 2007).
38. Meyer zu Eissen, S., Stein, B. & Kulig, M. Plagiarism detection without reference collections. In *Advances in Data Analysis* (eds Decker, R. & Lenz, H. J.) 359–366 (Springer Berlin Heidelberg, 2007).
39. Donaldson, J. L., Lancaster, A.-M. & Sposato, P. H. A plagiarism detection system. In *Proc. 12th SIGCSE Technical Symposium on Computer Science Education* https://doi.org/10.1145/953049.800955 (ACM, 1981).
40. Yang, B., Yih, W.-T., He, X., Gao, J. & Deng, L. Embedding entities and relations for learning and inference in knowledge bases. In *3rd International Conference on Learning Representations* (ICLR, 2015).
41. Dettmers, T., Minervini, P., Stenetorp, P. & Riedel, S. Convolutional 2D knowledge graph embeddings. In *Proc. AAAI Conference on Artificial Intelligence* 1811–1818 (AAAI, 2018).
42. Trouillon, T., Welbl, J., Riedel, S., Gaussier, E. & Bouchard, G. Complex embeddings for simple link prediction. In *Proc. 33rd International Conference on Machine Learning* (eds. Balcan, M. F. & Weinberger, K. Q.) 2071–2080 (PMLR, 2016).
43. Lu, Y. et al. Unified structure generation for universal information extraction. In *Proc. 60th Annual Meeting of the Association for Computational Linguistics* (eds Muresan, S. et al.) 5755–5772 (ACL, 2022).
44. Li, X. et al. TDEER: an efficient translating decoding schema for joint extraction of entities and relations. In *Proc. Conference on Empirical Methods in Natural Language Processing* (eds Moens, M.-F. et al.) 8055–8064 (ACL, 2021).
45. Yamada, I., Asai, A., Shindo, H., Takeda, H. & Matsumoto, Y. LUKE: deep contextualized entity representations with entity-aware self-attention. In *Proc. 2020 Conference on Empirical Methods in Natural Language Processing* (eds Webber, B. et al.) 6442–6454 (ACL, 2020).
46. Bhardwaj, P., Kelleher, J., Costabello, L. & O'Sullivan, D. Poisoning knowledge graph embeddings via relation inference patterns. In *Proc. 59th Annual Meeting of the Association for Computational Linguistics* (eds Zong, C. et al.) 1875–1888 (ACL 2021).
47. Pezeshkpour, P., Tian, Y. & Singh, S. Investigating robustness and interpretability of link prediction via adversarial modifications. In *Proc. 2019 Conference of the North American Chapter of the Association for Computational Linguistics* (eds Burstein, J. et al.) 3336–3347 (ACL, 2019).
48. Bhardwaj, P., Kelleher, J., Costabello, L. & O'Sullivan, D. Adversarial attacks on knowledge graph embeddings via instance attribution methods. In *Proc. 2021 Conference on Empirical Methods in Natural Language Processing* (eds Moens, M.-F. et al.) 8225–8239 (ACL, 2021).
49. Li, Q., Wang, Z. & Li, Z. PAGCL: an unsupervised graph poisoned attack for graph contrastive learning model. *Future Gener. Comput. Syst.* **149**, 240–249 (2023).
50. You, X. et al. MaSS: model-agnostic, semantic and stealthy data poisoning attack on knowledge graph embedding. In *Proc. ACM Web Conference* (eds Ding, Y et al.) 2000–2010 (ACM, 2023).
51. Betz, P., Meilicke, C. & Stuckenschmidt, H. Adversarial explanations for knowledge graph embeddings. In *Proc. 31st International Joint Conference on Artificial Intelligence* (ed. De Raedt, L.) 2820–2826 (IJCAIO, 2022).
52. Zhang, H. et al. Data poisoning attack against knowledge graph embedding. In *Proc. 28th International Joint Conference on Artificial Intelligence* (eds Kraus, S.) 4853–4859 (IJCAI, 2019).
53. Gao, Z., Ding, P. & Xu, R. K. G.- Predict: a knowledge graph computational framework for drug repurposing. *J. Biomed. Inform.* **132**, 104133 (2022).
54. Vashishth, S., Sanyal, S., Nitin, V., Agrawal, N. & Talukdar, P. InteractE: improving convolution-based knowledge graph embeddings by increasing feature interactions. In *Proc. AAAI Conference on Artificial Intelligence* 3009–3016 (AAAI, 2020).
55. Han, S. et al. Standigm ASK™: knowledge graph and artificial intelligence platform applied to target discovery in idiopathic pulmonary fibrosis. *Brief. Bioinform.* **25**, bbae035 (2024).
56. Zheng, S. et al. PharmKG: a dedicated knowledge graph benchmark for bomedical data mining. *Brief. Bioinform.* **22**, bbaa344 (2021).
57. Wei, C.-H., Kao, H.-Y. & Lu, Z. PubTator: a web-based text mining tool for assisting biocuration. *Nucleic Acids Res.* **41**, W518–W522 (2013).
58. Greenhalgh, T. How to read a paper. The Medline database. *Brit. Med. J.* **315**, 180–183 (1997).
59. de Marneffe, M.-C. & Manning, C. D. The Stanford typed dependencies representation. In *Proc. Workshop on Cross-Framework and Cross-Domain Parser Evaluation* (eds Bos, J. et al.) 1–8 (ACL, 2008); https://doi.org/10.3115/1608858.1608859
60. Page, L., Brin, S., Motwani, R. & Winograd, T. The PageRank citation ranking: bringing order to the web. http://ilpubs.stanford.edu:8090/422/ (Stanford InfoLab, 1999).
61. Bodenreider, O. The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Res.* **32**, D267–D270 (2004).
62. Koh, P. W. & Liang, P. Understanding black-box predictions via influence functions. In *Proc. 34th International Conference on Machine Learning* (eds Precup, D. & Teh, Y. W.) 1885–1894 (PMLR, 2017).
63. Bianchini, M., Gori, M. & Scarselli, F. Inside PageRank. *ACM Trans. Internet Technol.* **5**, 92–128 (2005).
64. Luo, R. et al. BioGPT: generative pre-trained transformer for biomedical text generation and mining. *Brief. Bioinform.* **23**, bbac409 (2022).
65. Yuan, H. et al. BioBART: Pretraining and evaluation of a biomedical generative language model. In *Proc. 21st Workshop on Biomedical Language Processing* (eds Demner-Fushman, D. et al.) 97–109 (ACL, 2022).
66. Liu, Y. et al. G-Eval: NLG Evaluation using GPT-4 with better human alignment. In *Proc. 2023 Conference on Empirical Methods in Natural Language Processing* (eds Bouamor, H. et al.) 2511–2522 (ACL, 2023).
67. Chen, J., Lin, H., Han, X. & Sun, L. Benchmarking large language models in retrieval-augmented generation. In *Proc. AAAI Conference on Artificial Intelligence* (eds Woodridge, M. et al.) 17754–17762 (2024).
68. Ranjit, M., et al. Retrieval augmented chest X-ray report generation using OpenAI GPT models. In *Proc. 8th Machine Learning for Healthcare Conference* (eds Deshpande, K. et al.) 650–666 (PMLR, 2023).

69. Chandak, P., Huang, K. & Zitnik, M. Building a knowledge graph to enable precision medicine. *Sci. Data* **10**, 67 (2023).
70. Percha, B. & Altman, R. A global network of biomedical relationships derived from text. *Zenodo* zenodo.org/records/1035500 (2017).
71. Junwei, Y. et al. Poisoning medical knowledge using large language models v.1.0.1. *Zenodo* https://doi.org/10.5281/zenodo.13191322 (2024).
72. Lee, J. et al. BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* **36**, 1234–1240 (2020).

Acknowledgements

We would like to extend our thanks to B. Feng from the International Digital Economy Academy for his assistance in revising our manuscript. We are also grateful to Z. Wu and K. Zheng from Peking University for their insightful discussion on technical implementation. J.Y., S.M., Zequn Liu, W.J., L.L. and M.Z. are partially supported by the National Key Research and Development Programme of China with grant no. 2023YFC3341203 as well as the National Natural Science Foundation of China with grant nos. 62276002 and 62306014.

Author contributions

J.Y., T.C., M.Z. and S.W. contributed to the conception and design of the whole work. J.Y., S.M., Zequn Liu, Z.X. and S.W. contributed to the data curation. J.Y., S.M., Z.L. and Z.X. contributed to the technical implementation. J.Y., H.X., Z.X., M.Z. and S.W. contributed to the writing. All authors contributed to the drafting and revision of the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42256-024-00899-3.

Correspondence and requests for materials should be addressed to Zhiping Xiao, Ming Zhang or Sheng Wang.

Peer review information *Nature Machine Intelligence* thanks Ping Zhang and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.