Identifying Effective Strategies to Mitigate Misinformation in Knowledge Graph Data Voids

Reem Hazim
Computer Science, NYUAD
rh3015@nyu.edu

Advised by: Azza Abouzied

ABSTRACT

Misinformation has recently garnered significant attention as a growing threat to numerous aspects of society such as political systems, healthcare systems, and economies. The recent rise in misinformation and disinformation campaigns is driven by the public's growing reliance on digital platforms for news consumption [25].

In response to this challenge, significant research has been conducted to identify misinformation on social media platforms. Recently, however, researchers have turned their attention to the role of search engines in facilitating the spread of misinformation by leading unsuspecting users to sites that host questionable content, particularly when they search for a term that has limited credible content [10]. These search topics, known as "data voids" [10], have been studied in the context of search engines but have yet to be examined in knowledge graphs– critical structures for question-answering tasks in search engines. This paper investigates data voids in the context of knowledge graphs and proposes strategies to mitigate misinformation in such knowledge structures. We identify four strategies (greedy, approximate greedy, neighbor, and multi-objective greedy) that can be used by a mitigator to counter a disinformation campaign by promoting a true fact in the knowledge graph.

KEYWORDS

misinformation, disinformation, data voids, knowledge graphs, knowledge graph embeddings, search engines, link prediction, KG poisoning

Reference Format:
Reem Hazim. 2023. Identifying Effective Strategies to Mitigate Misinformation in Knowledge Graph Data Voids. In *NYUAD Capstone Project 2 Reports, Fall 2023, Abu Dhabi, UAE.* 15 pages.

1 INTRODUCTION

Misinformation has recently garnered significant attention as a growing threat to numerous aspects of society such as political systems, healthcare systems, and economies. The recent rise in misinformation and disinformation campaigns is driven by the public's growing reliance on digital platforms for news consumption [25]. These platforms have also facilitated the sharing of large amounts of information at a low cost, enabling malicious actors to scale up their operations. Some examples of recent disinformation threats with devastating impacts include the misinformation campaigns that likely influenced the British Brexit referendum [17], Russia's interference in the 2016 U.S. presidential elections, and the growing vaccine misinformation that has hindered some countries' recovery efforts from COVID-19 [2].

In response to the growing influence of misinformation on public discourse, many researchers have attempted to detect misinformation and mitigate its effects through a variety of methods and tools [12, 17]. Most of these studies focus on social media platforms where disinformers can easily reach a large audience. However, recently, researchers have been looking into the role of search engines in facilitating misinformation and leading unsuspecting search users to sites that host questionable content [10]. In particular, Boyd and Golebiewski identify a phenomenon where a lack of high-quality and authoritative content about a search query can cause search engines to return low-quality and problematic content. They call this phenomenon a "data void". Disinformers and malicious actors can manipulate these data voids to spread false information in certain scenarios [10].

For example, "Sutherlands Springs" is a search term that refers to a small town in the United States. The town received little to no public attention until a mass shooting occurred that propelled the term into mainstream media. Given the low amount of web content surrounding this small town, manipulators were able to shape search engine results by posting problematic content on a variety of platforms [10].

While studies on data voids have examined the phenomenon in search engines and on social media [10, 14, 19], there have not been any studies about the role of data voids in knowledge graphs. In fact, despite the prevalent use of knowledge graphs in many critical applications, there has been limited research about their susceptibility to adversarial attacks [7, 21, 31]. Therefore, this study examines data voids in the context of knowledge graphs and identifies strategies to mitigate misinformation in knowledge graphs.

2 RELATED WORK

2.1 Data Voids

"Data void" is a recent term coined by Michael Golebiewski and Danah Boyd in a research report titled, "Data Voids: Where Missing Data Can Easily Be Exploited" [10]. The term describes the phenomenon where search engines have very limited data about a search term or query. Therefore, when a user enters such a query, the search engine will return low-quality or problematic results because there is no credible content on that topic. The significance of data voids is that they can be intentionally exploited by malicious actors to spread misinformation and problematic content through search engines [10].

Data voids are usually harmless until the number of searches for a term suddenly increase. For instance, when a breaking news incident occurs, it may trigger a flood of searches for a new search query. Since it takes time for trusted content producers to publish high-quality content about a breaking news incident, bad actors can take advantage of these data voids by producing fake content that will likely surface in users' search results [10].

Boyd and Golebiewski argue that data voids cannot be completely solved. Instead, they consider them a security vulnerability that search engines must continously manage and respond to [10]. One way to mitigate data voids is to support content creators in producing high-quality content to fill these voids. Google, one of the world's most popular search engines [1], attempted to identify and highlight content gaps on the web through a recent beta product called Question Hub. Launched in 2019, Question Hub allows users to inform Google when their search query does not yield relevant results. Google collects these queries, sorts them by topic, and shares them with publishers and content creators [4]. However, Google shut down the product in January 2023 after concluding beta testing, and it is unclear if the product will be discontinued or if it will be merged with existing tools, such as Google for Creators [1] [24].

[1]https://creators.google/en-us/

Given the novelty of the term, there have been very few research studies about data voids. Norocel and Lewandowski analyzed data voids in Google searches for right-extremist terms after the 2015 humanitarian crisis in Europe. To create the search queries, the authors combined municipality names in Sweden and Germany (two countries that experienced significant growth in extreme-right voices during the 2015 crisis) with three types of keywords: mainstream keywords (e.g. refugee crisis), radical-right keywords that may be known to an ordinary user (e.g. asylum parasite) and extreme-right keywords that are rarely known to an ordinary user (e.g. rapefugees). The authors note that, while search engines such as Google regularly update their algorithms to decreases the ranking of extreme-right content, localized searches that contain the name of specific municipalities remain highly susceptible to data voids because they yield fewer results, allowing radical content to become more visible. Therefore, they add the names of specific municipalities to their search queries to surface data voids. After running the queries through Google, they find that the more extremist queries returned pages from a higher concentration of sources than the mainstream queries, which returned more diverse sources. In particular, mainstream queries returned results from a mixture of mainstream sources like news websites, government websites, and social media posts or blogs, whereas more extreme queries return more extreme-right news websites due to the lack of mainstream content containing these keywords [19].

Copeland et al. also study existing data voids, but they focus on contextualizing them using a harms framework and mapping the timeline of 'breaking news' data voids using data from Google search trends, media coverage and Wikipedia page histories. Their study reveals two main states in the life cycle of a data void: a "long tail" that slowly accumulates disinformation over time without receiving much public attention, and an "interest spike" that brings attention to the data void when a breaking news incident occurs, after which the void is quickly filled with content from authoritative sources. In addition, they find that mainstream media sources play a role in amplifying disinformation by prompting people to search for queries associated with data voids. They also argue that the 'long tails' of data voids are just as harmful as the 'interest spikes' because they allow a community of disinformers to grow and develop under the public radar [11].

While both of these studies focus on data voids in Google, data voids are not exclusive to search engines – they may also exist on social media platforms. Many journalists use Twitter and Facebook to search for information about breaking news [10], and young users have been increasingly relying on social media platforms such as Facebook and Tiktok as their main source of news [12]. Since these platforms only have a fraction of the data that search engines have access to, they are even more vulnerable to data voids.

To help independent journalists visualize and understand data voids on Facebook, Flores-Saviaga et al. built a collaborative, machine learning-powered tool that can help journalists fill data voids with authoritative content. The platform takes a list of Facebook pages and groups as input from the user and displays a visual summary of relevant metrics such as the distribution of topics across posts, the credibility of post authors and their political leaning, and the level of engagement with each topic. The tool also highlights potential data voids on the given pages, but the authors do not specify the mechanism they use to identify data voids. In addition, they provide collaboration tools for journalists such as a chat functionality and document sharing.

2.2 Knowledge Graphs and Knowledge Graph Embeddings

A Knowledge graph (KG) is a type of labeled and directed graph that represents real-world information in a structured manner. Nodes in knowledge graphs usually represent entities such as people, organizations or places, and labels represent relations between the entities [22]. More formally KGs can be defined a collection of triples (facts), $\mathcal{KG} = \{(h, r, t) \mid h, t \in \mathcal{E}\ and\ \mathrm{r} \in \mathcal{R}\}$, where $\mathcal{E}$ is a set of entities, $\mathcal{R}$ is a set of relations, and h, r, t are the head, relation and tail in a triple [7, 22].

Knowledge Graphs are very powerful because of their ability to represent structured information in a format that is readable by machines; therefore, they are used extensively in many areas such as healthcare, finance [20], and question answering in search engines [3]. Examples of KGs include FreeBase [8], WikiData [29], DBPedia [5], Yago [26] and the Google Knowledge Graph[2].

Because the information stored in KGs is usually incomplete, KGs are normally used under the Open World Assumption (OWA). The OWA holds that if a fact is missing from a KG, it may either be incorrect or just missing. This is in contrast with the closed world assumption (CWA), where a fact not present in the KG is considered incorrect [15]. The incompleteness of KGs motivates a task known as *Knowledge Graph Completion* where the goal is to identify new facts that can be added to the KG. One way to do so is by looking at existing facts in the KG and using them to infer missing facts. This approach is known as *Link Prediction* (LP). In practice, link prediction is done by predicting the missing entity in *(h, r, ?)* (tail prediction) or *(?, r, t)* (head prediction). The missing entity being predicted is known as the *target* entity [22].

Much research has been done on link prediction, and while there exists a variety of approaches to the task, most link prediction models nowadays use machine learning to learn low-dimensional vector representations of entities and relations. These vector representations are known as Knowledge Graph Embeddings (KGE), and they aim to capture semantic relationships between entities and relations [22]. Following standard convention, we use *italics* to represent an element in a knowledge graph (e.g. r) and **bold** to represent the embedding of that element ($\mathbf{r}$).

KGEs are learned by training on a KG dataset and optimizing the embeddings to maximize the likelihood of true facts and minimize it for false facts. Negative triples are false facts that are created during the training process by corrupting positive triples and replacing their head or tail with a random entity. To predict the missing head entity in a triple *(?, r, t)*, a scoring function $f(\mathbf{e}, \mathbf{r}, \mathbf{t})$ is applied to the triple for every possible entity e. The entities are then ranked according to their scores, and the entity with the highest score is the entity predicted by the model. There are several standard metrics used to evaluate the performance of a KGE model, and most of them use the ranks of the missing entities in a testing set. Mean Rank (MR) is computed as the average of all the ranks of the correct entity; the lower the the MR, the better the model is performing. Mean Reciprocal Rank (MRR) is another popular metric, defined as the average of the inverse of the ranks. *Hits@K* refers to the proportion of ranks less *K*. For example, *Hits@1* refers to the proportion of facts where the LP model gave the correct entity a ranking of 1. *Hits@10* refers to the proportion of facts where the correct entity was ranked among the top 10 entities [22].

Many powerful KGE models have been developed over the past few years. Rossi et al. carry out a comparative analysis between KGE models, and they divide them into 3 categories based on architecture: Tensor Decomposition models, Geometric models and Deep Learning models. Tensor decomposition models such as ComplEx [28], DistMult [30] and SimplE [16] use 3rd-order tensors to represent interactions between elements in a KG. The tensors are decomposed into vectors to produce the embeddings for entities and relations, and facts are scored by combining the embeddings of the facts' elements. Geometric models like TransE [9] and RotatE [27] consider relations as geometric transformations in the embedding space. In particular, they consider the tail embedding to be a spatial transformation of the head embedding that depends on the value of the relation embedding ($\mathbf{t} = \tau(\mathbf{h},\mathbf{r})$). These models score facts by calculating the distance between the tail entity and $\tau(\mathbf{h}, \mathbf{r})$. Finally, Deep Learning models like ConvE [13] use deep neural networks to learn embeddings and perform LP [22].

Rossi et al. compare several KGE models of all three types in terms of effectiveness and efficiency (time required for training and prediction). Notably, they find that ComplEx, HAKE and InteractE are the fastest models in their categories.

[2]https://developers.google.com/knowledge-graph/

They also find that different models have very different predictive performance when it comes to different relations types such as symmetric, transitive, and reflective relations. They use AnyBURL [18], a rule-based system that does not use embeddings, as a baseline and find that it outperforms most embedding-based models, performing very similarly to ComplEx, which is the best-performing KGE model [22].

Similarly, Huynh and Papotti build a novel benchmark for comparing fact-checking algorithms (which includes both embedding-based and non-embedding-based algorithms). Their data generator takes as input a predicate (relation) and produces training and testing data of varying complexity, based on the following properties of data: popularity, transparency (ambiguity of false facts), homogeneity (semantic similarity between true and false facts), and functionality (1-to-1 relations). They find that fact-checking algorithms perform better on more popular and functional facts and in scenarios with higher homoegeneity and transparency [15].

Another domain in knowledge graph research that can be beneficial to our research problem is link prediction explainability. Studies in this field try to understand why link prediction models make specific predictions, typically by identifying the edges in the graph that have the most influence on a prediction task. Rossi et al. build Kelpie, a post-hoc link prediction interpretability tool that takes a prediction as input and identifies the facts in the KG that enable it [23]. Given an input fact *x = (h, r, t)*, Kelpie identifies two types of explanations. Necessary explanations are the smallest set of facts needed to predict *x*. Sufficient explanations are the smallest set of training facts containing *h* that, if applied to another fact *x' = (h', r', t')* by replacing *h* with *h'*, can cause the model to switch its prediction of the tail from *t'* to *t*. Kelpie finds these explanations by first calculating a *promisingness* score and only keeping the most promising triples. Then, they calculate the effect of each fact on the model by removing/adding the fact and retraining the model efficiently using a method called post-training. Finally, they return the top relevant facts [23]. In this paper, we use Kelpie to produce the necessary explanations for two 'competing' facts and we simulate a scenario where a disinformer and a mitigator are filling up a data void by adding explanations for their respective facts.

2.3 KGE Data Poisoning Attacks

Although KGs are being used extensively, they are not completely robust to attacks. FreeBase, for example, is built on user-submitted wiki entries. If a model is trained on an unreliable knowledge base, it may learn biased embeddings that negatively impact downstream applications. A recent line of research has been investigating the security vulnerabilities of KGE models. In particular, a few studies have looked at possible adversarial attacks against KGE models [7, 21, 31]. These studies have focused on *data poisoning attacks* where an attacker has access to the training data of an algorithm and can manipulate a small portion of the data to influence the way the trained model behaves at prediction time [31]. The following studies propose strategies to *promote* or *degrade* the plausibility of a missing *target* fact by adding or removing facts to the training set of the model. These fact additions or deletions are known as *perturbations* or *adversarial edits* in the KGE data poisoning literature.

Zhang et al. propose a collection of direct (directly corrupting the embeddings of the targeted fact) and indirect (using proxy entities to influence the target fact) data poisoning strategies against KGEs [31]. They assume that the attacker has a limited budget of M perturbations (fact additions or deletions) and focus mainly on degrading target facts that do not exist in the training set. To perform a direct attack against a target fact, they identify the optimal direction to shift its entity's embeddings and rank all the possible perturbations by how much they shift the target fact in the desired direction. They then perform the top M ranking perturbations [31]. For the indirect attack, they search the space of all the possible K-hop paths from the target fact, find the best P paths that propagate the influence from a proxy entity to the target entity, then find the best proxy entities to delete or add in those paths. They evaluate their strategies on the FB15k (subset of FreeBase) and WN18 (subset of WordNet) databases using the TransE, TransR and RESCAL KGE models. They find that both of their attack strategies are effective at degrading the target facts, with direct strategies being more effective than indirect strategies [31].

Zhang et al. use a mathematical approach to calculate the impact of all possible perturbation candidates on the target fact. This approach has a very large search space and may run into scalability issues, so Bhardwaj et al. seek to reduce the candidate search space by relying on relation inference patterns like symmetry, inversion and composition to improve the model's plausibility score on a set of decoy facts [7]. Given a target entity *(h, r, t)*, they first use the additive and multiplicative properties of geometric models to identify suitable adversarial (symmetric, inverse and composition) relations r_i to the target relation. Then, they find the optimal decoy triple, which is either an object-side *(s, r, o')* or subject-side *(s', r, o)* negative of the target triple. Finally, they create the adversarial triple by combining the decoy entity with the adversarial relation, and they add it to the training set. They show that their strategy is more effective than Zhang et al.'s, and their symmetry attack in particular performs better than the rest [7].

Pezeshkpour et al. rely on another approach to solve the issue of the combinatorial search space for perturbations

[21]. Similarly to Zhang et al., they use gradient optimization to identify the optimal direction in which to shift the embeddings of the target fact. However, instead of searching the space of all perturbation candidates, they use the gradient to calculate the embeddings of the optimal perturbation fact, and then they train an *inverter* network to decode the embeddings back to graph elements that can then be added to the graph. Their additive attacks can reduce the hits@1 performance of DistMult and ConvE by 27.3% and 50.7% respectively. They also use their adversarial facts to explain model predictions by identifying the most influential neighbors for a predicted link [21]. Table 1 summarizes the comparison between the three KGE data poisoning studies.

3 METHODOLOGY

Let us assume that we have a given KG k, and that there are two agents, a *disinformer* who wants to promote the plausibility of a *false* triple (h, r, t_f), and a *mitigator* who wants to promote the plausibility of a *true* triple (h, r, t_t). We call the true triple and the false triple, *target* facts. Note that the mitigator and the disinformer are competing to promote alternative target facts, in the sense that if a user runs the link prediction algorithm on *(h, r, ?)*, the likelihood of the two tails t_t and t_f will be compared as part of the prediction process, and one tail will be ranked higher than the other. Each agent's goal, therefore, is to ensure that their fact is ranked more highly than the other agent's fact. We further assume that there is a data void surrounding both target facts, such that both facts are missing from k, and there are no facts relevant to either target fact in k. Each agent is trying to to increase the link prediction ranking of their target fact in this data void. Neither the mitigator nor the disinformer can add their target facts directly to k due to prohibitive cost. Therefore, they have to find strategies to increase the rank of their respective target facts by adding *proxy* facts to k. We call the set of proxy facts for each agent the *budget* of that agent. The disinformer has budget $budget_d$ and the mitigator has budget $budget_m$. Each proxy fact has a specific cost associated with it, and the most important and impactful facts are the most expensive to add to k.

Our goal is to simulate an experiment where the disinformer and mitigator take turns adding proxy facts from their budget to k by following specific strategies. We would like to identify the best strategy for the mitigator should use given the strategy used by the disinformer. We consider two main metrics to measure a strategy's efficacy: the Area Under the Curve of the rankings for each round of the simulation, and the cumulative cost associated with the strategy.

4 EXPERIMENT

This section describes the process of setting up and running the experiment.

4.1 Model, Data and Settings

Given Rossi et al.'s finding that ComplEx [28] is the best-performing KGE model, we run all our experiments on this model. We specifically use the PyTorch implementation of ComplEx provided by Rossi et al., and we also rely on the hyperparameters they recommend. A table of all the hyperparameters and the performance of each trained model using these hyperparameters can be found on their Github repository[3]. We set the random seed to 42 to make our model's predictions reproducible.

We run all our experiments on the FB15k-237 knowledge graph. The graph contains 40943 entities and 237 relations in total. Since FB15k-237 is a very large knowledge graph, it would be computationally expensive to run our simulations on the entire graph, especially since our experiment likely does not use the entire graph. To reduce the computational complexity of the experiment, we only run the simulations on a subgraph of FB15k-237 that contains 5% of the original graph's triples. Specifically, we generate a subgraph containing the pair of true and false target facts we are investigating by performing a Breadth-First Search starting from the target facts and moving further into the graph until the subgraph contains 5% of the knowledge graph. We generate a separate subgraph for every pair of target facts in our input.

We would be interested in expanding the scope of our analysis and verifying our results on other datasets as well. Since the WN18 and YAGO-3 datasets are also commonly used in KGE data poisoning papers, our next goal is to test our experiments on these datasets as well.

4.2 Fact Selection

We selected all our target facts from the training set of FB15k-237. Note that the 'true' and 'false' target facts we select for our simulation are both actually true (both are taken from the training dataset of the KG); however, in the experiment, we consider one of them to be misinformation.

We formulate our target facts as (h, r, t_t) and (h, r, t_f); therefore, the relation r must be either a many-to-many relation or a one-to-many relation for both facts to exist in the KG. We use the methodology suggested by Rossi et al. to find all the M-to-M and 1-to-M relations in FB15k-237. In particular, for each relation, they compute the average number of heads per tail and the average number of tails per head. The relation is considered many-to-many if both values are greater than 1.2, and it is considered one-to-many if only the average number of heads per tail is

[3]https://github.com/AndRossi/Kelpie

Paper	Datasets Used	KGE Models	Attack Type	Strategy	Assumptions	Promote/ Degrade	Metrics	Delete /Add Fact
Zhang et al.	FB15k, WN18	TransE, TransR, RESCAL	Direct, Indirect	Searching space of possible perturbations for top M perturbations	Attacker can make M perturbations at most, can add or delete facts, targeted triple does not exist in dataset	degrade, promote	MRR, Hits@10	delete, add
Bhardwaj et al.	FB15k-237, WN18	DistMult, ComplEx, ConvE, TransE	Direct	Using relation inference patterns to improve a set of decoy facts	Fact deletion is an expensive operation (use addition only)	degrade	MRR, Hits@10	add
Pezeshkpour et al.	YAGO3-10, WN18	DistMult, ConvE	Direct	Use gradient optimization to identify optimal embeddings of adversarial edits and decode the embeddings to their graph components using trained network	Approach can only be used on multiplicative KGE models	degrade	MRR, Hits@1	delete, add
Banerjee et al.	FB15K, WN18	TransE, TransR, DistMult	Direct	Use RL to find perturbations with low risk of exposure	Attacker has an exposure risk budget that they cannot exceed. High plausibility of fact being added corresponds to low risk	degrade	MRR, Hits@10	add

Table 1: A summary of comparison between KGE data poisoning attack strategies that have been developed in previous studies

greater than 1.2. Some examples of one-to-many facts include */film/director/film* (directedFilm) and */award/hall_of_fame/inductees./award/hall_of_fame_induction/inductee* (HallOfFameInductee). Some examples of many-to-many facts include */film/actor/film./film/performance/film* (actedIn) and */film/film/language* (filmLanguage). There are less one-to-many facts (14) than many-to-many (154).

After selecting a few one-to-many and many-to-many relations, we search for a pair of target facts that contain these relations and that satisfy the following criteria:

- **High Degree of Head Entity** We choose head entities h that have a high degree because we want target facts $(h, r, t_t) and$

After this selection process, we found 7 pairs of target (true and fact) facts that satisfy these criteria, shown in table 2. The facts span 2 relations, *film/actor/film./film/performance/film* (a many-to-many relation) and */film/director/film* (a one-to-many relation). We use the head entities *George Clooney*, *Ben Affleck*, and *Steven Spielberg* who were among the actors and directors with the highest degrees in FB15k-237.

We calculate a set of descriptive statistics, such as budget size, tail degree and head degree for each target fact in our input. These statistics are also shown in table 2.

4.3 Budget and Cost Assignment

We use the explanations from Rossi et al.'s Kelpie tool as the budget for each target fact. More specifically, we define the budget for a target fact *(h, r, t)* as a set of *n* 'necessary' tail explanations for that fact. The necessary tail explanations of a fact *(h, r, t)* are the smallest set of triples in the KG that contain *h* and that, if removed, would cause the tail ranking of the prediction *(h, r, ?)* to decrease the most. For instance, given a target fact *(Obama, president, United States)*, the necessary explanations for this fact could be *(Obama, bornIn, Hawaii)* and *(democratic party, includesMember, Obama)* (note that both contain the head entity, *Obama*). If these explanations were removed from the KG, the plausibility of *(Obama, president, United States)* would decrease greatly. For each target fact, we use Rossi et al.'s implementation of Kelpie to retrieve one necessary explanation and remove it from the KG, and we repeat this process until we have (n=25) explanations that we consider the budget for that target fact.

For the cost assignment, we consider cost to be proportional to the importance of a budget fact in promoting the target fact. Since Kelpie returns the explanations in order of their importance, we assign the cost of the budget facts in the same order they are returned by Kelpie, starting with cost = 25 (the first fact is the most important and thus the most expensive fact) and ending with cost = 1.

After retrieving the budget facts for both the true target fact and the false target fact, we identify any facts that overlap between both budgets and remove them from both. Given that we already selected our target facts to minimize the budget overlap, the number of overlapping facts was minimial (between 0 and 5).

4.4 Simulation

To setup our simulation, we remove the target facts (h, r, t_t) and (h, r, t_f) from the knowledge graph *k*. Then, we create a data void around both target facts by removing the facts of the disinformer budget $budget_d$ and the mitigator budget $budget_m$. At this point, we train the link prediction model on *k* and predict the plausibility of (h, r, t_t) and (h, r, t_f).

Our simulation consists of *n* rounds where *n* is the number of facts in $budget_d$ and $budget_m$. In each round, each of the mitigator and disinformer selects one fact from their budget and adds it to *k*. The agent selects which fact to add according to their chosen strategy. Our strategies and their descriptions are outlined in section 4.5. After both agents add their facts, we re-train the link prediction model on the updated knowledge graph and evaluate the new rankings of the target facts. We repeat this process for *n* rounds.

We run this simulation on every pair of facts shown in table 2. In every simulation, we assign, without loss of generalization, the first fact in each pair to be the mitigator and the second fact to be the disinformer. We then assign each agent a strategy from those discussed in Section 4.5. We run one simulation for every possible combination of agents and strategies (e.g. greedy mitigator and random disinformer, random mitigator and greedy disinformer, approximately greedy mitigator and greedy disinformer, etc.), but we exclude simulations where the disinformer uses the neighbor strategy, as we had an implementation issue that could not be resolved in time. Therefore, we had a total of (4 disinformer strategies x 5 mitigator strategies = 20) simulations per pair of facts. Since we have 7 pairs of facts, we run a total of 140 simulations. The entire experiment took 4 hours to run given that we ran every input pair's simulations in parallel on the university's high-performance computing cluster.

4.5 Strategies

We test 5 different strategies for adding facts to the data void:

- **Random Strategy** This is our baseline strategy. Following this strategy, an agent selects a random triple from their remaining budget facts and adds it to *k*. When we run a simulation that involves a random strategy, we repeat the simulation 5 times and we take the average of the rankings in each round across all runs to verify the robustness of our results. We set the random generator's seed to be 42 in the first run, and we increment it by 1 in every subsequent run (we use the seed 42 in the first run, 43 in the second run, 44 in the third, etc.).
- **Greedy Strategy** In each round, the agent's goal is to find the fact from their budget that, when added to the KG, will increase their target fact's plausibility the most. The agent does this by iterating through every proxy fact *p* in their budget, adding it to the KG *k*, training the link prediction model on *k+p*, evaluating the link prediction model on the target fact, then removing *p* from *k* and repeating the same process for the rest of their budget. The agent then selects the budget fact that, when added to *k*, results in the highest ranking for the target fact. We expect this strategy to yield the best result.

	Fact	Budget	Tail degree	Head degree	Avg degree of budget heads	Avg degree of budget tails
1	Ben Affleck /film/director/film The Town	22	47	162	124.68	85.95
	Ben Affleck /film/director/film Argo	22	123	162	115.05	107.73
2	Steven Spielberg /film/director/film Amistad	21	53	144	91.48	95.95
	Steven Spielberg /film/director/film Saving Private Ryan	21	108	144	109.57	102.76
3	Steven Spielberg /film/director/film The Adventures of Tintin	20	48	143	90.4	101.2
	Steven Spielberg /film/director/film Saving Private Ryan	20	108	143	103.25	100.3
4	Ben Affleck /film/actor/film./film/performance/film Shakespeare in Love	25	127	169	125.24	134.28
	Ben Affleck /film/actor/film./film/performance/film Jersey Girl	25	36	169	145.04	86.44
5	Ben Affleck /film/actor/film./film/performance/film Shakespeare in Love	24	127	172	123.88	129.62
	Ben Affleck /film/actor/film./film/performance/film Pearl Harbor	24	62	172	140.33	94.25
6	Ben Affleck /film/actor/film./film/performance/film Shakespeare in Love	24	127	172	121.62	134.67
	Ben Affleck /film/actor/film./film/performance/film Gigli	24	41	172	145.79	95.42
7	George Clooney /film/actor/film./film/performance/film Ocean's Twelve	23	50	137	106.26	87.22
	George Clooney /film/actor/film./film/performance/film Good Night, and Good Luck.	23	71	137	99.48	98.35

Table 2: Our 7 pairs of target facts and a set of descriptive statistics about each fact. We usually consider the first fact to be the mitigator's true fact and the second fact to be the disinformer's false fact, without loss of generality. "Budget" refers to the number of triples in the budget for each fact. "Tail degree" refers to the degree of the tail entity. "Head degree" refers to the degree of the head entity. Note that the head degree is different across input pairs, even for inputs with the same head entity (e.g. inputs 4 and 5). This is because we generate a different subgraph for every input pair (Section 4.1), so some triples that contain the head entity might not be included in the subgraph. For "Avg degree of budget head", we iterate through all the triples in the budget of that fact and we retrieve the head entity's degree for each triple, then we take their average. "Avg degree of budget tails" performs the same measurement on the tails of the budget facts. They are both proxy measures for how well-connected the facts' budgets are.

- **Approximate Greedy Strategy** The greedy strategy is computationally expensive because it requires retraining the link prediction model in every round of the simulation. We propose a computationally cheaper strategy, the approximate greedy strategy, that inserts the budget facts based on their order of importance as given by Kelpie. In other words, we insert the facts in the same order they are given by the Kelpie explanation generator. While this strategy is computationally cheaper than the greedy strategy, it is more costly according to our definition of cost because it selects the facts in the same order as their cost (see Section 4.3).
- **Neighbor Strategy** The idea behind this strategy is to add the budget facts in order of their distance to the target fact, starting with the budget facts closest to the target and ending with the ones farthest away. This strategy assumes that the proxy facts that are closest to the target fact will have the highest impact on the ranking of the target fact.
  Since the facts themselves are edges *(h, r, t)* and not nodes, we create our own metric for measuring distance between the target facts and budget facts. Given a target fact (h, r, t_t), any budget fact we retrieve from Kelpie will contain the target fact's head entity h by definition because it is a Kelpie explanation (Section 4.3). Therefore, a budget fact can either be (h_b, r_b, h), where h is the tail entity, or (h, r_b, t_b) where h is the head entity. For example, if an agent's target fact is *(Steven Spielberg, directedFilm, The Town)*, then *(Steven Spielberg, bornIn, Cincinnati)* could be a fact in the budget of that agent.
  To find the distance, we first remove the budget fact from the knowledge graph. Then, we choose the head or tail entity (h_b or t_b) in the budget fact that is not equal to h (e.g. in the example above, we would choose *Cincinnati*) and we find the length of shortest path from that entity to t_t (The Town) using Dijkstra's algorithm. We consider this value to be the distance of that budget fact from the target fact. We then add the budget fact back to the knowledge graph. After calculating the distance for all the facts in the budget, we sort them in increasing order of distance, and throughout the simulation, the agent will add them back to the KG in the same order as their distance.
  One major limitation of this strategy is that, because the budget facts are all very closely related to the target facts, the distance value (length of the shortest path) is often between 0 and 2. Given this small range, the facts might be added in practically random order.
- **Multi-Objective Greedy** This strategy is the same as the **Greedy** strategy, but it adds cost as a consideration when selecting budget facts. Similarly to greedy, it evaluates the influence of each budget fact on the ranking of the target fact by adding the budget fact to the KG, retrainig the lp model, and evaluating the model on the target fact. In addition, the multi-objective greedy strategy then uses the following function to assign a

value v to each budget fact

$$v = r \times w_r + c \times w_c$$

where r is the ranking of the target fact after that budget fact was added, c is the cost of that budget fact, w_r is the weight assigned to the ranking and w_c is the weight assigned to the cost. We set w_c and w_r both to be equal to 1. Since lower rank and lower cost indicates better performance, our objective is to minimize v. Therefore, in every round, we recalculate v for all remaining budget facts (since r will change after each round, given that new facts are added to k), and we select the budget fact with the lowest v.

5 RESULTS AND ANALYSIS

Figures 1- 4 show the results of the experiment. Recall that we run simulations for every possible combination of strategies and agents. To visualize the results more easily, we separate them into 4 figures where each figure fixes the disinformer's strategy as constant and shows the performance of all the mitigator strategies. For instance, figure 1 shows the results of all the simulations where the disinformer used the approximate greedy strategy. The rows represent the input pairs 1-7 from table 2, and the columns represent the different mitigator strategies. For example, in figure 1, the graph in the first row, first column shows the results of running the simulation on the first pair of facts in table 2, where the mitigator was using the neighbor strategy and the disinformer was using the approximate greedy strategy.

The line plots in the figure show the ranking of the *true* fact (left y-axis) and the cumulative cost accrued by the mitigator (right y-axis) throughout the rounds of the simulation. There are 4 lines in each plot; the solid lines represent the true fact's rankings and the faded lines represent the cost. The red lines represents the rankings/cost when the mitigator used the experimental strategy (greedy, approximate greedy, neighbor, or multi-objective greedy), and the blue lines represent the rankings when the mitigator used the baseline strategy (random). A lower ranking indicates a better performance.

We use the AUC score as a metric to measure the performance of a strategy. The AUC score is calculated as the area under the curve of the rankings in the experimental strategy minus the area under the curve of the rankings in the random strategy:

$$AUC = AUC_{exp} - AUC_{rand}$$

Effectively, this AUC score measures the area between the solid red and blue lines in each graph.

Overall, looking at the AUC rankings of the simulations, it appears that the greedy, approximate greedy, and neighbor strategies all perforn well when it comes to boosting the ranking of the mitigator's target fact, regardless of what strategy the disinformer employs. However, the greedy strategy appears to perform slightly better than the other two because it updates its selection of the best budget fact every round while taking into account the new facts that were added to the KG in the previous round. On the other hand, the approximate greedy and neighbor strategies both determine the order of the facts they will insert before the simulation starts, so they do not take into account the disinformer's strategy or the facts being added throughout the experiment. It is surprising, however, that the neighbor strategy performs so well, especially given that it adds the triples that are closest to the target fact in the graph, and these triples are not necessarily the most important or relevant to the target facts.

The multi-objective greedy strategy appears to be the worst performer, yielding rankings that are even worse than the random baseline strategy. It is possible that, since this strategy takes into account cost as well as ranking, it may implicitly weight the cost more strongly than the rankings because they are on different scales, so it ends up adding the cheaper budget facts that have less relevance to the target fact. It is also possible that the rankings and the cost may need to be standardized in some way before being added to produce v.

Cost-wise, the approximate greedy experiment is the most costly one by design because it adds the most expensive facts first. Interestingly, however, the greedy and neighbor strategies, which perform similarly to approximate greedy, tend to have cost trends that are similar to the random baseline. This suggests that the neighbor and greedy strategies are able to find cheaper facts that, when added to the KG, produce the same effect as the more expensive facts used by the approximate greedy strategy. These strategies may be useful to a mitigator with a limited budget who wants to maximize their rank as quickly as possible. On the other hand, the multi-objective strategy consistently has a lower cost than the random baseline. This confirms our hypothesis that the strategy is likely not working as intended, as it may be weighing the cost of the budget fact more than its impact on the ranking of the target fact. One solution to rectify this issue may be to assign a higher weight to the rankings than the cost (e.g. $w_r = 2$ and $w_c = 1$). We leave this exploration for future work.

We run a set of statistical tests to corroborate our findings from the matrix plots. The results of the tests are shown in figure 5. We fix the disinformer's strategy in each subfigure and run a t-test to compare the average AUC scores of different mitigator strategies. The rows and columns represent mitigator strategies. The two numbers in each cell are the mean AUC scores that were produced when the mitigator used the column strategy (first number) and the row strategy (second number), and the colors represent the significance

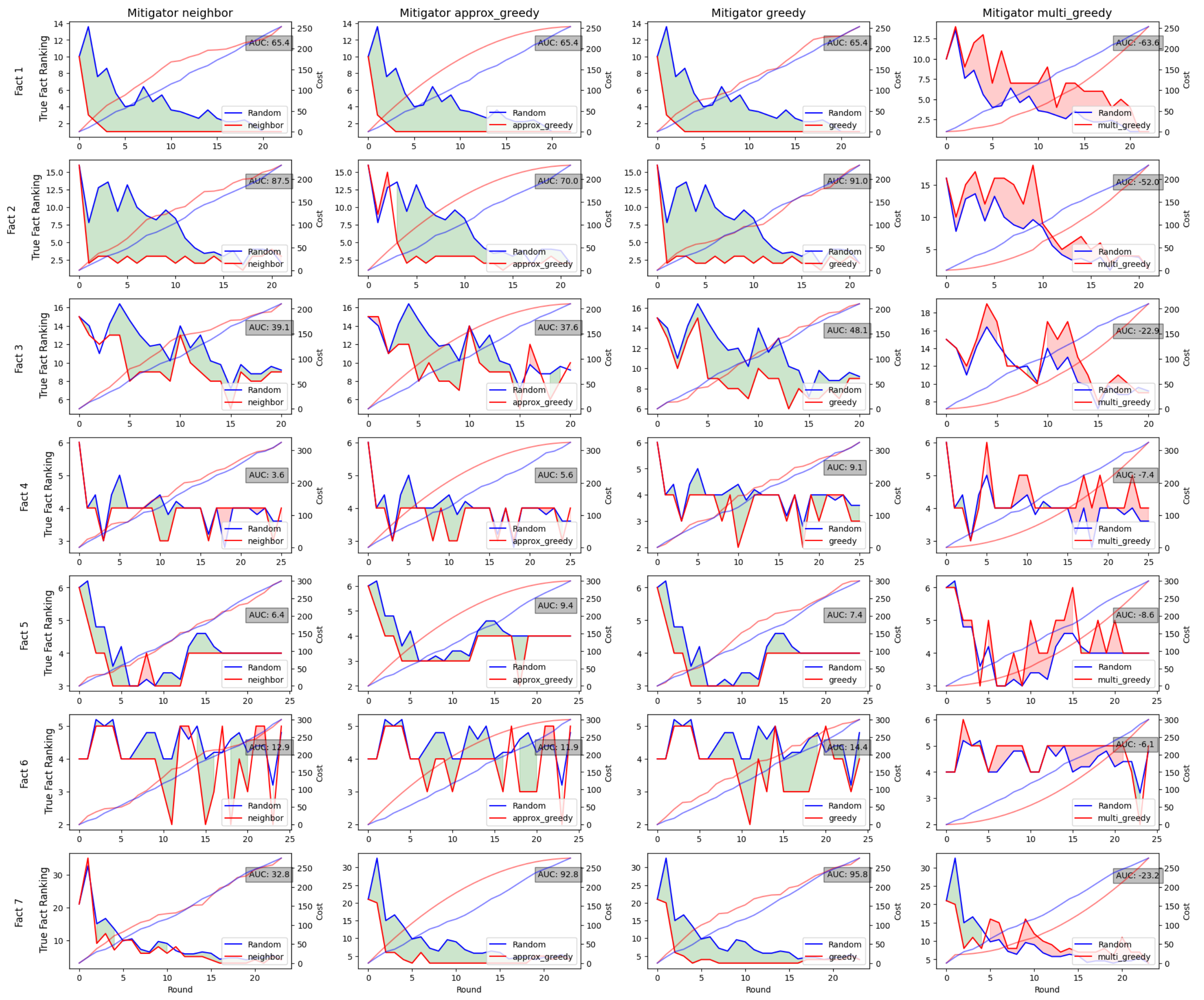


Figure 1: ***Ranking of the mitigator's when the mitigator is using different strategies and the disinformer is using the approximate greedy strategy*** **- This figure shows the ranking of the true target fact (the mitigator's fact) and the cumulative cost accrued by the mitigator in each round, focusing only on simulations where the disinformer was using an approximate greedy strategy. The x-axis represents the rounds in the simulation, where in each round, each of the mitigator and the disinformer add one proxy fact to the KG. The left y-axis represents the rank of the true target fact. A lower rank indicates a better performance. The right y-axis represents the cumulative cost that the mitigator accrues when they add a new fact to the KG in each round. The solid red line represents the ranking of the true fact when the mitigator used an experimental strategy (neighbor, greedy, approximate greedy, or multi-objective greedy). The solid blue line represents the ranking of the true fact when the mitigator used a baseline (random) strategy. If the fill between the solid red line and the solid blue line is green, then the experimental strategy is performing better than the baseline strategy. If it is red, then the experimental strategy is performing worse than the random strategy. The faded red and blue lines represent the cumulative costs of the experimental strategy and the baseline strategy respectively. The AUC score in each graph represents the area between the solid red and solid blue lines (see Section 5 for the AUC calculation).**

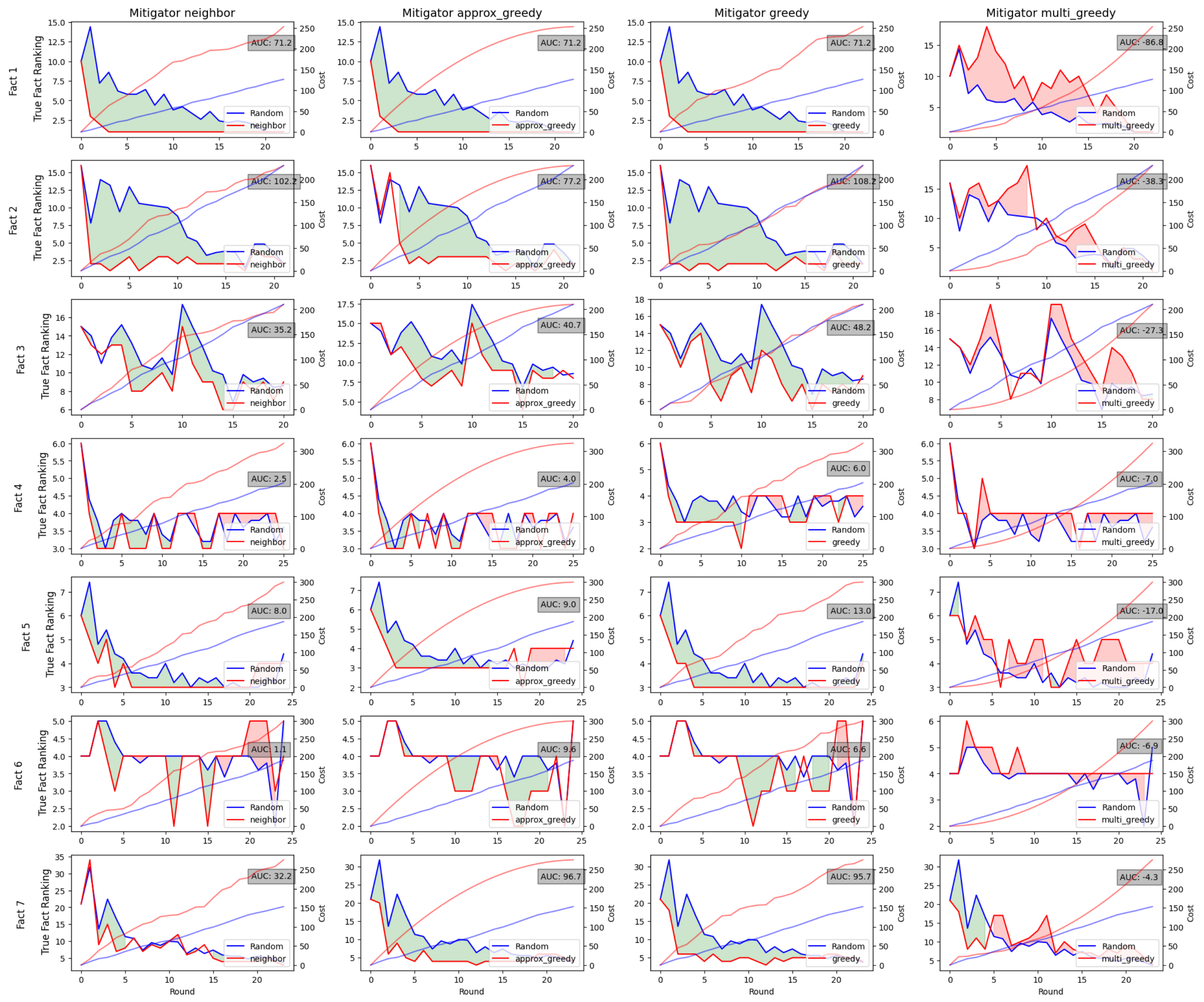


Figure 2: Ranking of the mitigator's when the mitigator is using different strategies and the disinformer is using the greedy strategy. See Figure 1 for a more detailed explanation of the figure.

level of the differences. For instance, in the third column of figure 5(a), the multi-objective greedy strategy has a significantly different effect from all the other strategies, as we noted from in the plots. The mean AUC of mitigators that use the multi-objective greedy strategy is -192.41, which suggests that this strategy performs worse than the baseline. Surprisingly, the AUC scores produced by the neighbor strategy appear to be significantly less than both the approximate greedy strategy and the greedy strategy in all cases, except when the disinformer uses the multi-objective greedy strategy, in which case there is no significant difference between the neighbor strategy and the approximate greedy strategy. This may be the case because the multi-objective greedy strategy is the weakest strategy, so if a disinformer is using that strategy, the mitigator can outperform it even using a simple strategy like the neighbor strategy. There is no significant difference between the approximate greedy and greedy strategies in all cases, so it would make most sense for a mitigator to use the greedy strategy to reduce their costs.

6 LIMITATIONS AND FUTURE WORK

While our work establishes a few ideas for strategies that mitigators can use to promote a target fact in a data void,

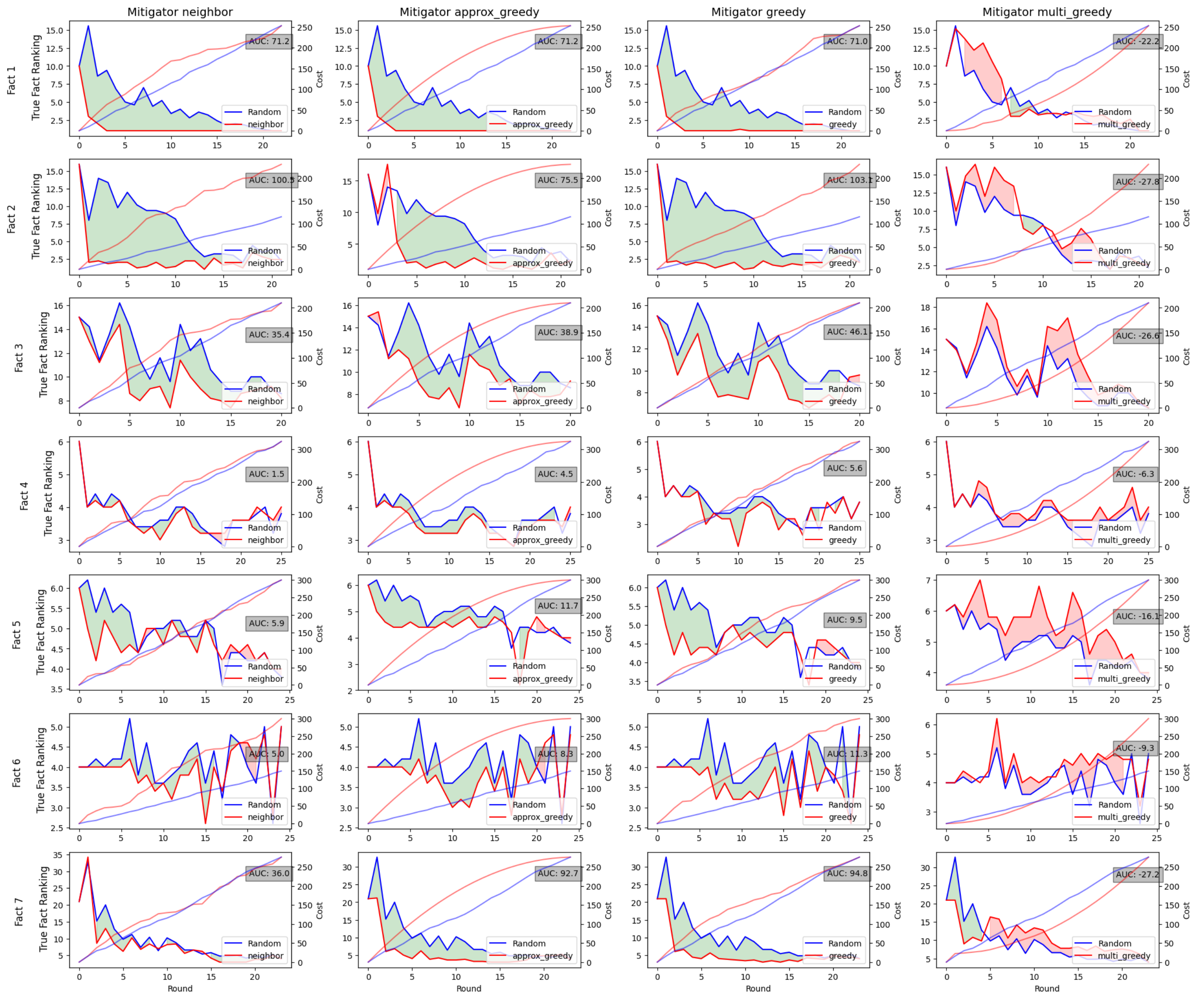


Figure 3: Ranking of the mitigator's when the mitigator is using different strategies and the disinformer is using the Random strategy. See Figure 1 for a more detailed explanation of the figure

there are several limitations that may need to be addressed. Firstly, the sample size of 7 inputs might be too small to establish any concrete findings about which strategies are most effective. Given more time, we would like to run the simulation on more inputs, possibly from other datasets such as WN18 and YAGO3-10. The potential issue with using these other datasets is that they have far fewer relations than FB15k-237, so it may be difficult to find input facts that meet our selection criteria.

Another limitation is the multi-objective greedy strategy, which did not perform as we expected. There are opportunities to improve the design of the strategy by standardizing the cost and ranking scores or changing their weights, which would prevent the algorithm from assigning too much weight on the cost.

In addition, that we only test our simulation on the ComplEx model, although our strategies are model-agnostic. As a robustness check, in the future we would like to run the experiment on other types of link prediction models such as TransE and ConvE.

Future research may explore other cost-effective strategies a mitigator can pursue. Furthermore, since our experiment assumes that the disinformer and mitigator have the same

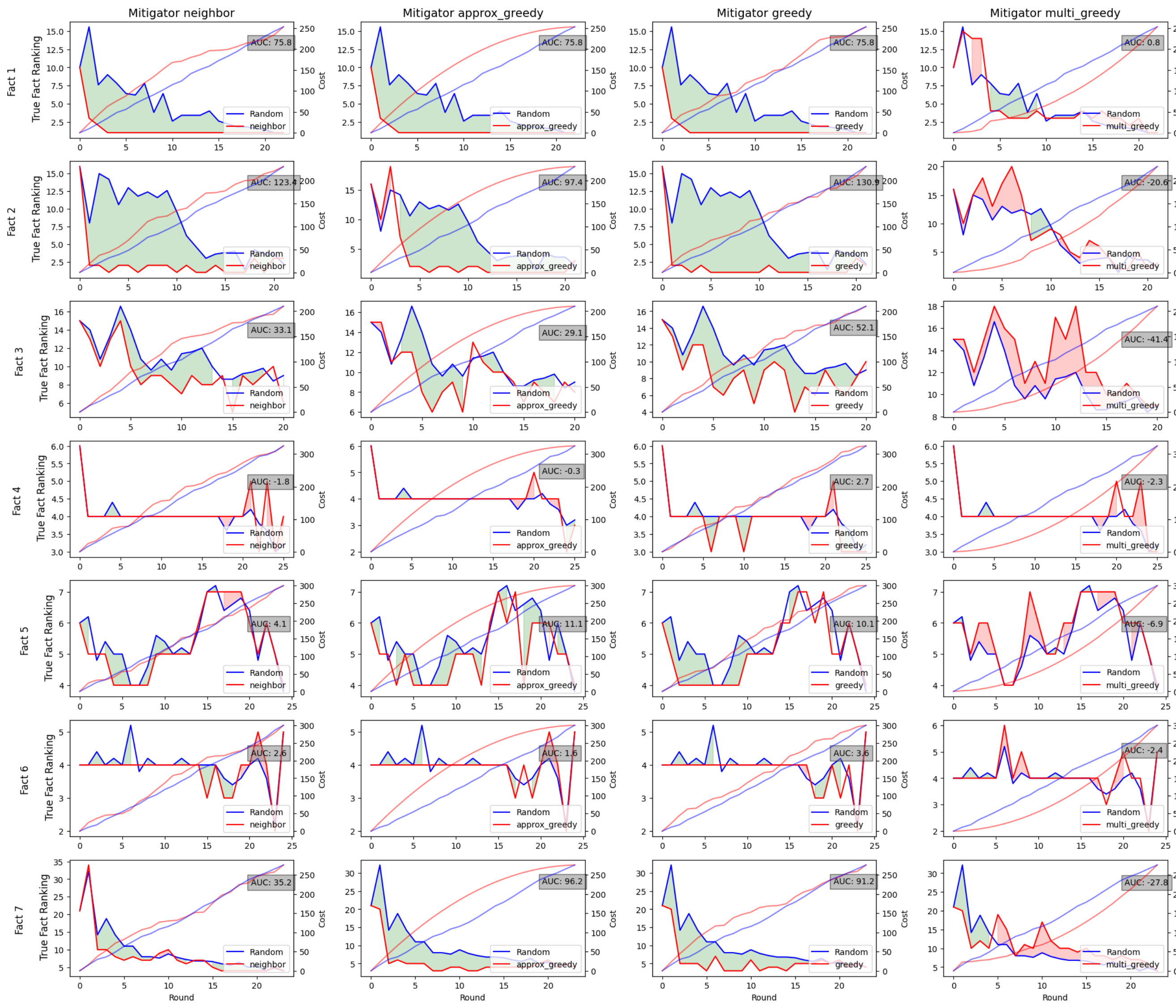
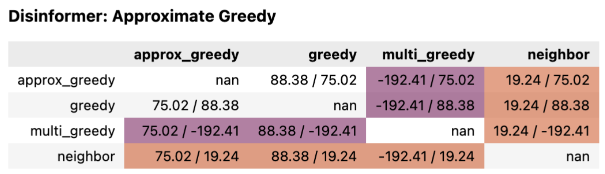


Figure 4: Ranking of the mitigator's when the mitigator is using different strategies and the disinformer is using the Multi-Objective Greedy strategy. See Figure 1 for a more detailed explanation of the figure

budget size and can add facts at the same time, future research may address more realistic scenarios such as imbalances between the mitigator and disinformer's budgets or situations where the mitigator has a delayed start relative to the disinformer.

7 CONCLUSION

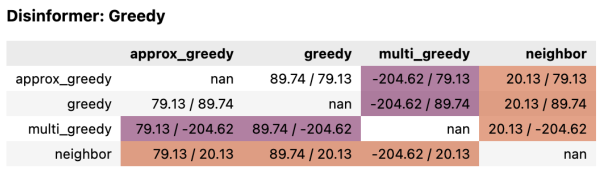
Although knowledge graphs are being used extensively for a wide range of applications in industry, their security vulnerabilities have not been studied comprehensively. Our work builds on existing research surrounding adversarial attacks on knowledge graphs, and we propose strategies to mitigate misinformation in knowledge graph data voids. We find that the most effective strategies are the greedy and approximate greedy strategies, but the approximate greedy strategy is more expensive. We also find that the neighbor strategy, which is less sophisticate than greedy and approximate greedy, can perform at a similar level under specific conditions, namely when the disinformer is using the multi-objective greedy strategy. Finally, we find that our multi-objective greedy strategy greatly underperforms the random strategy baseline because it selects the cheaper budget facts that are less relevant to the target fact.



Disinformer: Approximate Greedy

	approx_greedy	greedy	multi_greedy	neighbor
approx_greedy	nan	88.38 / 75.02	-192.41 / 75.02	19.24 / 75.02
greedy	75.02 / 88.38	nan	-192.41 / 88.38	19.24 / 88.38
multi_greedy	75.02 / -192.41	88.38 / -192.41	nan	19.24 / -192.41
neighbor	75.02 / 19.24	88.38 / 19.24	-192.41 / 19.24	nan

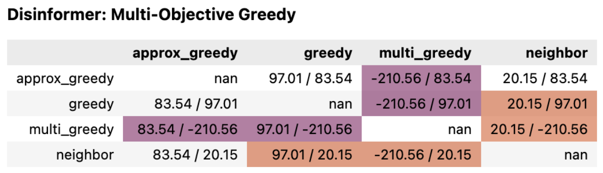
(a) Disinformer: Approximate Greedy



Disinformer: Greedy

	approx_greedy	greedy	multi_greedy	neighbor
approx_greedy	nan	89.74 / 79.13	-204.62 / 79.13	20.13 / 79.13
greedy	79.13 / 89.74	nan	-204.62 / 89.74	20.13 / 89.74
multi_greedy	79.13 / -204.62	89.74 / -204.62	nan	20.13 / -204.62
neighbor	79.13 / 20.13	89.74 / 20.13	-204.62 / 20.13	nan

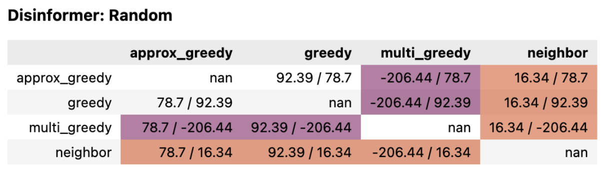
(b) Disinformer: Greedy



Disinformer: Multi-Objective Greedy

	approx_greedy	greedy	multi_greedy	neighbor
approx_greedy	nan	97.01 / 83.54	-210.56 / 83.54	20.15 / 83.54
greedy	83.54 / 97.01	nan	-210.56 / 97.01	20.15 / 97.01
multi_greedy	83.54 / -210.56	97.01 / -210.56	nan	20.15 / -210.56
neighbor	83.54 / 20.15	97.01 / 20.15	-210.56 / 20.15	nan

(c) Disinformer: Multi-Objective Greedy



Disinformer: Random

	approx_greedy	greedy	multi_greedy	neighbor
approx_greedy	nan	92.39 / 78.7	-206.44 / 78.7	16.34 / 78.7
greedy	78.7 / 92.39	nan	-206.44 / 92.39	16.34 / 92.39
multi_greedy	78.7 / -206.44	92.39 / -206.44	nan	16.34 / -206.44
neighbor	78.7 / 16.34	92.39 / 16.34	-206.44 / 16.34	nan

(d) Disinformer: Random

Figure 5: The results of comparing the average AUC scores for different mitigator strategies. For every pair of mitigator strategies (e.g. greedy and approximate greedy), we run an independent t-test to compare the average AUC scores of the true fact when the mitigator was using those strategies. We specifically use Welch's t-test to correct for unequal variances. The cells in the tables contain the mean AUC score for the column's strategy and the row's strategy respectively. The color represents the significance level of the difference between them (orange: $p < 0.1$, purple: $p < 0.05$).

REFERENCES

[1] [n.d.]. google.com Market Share, Revenue and Traffic Analytics. https://www.similarweb.com/website/google.com/

[2] [n.d.]. Measuring the impact of COVID-19 vaccine misinformation on vaccination intent in the UK and USA | Nature Human Behaviour. https://www.nature.com/articles/s41562-021-01056-1

[3] 2012. Introducing the Knowledge Graph: things, not strings. https://blog.google/products/search/introducing-knowledge-graph-things-not/

[4] 2019. Identify Content Gaps Online with Question Hub. https://blog.google/intl/en-in/products/explore-communicate/identify-content-gaps-online-with/

[5] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web*, Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 722–735.

[6] Prithu Banerjee, Lingyang Chu, Yong Zhang, Laks V.S. Lakshmanan, and Lanjun Wang. 2021. Stealthy Targeted Data Poisoning Attack on Knowledge Graphs. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. 2069–2074. https://doi.org/10.1109/ICDE51399.2021.00202

[7] Peru Bhardwaj, John Kelleher, Luca Costabello, and Declan O'Sullivan. 2021. Poisoning Knowledge Graph Embeddings via Relation Inference Patterns. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Online, 1875–1888. https://doi.org/10.18653/v1/2021.acl-long.147

[8] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data* (Vancouver, Canada) *(SIGMOD '08)*. Association for Computing Machinery, New York, NY, USA, 1247–1250. https://doi.org/10.1145/1376616.1376746

[9] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems*, C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (Eds.), Vol. 26. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf

[10] Danah Boyd and Michael Golebiewski. 2018. Data Voids: Where Missing Data Can Easily Be Exploited. https://datasociety.net/library/data-voids-where-missing-data-can-easily-be-exploited/ Publisher: Data & Society Research Institute.

[11] Rafiq Copeland, Jenny Fan, and Tanay Jaeel. 2020. Filling the data void. https://datavoids.2020.bkmla.org/#Part1

[12] Ollie Davies. 2023. What happens if teens get their news from TikTok? *The Guardian* (Feb. 2023). https://www.theguardian.com/media/2023/feb/22/what-happens-if-teens-get-their-news-from-tiktok

[13] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D Knowledge Graph Embeddings. arXiv:cs.LG/1707.01476

[14] Claudia Flores-Saviaga, Shangbin Feng, and Saiph Savage. 2022. Datavoidant: An AI System for Addressing Political Data Voids on Social Media. *Proceedings of the ACM on Human-Computer Interaction* 6, CSCW2 (Nov. 2022), 503:1–503:29. https://doi.org/10.1145/3555616

[15] Viet-Phi Huynh and Paolo Papotti. 2019. A Benchmark for Fact Checking Algorithms Built on Knowledge Bases. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM, Beijing China, 689–698. https://doi.org/10.1145/3357384.3358036

[16] Seyed Mehran Kazemi and David Poole. 2018. SimplE Embedding for Link Prediction in Knowledge Graphs. arXiv:stat.ML/1802.04868

[17] HANNAH MARSHALL and ALENA DRIESCHOVA. 2018. Post-Truth Politics in the UK's Brexit Referendum. *New Perspectives* 26, 3 (2018), 89–106. https://www.jstor.org/stable/26675075 Publisher: Institute of International Relations, NGO.

[18] Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. 2019. Anytime Bottom-Up Rule Learning for Knowledge Graph Completion. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 3137–3143. https://doi.org/10.24963/ijcai.2019/435

[19] Ov Cristian Norocel and Dirk Lewandowski. 2023. Google, data voids, and the dynamics of the politics of exclusion. *Big Data & Society* 10, 1 (Jan. 2023), 20539517221149099. https://doi.org/10.1177/20539517221149099 Publisher: SAGE Publications Ltd.

[20] Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. 2019. Industry-Scale Knowledge Graphs: Lessons and Challenges. *Commun. ACM* 62, 8 (jul 2019), 36–43. https://doi.org/10.1145/3331166

[21] Pouya Pezeshkpour, Yifan Tian, and Sameer Singh. 2019. Investigating Robustness and Interpretability of Link Prediction via Adversarial Modifications. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 3336–3347. https://doi.org/10.18653/v1/N19-1337

[22] Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. 2021. Knowledge Graph Embedding for Link Prediction: A Comparative Analysis. *ACM Transactions on Knowledge Discovery from Data* 15, 2 (April 2021), 1–49. https://doi.org/10.1145/3424672

[23] Andrea Rossi, Donatella Firmani, Paolo Merialdo, and Tommaso Teofili. 2022. Explaining Link Prediction Systems Based on Knowledge Graph Embeddings. In *Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA) *(SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 2062–2075. https://doi.org/10.1145/3514221.3517887

[24] Barry Schwartz. 2022. Google Question Hub to close down. https://searchengineland.com/google-question-hub-to-close-down-390375

[25] Elisa Shearer and Amy Mitchell. 2021. News Use Across Social Media Platforms in 2020. https://www.pewresearch.org/journalism/2021/01/12/news-use-across-social-media-platforms-in-2020/

[26] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A Core of Semantic Knowledge. In *Proceedings of the 16th International Conference on World Wide Web* (Banff, Alberta, Canada) *(WWW '07)*. Association for Computing Machinery, New York, NY, USA, 697–706. https://doi.org/10.1145/1242572.1242667

[27] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. arXiv:cs.LG/1902.10197

[28] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. arXiv:cs.AI/1606.06357

[29] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A Free Collaborative Knowledgebase. *Commun. ACM* 57, 10 (sep 2014), 78–85. https://doi.org/10.1145/2629489

[30] Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. arXiv:cs.CL/1412.6575

[31] Hengtong Zhang, Tianhang Zheng, Jing Gao, Chenglin Miao, Lu Su, Yaliang Li, and Kui Ren. 2019. Data Poisoning Attack against Knowledge Graph Embedding. (2019), 4853–4859. https://www.ijcai.org/proceedings/2019/674