Auditing the Sensitivity of Graph-based Ranking with Visual Analytics

Tiankai Xie, Yuxin Ma, Hanghang Tong, My T. Thai, Ross Maciejewski

Fig. 1. Sensitivity analysis of HITS on political blogs. (1) A predefined rule is set to exclude all perturbations that would cause the rankings of the top-5 blogs to decrease. (2) The blog `liberaloasis.com` has the largest influence under this constraint, and its removal can increase the rankings of the conservative blogs while decreasing the rankings of the liberal blogs. (3) The influence overview indicates that nearly 2/3 of the influenced nodes see a ranking increase. (4) The ranking change distribution view further shows that most of the ranking-increased nodes are conservative blogs and most of the ranking-decreased nodes are liberal blogs, from which the top-3 heavily influenced nodes are ranked 200th or below. (5) The top-k proportional view shows that the proportion of liberal blogs decreased from 82% to 77% in the top-100 due to the perturbation.(6, 7) The influence graph view implies that the removal of `liberaloasis.com` has a direct influence on the majority of the liberal nodes (including the top-3 influenced nodes), and as the influence distance increases, more conservative nodes are indirectly influenced.

Abstract—Graph mining plays a pivotal role across a number of disciplines, and a variety of algorithms have been developed to answer who/what type questions. For example, what items shall we recommend to a given user on an e-commerce platform? The answers to such questions are typically returned in the form of a ranked list, and graph-based ranking methods are widely used in industrial information retrieval settings. However, these ranking algorithms have a variety of sensitivities, and even small changes in rank can lead to vast reductions in product sales and page hits. As such, there is a need for tools and methods that can help model developers and analysts explore the sensitivities of graph ranking algorithms with respect to perturbations within the graph structure. In this paper, we present a visual analytics framework for explaining and exploring the sensitivity of **any** graph-based ranking algorithm by performing perturbation-based what-if analysis. We demonstrate our framework through three case studies inspecting the sensitivity of two classic graph-based ranking algorithms (PageRank and HITS) as applied to rankings in political news media and social networks.

Index Terms—Graph-based ranking, sensitivity analysis, visual analytics

1 Introduction

Currently, the development of visual analytics methods and tools for explainable artificial intelligence (XAI) primarily tackles analytical tasks in vector-space learning, such as classification [4, 49], clustering [10, 27], and outlier detection [27, 48]. However, graph-based learning algorithms are significantly different from vector-space representations, and these differences have not been sufficiently studied in the visual analytics community. Specifically, graph-based ranking algorithms have received little attention; however, algorithms such as

- *T. Xie, Y. Ma, and R. Maciejewski are with Arizona State University. E-mail: {txie21, yuxinma, rmacieje}@asu.edu.*
- *H. Tong is with the University of Illinois at Urbana-Champaign. E-mail: htong@illinois.edu.*
- *M. Thai is with the University of Florida. E-mail: mythai@cise.ufl.edu.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

PageRank [32] and HITS [23] are foundational in industrial information retrieval settings. In these information retrieval settings, a person searches a graph-based dataset looking for relevant objects, and the resultant ranking order has a major impact with respect to exposure. For example, recent work by Singh and Joachims [34] demonstrated that the exposure of resumes to potential employers could be reduced by upwards of 30% if an item's rank fell by as little as three places.

Previous work [11, 21, 22, 31] has demonstrated that the results of such graph-ranking algorithms can be highly sensitive to perturbations within the graph structure, and these sensitivity issues give rise to ranking manipulations. Given the importance of the ranking results, it is imperative that algorithm designers and analysts understand the underlying algorithmic sensitivities and vulnerabilities. Consider a news navigation website [3] where the consumer can search political-related blogs and posts. The search result rankings are determined by a graph ranking algorithm, and higher ranked stories are more likely to be read and shared. Here, one could imagine a nation-state actor that would want to promote biased content. The nation-state actor can create webpages to add various links in the graph structure, or even identify websites to shadow ban, which could manipulate the ranking results so that certain political opinions are more exposed to the public. Given the importance of such rankings, it is critical that model developers have access to tools that can support them in understanding the ranking methods' sensitivity to structural changes in the graph.

In this paper, we propose a modularized visual analytics framework that facilitates auditing and diagnosing **any** graph-based ranking method's sensitivities by performing what-if analysis over a given graph dataset via node perturbation. The interactive perturbation of a graph's nodes through coordinated views enables analysts to explore and identify algorithmic sensitivities. A summarization view for the sensitivity index (i.e., the degree of the ranking method's sensitivity to the perturbation) facilitates the identification of the graph-ranking method's instance-level sensitivity. A group of views quantifies the impacts of perturbations through the comparison of statistical information about the ranking results, and a local graph influence view supports the inspection of ranking changes due to changes in the graph topology. To demonstrate our framework, we explore three case studies using real-world datasets including: a Facebook social circle [29], Political blogs [3], and a Reddit interaction network [25]. While our framework is designed to support any general graph-based ranking algorithm, only PageRank and HITS are used for demonstration purposes. Our contributions include:

- A visual analytics framework that facilitates performing what-if analysis on graph data to reveal the instance-level sensitivity in terms of **any** graph-based ranking method, and;
- A novel representation of the influence graph caused by specific perturbations to illustrate the relationship between ranking influence and topological structures.

2 Related Work

Our work focuses on explaining the sensitivities of graph-based ranking models in relationship to perturbations in the graph structure. In this section, we review recent work on ranking tasks and sensitivities in graph theory and visualization methods applied to graphs and rankings.

2.1 Graph Ranking

Graph ranking methods are ubiquitous, with applications of these algorithms found in web searches, e-commerce, hiring, and numerous other domains. The most well-known methods for graph ranking include Google's PageRank [32] and Kleinberg's HITS [23]. PageRank computes the quality of hyperlinks to webpages in order to approximate the importance of webpages. Webpages are treated as nodes of a graph, and the edges of this graph are the hyperlinks between webpages. Given a graph $\mathbf{G}$ with n nodes, PageRank computes the importance of nodes as:

$$\boldsymbol{r} = c\boldsymbol{A}\boldsymbol{r} + (1-c)\boldsymbol{t} \tag{1}$$

where $\mathbf{r}$ is a vector of size n that denotes the *PageRank Value (or PR Value)* for each node in the graph $\mathbf{G}$. The higher the PR value of a node, the higher ranked (or more important) the node is. $\mathbf{A}$ is the normalized adjacency matrix of graph $\mathbf{G}$, c is a constant damping factor, which is typically set as 0.85 [9], and $\mathbf{t}$ is the teleportation vector, representing the initial PR value for each node. $\mathbf{t}$ is typically the uniform probability distribution $\frac{1}{n}$ $\mathbf{1}$. The computation of $\mathbf{r}$ will eventually converge at the final ranking value for each node.

Unlike PageRank, HITS (Hyperlinked Induced Topic Search) [23] treats webpages as "authorities" and "hubs", where "authorities" denote the webpages with more incoming hyperlinks, and "hubs" denote the webpages with more outgoing hyperlinks. Two update rules are applied to compute each node's "authority" score and "hub" score:

$$\begin{cases} auth(v) = \sum_{w \in V_{to}} hub(w) \\ hub(v) = \sum_{w \in V_{from}} auth(w) \end{cases} \tag{2}$$

where V_{to} is a set of all nodes which point to the node v, and V_{from} is a set of all nodes that node v points to. The authority and the hub vector are initialized to 1 for each node, and the two computations are performed repeatedly until the values of the two vectors converge.

Both PageRank and HITS are propagation-walked-based techniques, and there are numerous variants of these algorithms. For example, the Personalized PageRank algorithm [32] initializes a biased teleportation vector instead of one that has the same teleportation value for all nodes in a graph. The biased teleportation vector enables the generation of distinct ranking results that can be personalized for a set of nodes that the vector corresponds to. ItemRank [16] replaces the adjacency matrix with the stochastic matrix of a graph to transform PageRank into a biased version, which is commonly applied in recommendation systems. IsoRank [35] treats the task of comparing sets of graphs to find correspondences between nodes as the eigenvalue problem and has been used to align protein to protein interaction networks. TwitterRank [45] utilizes a transition probability matrix in which the similarity between twitterers on certain topics is used to identify the topic-sensitive influential twitterers, and TopicRank [8] uses the semantic relationship between topics as part of the document ranking process. These methods all have the same underlying data structure requirements, which enables our framework to seamlessly swap between algorithms.

2.2 Graph Auditing

While numerous graph-based ranking algorithms have been developed, it is only recently that researchers have begun exploring methods to audit network/graph learning methods in an attempt to identify issues of fairness and bias. Kang et al. [22] define the *PageRank Auditing Problem* as the task of finding the k graph elements (nodes or edges) that have the largest influence on the overall ranking changes of a given graph. Kang et al. measure the ranking changes of a given graph to compute the derivative of a loss function of the Pagerank vector over the adjacency matrix. However, in this method, the k graph elements have to be computed k times in total, and each time the ranking method has to be rerun and the derivative values must be re-computed.

Other sensitivity analysis methods have focused on performing perturbations on graph elements to explore vulnerabilities of graph-based learning models [12–14, 36, 51]. For example, Ng et al. [31] utilize the techniques in matrix perturbation theory and coupled Markov chain theory to evaluate the stability of PageRank and HITS. Chartier et al. [11] perform a comparative analysis of the effects of perturbations on graph structures exploring various ranking methods. However, such definitions of auditing only determine **which** elements have a high influence for a specific ranking method. They do not provide visual details on **why** and **how** the perturbations influence ranking. Our framework is designed to support the analysis of multiple perturbations to help explain potential vulnerabilities in graph-based ranking algorithms.

2.3 Graph Drawing and Ranking Visualization

A key mechanism for supporting the analysis of graph-based ranking algorithms is visualization. Graph drawing methods focus on algorithms that optimize the layout of the graph structure to highlight key features in a graph. Numerous graph drawing methods have been developed (see recent state-of-the-art reports for a comprehensive summary [5, 6, 38]),

with recent methods focusing on exploiting deep learning approaches to generate graph layouts for large datasets [28, 42], edge bundling to improve layout readability [37, 43], and extensions to the force-directed layout [50]. In this framework, we apply a customized radial graph layout with redesigned encodings to reveal the effects of perturbation.

Another major task in our proposed framework is to reveal the relationships between a graph node's ranking change and its topological structure, and many approaches have been proposed to visually represent ranking changes. For example, a variety of systems [30, 33, 39, 47] have been developed to support visualizing ranking changes in time-series data. Lu et al. [30] explore the use of parallel coordinate plots to visualize ranking changes through time. Shi et al. [33] use a stacked area chart where the order shows the ranking and the stacks show the proportional changes. Vuillemot et al. combine a table view and a line chart to show ranking changes over time for a group of tabular data. Similarly, Xia et al. [47] show the PageRank results of top Wikipedia topics and reveals the connection between those topics.

Other research [17, 26, 44] has focused on ranking and multi-dimensional data comparisons. Clustervision [26] provides a proportional bar and radar glyph to show metrics for each cluster in a ranking list. SRVis [44] integrates spatial data into ranking visualization to support decision making. LineUp [17] utilizes bar charts to de-factorize and combine attributes to show the rank changes under selected attributes. Podium [40] uses an SVM classifier to help analysts rank the tabular data to fit their mental model, and GUIRO [7] is used to improve matrix reordering methods with animated reordering representations. However, none of those approaches explore the relationship between ranking changes and the topological structure of graphs.

3 Design Overview

Given the large scale use of graph-based ranking algorithms, we have developed a visual analytics framework to support developers and analysts in exploring and explaining ranking sensitivities. Our framework is designed to be robust to general graph-based ranking algorithms, and supports the removal of nodes as the key perturbation method. While other types of perturbations exist, such as adding/removing nodes/edges [20, 22, 41], our framework focuses on the removal of a node and its corresponding edges as a proof-of-concept interaction. The removal perturbation allows us to constrain the computational and exploration space. However, the interactions and design are robust to all types of perturbations, which will be explored in future work. Our potential target audience includes researchers, developers, and analysts who are building and/or deploying graph-based ranking applications. Our goal is to facilitate those experts analysis of the sensitivity of their chosen ranking methods and support them in auditing the applied ranking algorithms before deployment.

3.1 Analytical Tasks

After reviewing recent literature on graph auditing [21,22], we extracted common high-level tasks for the auditing process. These tasks were refined with our co-authors, domain-experts in graph-mining.

T1 Summarize instance-level sensitivity. The key analytic task is to identify an individual node's ranking and the sensitivity of this node's ranking to changes in the graph structure. Our framework is designed to provide an overview of the ranking results, and enables analysts to explore any node's sensitivity to perturbation. For example:

- **T1.1** Which perturbation causes the largest ranking changes?
- **T1.2** How do perturbations cause ranking changes, i.e., are there topological features that leading to ranking instability?
- **T1.3** Are nodes with specific attributes more sensitive to perturbations in the network, i.e., does the removal of a node of group A lead to changes in the ranking of nodes in group B, where groups are defined by some underlying network attribute.

T2 Diagnose the perturbation effects. What-if analysis [46] has previously been used for XAI as a mechanism to investigate machine learning model performance for a range of data features. In our framework, we adopt this idea of what-if analysis to measure the output of any graph-based ranking algorithm by perturbing the input. This enables model developers to measure and explore the ranking changes and corresponding effects. As we focus on **removing nodes** as the perturbation mechanism, a key analytical task is to support diagnosing changes caused by node removals including:

- **T2.1 Summarize the ranking influence of perturbation** The system should provide a summary of how perturbation has impacted the graph-based ranking results.
- **T2.2 Enable the ranking influence comparison between subgroups** Each graph node represents an instance and may have attributes/labels that can be used to define class membership. Questions about ranking changes are often strongly tied to questions of fairness related to graph attributes, for example, given a hiring database, are the ranking of female applicants more sensitive to changes in the graph structure than male applicants?
- **T2.3 Identify the topological influence caused by perturbations** The system also needs to support analysts in exploring how perturbations have influenced the graph topology.

T3 Enable progressive analysis. The system should support the analysis of multiple perturbations as analysts explore what-if scenarios.

3.2 Design Requirements

From the task requirements, we engaged in an agile design process with our domain experts, iterating over various visualization and interaction designs. Based on our discussions, prototyping and feedback, we have mapped different analytic tasks to a set of design requirements.

D1 Visualize the Instance-level Sensitivity. The system should visualize ranking and auditing results for all instances (T1). The view for summarizing the instance-level sensitivity should include the sensitivity index (T1.1) for all nodes with respect to the node attributes (T1.3).

D2 Visualize the Effect of Perturbation. The system should be able to guide analysts to explore the perturbation effect of certain node's removal and support interactions such as sorting, searching and filtering to inspect the auditing results and corresponding perturbation effects (T2, T3). This view should include:

- *D2.1 Influence Overview*, which summarizes the perturbation's influence, the degree of ranking changes, and the proportion of nodes whose rankings are increased/decreased, etc. (T1.2, T2.1, T3)
- *D2.2 Distribution View*, which shows how the ranking position changes caused by the perturbation are distributed for each instance and the ranking distribution for each group of nodes. (T2.2, T1.3)
- *D2.3 Ranking Change Detail View*, which lists the influenced nodes for this perturbation. The view should support basic query operations, e.g., sorting, filtering and searching, etc.(T2.1, T3)
- *D2.4 Local Influence Graph View*, which illustrates the relationship between the ranking changes of nodes and the topological changes caused by the perturbation. (T2.3, T3)

4 Visual Analytics Framework

Based on the analytic tasks and design requirements, we have developed a visual analytics framework (Figure 2) for auditing, diagnosing, and analyzing graph-based ranking methods' sensitivities to instance-level perturbation through what-if analysis. The framework is designed to first compute a sensitivity calculation for each node of the graph and integrate the results to form a list of all sensitivity information (Figure 2 (A)). Once the precomputation is loaded, the analyst interacts with the system by choosing nodes to perturb, and the framework calculates and visualizes the perturbation effects (Figure 2 (B)). The framework also supports filtering the list based on the analyst-defined rules to support the inspection of the data under a variety of constraints (Figure 2 (C)). By supporting an iterative process of perturbing nodes and adding analyst-defined rules, this framework enables the auditing of the sensitivity of graph-based ranking algorithms.

The framework supports three main activities: instance-level sensitivity identification, perturbation diagnosis, and customized constraints

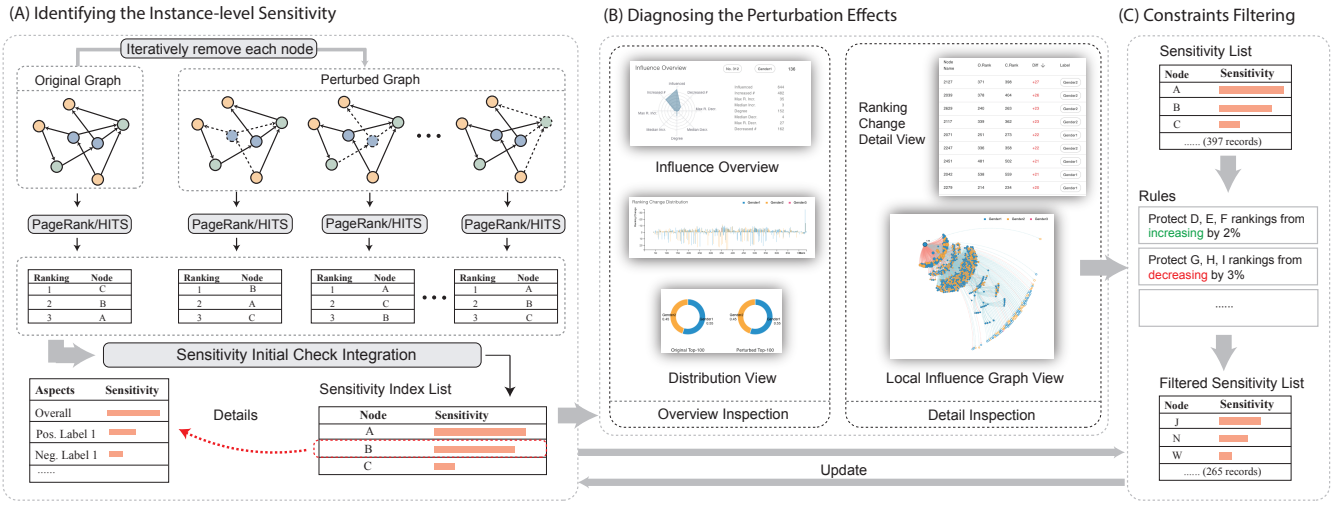

Fig. 2. A visual analytics framework for identifying, auditing, and diagnosing a ranking method's sensitivity to instance-level perturbations. The framework consists of Identifying the Instance-level Sensitivity, Diagnosing the Perturbation Effects, and Constraints Filtering. (A) Perturbation is applied for each node instance. The ranking methods are rerun for each new graph generated. Sensitivity to the ranking method for each node's removal is calculated. (B) The analyst can explore one of the nodes to see the perturbation effects through both overview inspection and detailed inspection. (C) Then the analyst can apply constraints to the sensitivity index list based on their findings, and the sensitivity index list is updated.

filtering. Through instance-level sensitivity identification, the analyst can explore an overview of ranking sensitivity with respect to perturbation (node removal). In the perturbation diagnosis, effects caused by the perturbation are displayed with respect to the statistical distribution, top-k distribution, and influenced paths. From the detailed influence view, the analyst can identify the potential constraints and further apply those constraints to filter the results. The analyst can repeat this process until they identify nodes of interest. Our framework is designed to be modular, enabling model designers to integrate any graph-based ranking algorithm. However, for discussion and demonstration purposes, we only explore PageRank and HITS.

4.1 Identifying the Instance-level Sensitivity

The first component of our framework is designed to support instance-level sensitivity analysis (or auditing) for nodes in the graph. Kang et al. [22] defined sensitivity auditing as finding the k graph elements (nodes or edges) that have the largest influence on the overall ranking changes of a given graph. This approach identifies the most influential element, removes this element, updates the graph structure, identifies the most influential element from the new graph structure, and continues repeating this process until k elements are found. While such a process is useful for identifying the most sensitive nodes, it does not directly incorporate a mechanism for measuring sensitivity for an individual node. In order to define sensitivity per node, we modify the definition of sensitivity auditing from Kang et al. [22].

Definition 1. *Given a graph G, an element (a node or an edge) to be removed el_{rm} and a graph ranking method f, the **graph ranking sensitivity auditing** can be defined as finding the **sensitivity index** for each graph element of G. We denote the sensitivity index of element el_{rm} as the degree of ranking method f's sensitivity to the perturbation caused by the removal of this element el_{rm}. The sensitivity index of this element el_{rm} is represented as:*

$$s[el_{rm}] = sen(f, el_{rm}) = L(rp, rp') \tag{3}$$

where rp and rp' denote the ranking positions for each node before and after the perturbation respectively (el_{rm} is not included in both rp and rp'), and L stands for a generic difference/distance measure between the ranking vectors before and after perturbation. $\mathbf{s}$ as the result represents the vector that contains the sensitivity index for every instance, and $s[el_{rm}]$ denotes the sensitivity index of el_{rm}.

Our proposed sensitivity index is used to help analysts compare sensitivities across removals of graph elements. As we focus on node removal in this work, el_{rm} can be replaced by v_{rm}, which denotes the removed node. While there are many metrics available for calculating L, we apply the L_1 norm as the sensitivity metric as it directly measures the accumulated ranking position changes over all nodes. Figure 2 (A) illustrates how the Instance-level sensitivity module performs the initial sensitivity index check on each instance. Applying Definition 1, our framework first calculates the ranking change for every node by removing each node (and its corresponding edges) from the graph and calculates the ranking method on the perturbed graph. The removal of a node may cause the ranking of other nodes to change in both ranking directions (increase or decrease). As such, the sensitivity can be summarized in multiple ways. For example, we could compute the overall positive/negative ranking change (or influence) that occurs when removing a node could be summarized or the class-specific positive/negative influence, where classes of nodes are defined based on their attributes and labels. In our framework, we calculate both a positive sensitivity index sen_p and a negative sensitivity index sen_n with respect to class labels as follows:

$$s^b_{pos}[v_{rm}] = sen_p(f, v_{rm}, b) = \begin{cases} \sum |rp_v - rp'_v|, rp_v - rp'_v > 0 \\ 0, otherwise \end{cases} \tag{4}$$

$$s^b_{neg}[v_{rm}] = sen_n(f, v_{rm}, b) = \begin{cases} \sum |rp_v - rp'_v|, rp_v - rp'_v < 0 \\ 0, otherwise \end{cases} \tag{5}$$

where $v \in V \wedge v \neq v_{rm} \wedge label(v) = b$, V is the set of all nodes in the graph, b is the class label, and rp_v and rp'_v are the ranking positions of node v before and after the perturbation respectively. $\mathbf{s_{pos}}$ and $\mathbf{s_{neg}}$ represent the vectors of positive and negative sensitive index for each instance, in which $s_{pos}[v_{rm}] = \sum_{b \in B} s^b_{pos}[v_{rm}]$ and $s_{neg}[v_{rm}] = \sum_{b \in B} s^b_{neg}[v_{rm}]$, where B is a set of labels. Eq. 3, Eq. 4 and Eq. 5, are applied in Algorithm 1 to realize the the initial sensitivity index check for every node and every available label. The system then visualizes the output in the sortable sensitivity index list (D1), which shows each node's current ranking and sensitivity indices with respect to the node's class label(s).

4.2 Diagnosing the Perturbation Effects

The second component of our framework is designed to support what-if analysis. In this module, analysts begin by selecting a node from the

sensitivity index list to further explore the perturbation effects. Several linked views are deployed:

Algorithm 1 Sensitivity Index Initial Check

Inputs: graph data G; ranking method f; node labels B;
Outputs: overall SI vector $\mathbf{s}$; positive/negative SI vector $\mathbf{s_{pos}}$/$\mathbf{s_{neg}}$; positive/negative SI vector in terms of labels $\mathbf{S_{pos}}$/$\mathbf{S_{neg}}$;
$\mathbf{r_{original}} \leftarrow f(G)$
for each node v in G.nodes **do**
  remove v and all connected edges **e** from G
  $\mathbf{r_{removed}} \leftarrow f(G)$
  calculate $\mathbf{r_{diff}}$ with $\mathbf{r_{original}}$ for each node in $\mathbf{r_{removed}}$
  for each $r_{diff}[i]$ in $\mathbf{r_{diff}}$ **do**
    if $r_{diff}[i] > 0$ **then**
      $s^b_{pos}[v] \leftarrow s^b_{pos}[v] + abs(r_{diff}[i])$ for b = label(i)
    end if
    if $r_{diff}[i] < 0$ **then**
      $s^b_{neg}[v] \leftarrow s^b_{neg}[v] + abs(r_{diff}[i])$ for b = label(i)
    end if
    $s[v] \leftarrow s[v] + abs(r_{diff}[i])$
  end for
  $s_{pos}[v] \leftarrow sum(s^b_{pos}[v])$; $s_{neg}[v] \leftarrow sum(s^b_{neg}[v])$ for b in B
  add all $s^b_{pos}[v]$ to $S_{pos}[v]$ and all $s^b_{neg}[v]$ to $S_{neg}[v]$ for b in B
  add v and **e** back to G
end for
Return $\mathbf{s}$, $\mathbf{s_{pos}}$, $\mathbf{s_{neg}}$, $\mathbf{S_{pos}}$, $\mathbf{S_{neg}}$

Influence Overview: The influence overview provides basic information on changes caused by removing a specific node (D2.1). These changes include 1) the number of influenced nodes which have ranking changes after the perturbation; 2, 3) the number of influenced nodes whose ranking increased/decreased after the perturbation; 4, 5) the max/min of increased/decreased ranking changes; 6, 7) the median of increased/decreased ranking changes; and 8) the degrees of the node. These 8 metrics provide the analyst with a statistical overview of the ranking influence of nodes due to perturbations. The radar chart is used to provide an overview of the sensitivity metrics with respect to the effects of a perturbation. (Figure 1 (3)) The radar chart allows for the further addition of new metrics and can also preserve the overall information of the perturbation when the analyst switches between multiple perturbation diagnoses through the tabs on the top of the view.

Influence Distribution View (D2.2): In addition to showing the snapshot of ranking changes, we also provide a ranking change distribution view and the top-k proportional distribution view.

Ranking Change Distribution View: A bar chart (Figure 3) is used to show the ranking change distribution. Each bar is a node. The position of the bar on the x axis denotes the original ranking position for the node. The height of the bar on the y axis denotes the ranking change for the node. Colors represent the node labels. We scale the axes of the bar chart such that a 90-degree clockwise rotation of the bar also allows the analyst to infer the future rank of the node.

Top-k Proportional Distribution View: Chartier et al. [11] noted that when applying graph-based ranking algorithms for search engines, there could be an argument that ranking changes of webpages that are not part of the top-k ranking are less important than those in the top-k. This argument can be extended to any general graph ranking problem, where the analyst can choose a k for which elements below this ranking will not be considered. For example, a hiring manager may not be interested in resumes ranked outside of the top-25, but it important to understand whether certain node attributes are underrepresented in the top-k. In the Top-k Proportional Distribution View, we use two donut charts to represent the proportions of nodes of different categories belonging to ranking 1 to ranking k before and after the perturbation (Figure 1 (5)), and k is interactively specified.

Influence Detail View: In the influence detail view, a data table is used to give the exact details about the node name, previous ranking, perturbed ranking, ranking difference, and labels. The analyst can sort all columns in the table, and, by hovering over a row, the location of the corresponding node will be highlighted in the local influence graph.

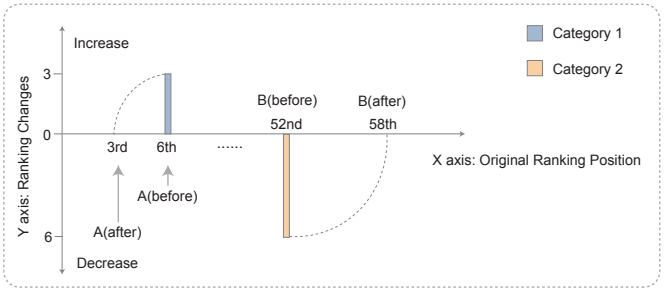

Fig. 3. Visual encoding of the Ranking Change Distribution View. We use a bar chart to show the ranking change distribution. Each bar is a node. The position of the bar on the x axis denotes the original ranking position for the node. The height of the bar on the y axis denotes the ranking change for the node. Colors represent the node labels. We scale the axes of the bar chart such that a 90-degree clockwise rotation of the bar also allows the analyst to infer the future rank of the node.

Influence Graph View: While summarizing the changes in rank is important, our domain experts also required the ability to explore the impacts on the graph topology caused by perturbations. Note that for the graph-based ranking algorithm, such as PageRank and HITS, the underlying logic propagates the ranking value until convergence. In other words, the probability value for each node is propagated via a directed link connected between a node and its predecessors [1]. In our case, perturbation is equivalent to removing the designated node and the links associated with it. As such, perturbation may cause two types of influence: *direct influence* and *indirect influence*. We consider direct influence to be the ranking changes of nodes due to the perturbation of their predecessors. Indirect influence is the ranking changes of nodes due to perturbations of any nodes that are not their predecessors. Understanding the direct and indirect influence is critical as analysts are interested in nodes that cause limited direct influence but have larger amounts of indirect influence. These types of nodes are prime candidates for shadowing banning and other types of attacks as their removal can greatly influence the ranking results, but the removal will be relatively unnoticed by their direct connections, making such an attack difficult to quickly identify.

To perform the direct/indirect influence analysis, we begin by building the influence graph which contains the topological relationships between the influenced nodes and the removed node. Here we introduce the concept of *influence distance*, which we define as the geodesic distance [2] between two nodes. We denote the influenced nodes, which are successors [3] of the removed node, as *hop-1* influenced nodes, which are also represented as the directly influenced nodes. Hop-1 influenced nodes have an influence distance of 1. Similarly, from the influenced nodes, we denote the hop-1 influenced nodes' successors as hop-2 influenced nodes, which have an influence distance 2, and so on and so forth. We define the influence distance to be the shortest distance between the removed node and the influenced node, which means that a hop-1 influenced node may also be a hop-3 influenced node's successor, but we only consider it as hop-1. Finally, for those nodes whose influence distance is infinite, we denote them as hop-inf influenced nodes. Algorithm 2 details our influence graph construction algorithm.

We visualize the influence caused by removing/perturbing a node (D2.4) as a customized radial graph layout (Figure 4). In this customized layout, the removed node is set as the center of the force, and the strength of the charge force for each type of the node (hop-1 node,

[1] A node that has a link that points to a given node in a directed path.
[2] The length of the shortest directed path connecting the two nodes. For an unweighted graph, the length is the number of edges in the shortest path. The distance is *infinite* if there is no path between two nodes.
[3] Any node whose geodesic distance is equal to one from a path starting at i.

Algorithm 2 Influence Graph Construction

Inputs: graph data G; removed node v_r; influenced nodes V'
Outputs: influence graph G'
initialize G'
initialize queue **q**
G'.addNode(v_r, hop=0)
q.push(v_r)
while q is not empty **do**
    $v \leftarrow$ **q**.pop()
    for $v_{neighbor}$ in v.neightbors() **do**
        if $v_{neighbor}$ is influenced and not visited **then**
            visited($v_{neighbor}$) $\leftarrow$ true
            G'.addNode($v_{neighbor}$, hop=v.hop + 1)
            G'.addEdge(v, $v_{neighbor}$)
            q.push($v_{neighbor}$)
        end if
    end for
end while
if any influenced node not in G' **then**
    for v_{remain} in remained influenced nodes **do**
        G'.addNode(v_{remain}, hop=inf)
        G'.addEdge(v, v_{remain})
    end for
end if
Return G'

hop-2 node, etc.) is increased gradually based on the number n of hop-n. In this way, all the nodes are clustered, and the influence graph forms a tree-like structure where the root of the tree starts from the top-left of the view and the branches spread towards the bottom-right of the view. Compared with a traditional force directed layout, this layout has two advantages: 1) All the influenced nodes are organized and positions of the nodes are relatively fixed in this layout, which enables the analyst to preserve their mental model. 2) Nodes of the same type are clustered in this view, enabling the analyst to explore the composition for each cluster, i.e., each group of hop-n nodes. The color of the edges is encoded with light blue and orange, which shows whether the influenced node increased/decreased. The nodes' colors encode their categories (using colored filling and black stroke for regular nodes, and white filling and colored stroke for hop-inf nodes), and the nodes' sizes and thickness of their incoming edges encode the absolute value of the ranking changes of nodes. We also provide interactive graph filtering so that analysts can filter by nodes whose rankings have increased/decreased, or nodes within certain influence distance ranges. The ranking change distribution view will also automatically update based on the ranges, Figure 1 (6) (7). By filtering, the analysts can answer questions related to the perturbation, such as whether the node has a large direct influence on specific categories, or whether the node has a large indirect influence on nodes that are far away.

4.3 Customized Constraints Filtering

As we hinted at with the discussion of exploring ranking changes with respect to node attributes, in real-world applications, measures of sensitivity may need to be done with respect to domain specific constraints. Consider the Google search engine as an example, where each website is more concerned about reaching the first page of search results as well as climbing the ranking on that first page. It is reported that Google traffic is captured by 91.5% of the first page search results [1], and there are also reports suggesting that top-3 results capture upwards of 75% of the clicks [2]. In such a setting, a domain owner would be interested in how sensitive their website is to ranking changes as a change in ranking from the first page of the search results to the second can have disastrous implications for web traffic.

Our framework enables customized constraints filtering functionality that allows analysts to add constraints to the sensitivity index list. Specifically, the analyst can define constraints that prevent selected nodes' rankings from increasing/decreasing by a certain degree. As Figure 2 (C) shows, the analyst may have domain knowledge of the data and may wish to add constraints to the sensitivity index list to filter

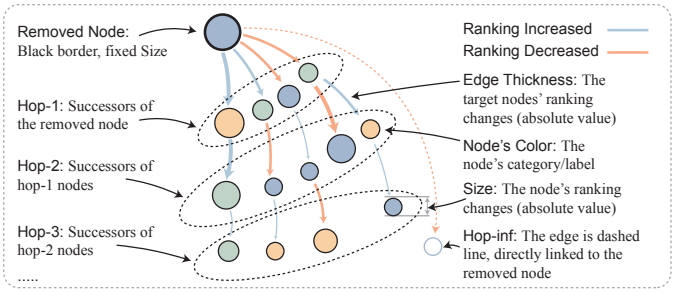

Fig. 4. Visual encoding of the Influence Graph View. The perturbed node is placed at the left top of the graph and all successors which have ranking changes are grouped as hop-1 nodes. The successors of any hop-1 nodes which have ranking changes are grouped as hop-2 nodes, so on and so forth. The nodes' colors encode their categories (using colored filling and black strokes for regular nodes, and white filling and colored strokes for hop-inf nodes), and the nodes' sizes and thickness of their incoming edges encode the absolute value of ranking changes of nodes. The orange and light blue edge colors denote a ranking decrease/increase respectively. Nodes that do not have connections with any of the nodes in the influence graph are hop-inf nodes and are connected to the perturbed node with a dashed arrow line.

out any perturbations that would violate the constraints. For example, if the analyst wants all the possible perturbations on the sensitivity index list that do not cause the top-3 nodes to experience ranking drops, the analyst can add a constraint: prevent top-3 nodes from ranking decreased by 0. The analyst can then sort the sensitivity index list and add the top-3 ranked nodes to the protected list by clicking the shield-like button for each of them and then configure a new rule "protect selected nodes from ranking decreased by 0". Finally, the analyst clicks the *Update Constraints* button to add the new rule. The newly configured rule will then be displayed on the *Rules* section (Figure 1 (1)) and the sensitivity index list will be automatically updated such that any potential perturbations in it will not result in the top-3 ranked nodes having their rank decreased. The analyst can also add more constraints as they may find certain nodes need to be protected from the perturbation during the diagnosis process. For example, if the analyst finds that a perturbation causes a significant ranking drop on the node, the analyst can use the lasso tool to select the node in the influence graph view (the node with significant ranking changes is encoded as a large circle in this view), and add it to the protected nodes and then apply a new rule. The analyst can then restart exploring the potential perturbations that do not violate the new rule. In this way, the analyst is able to explore possible perturbations under a variety of customized constraints.

5 Case Study and Expert Review

In this section, we present three case studies to demonstrate how our framework supports sensitivity auditing for graph-based ranking. We showcase how data scientists analyze the sensitivity of the PageRank algorithm on Facebook social network data, how ranking developers check the robustness of their graph-based ranking algorithms, how social scientists analyze the exposure of blogs, and how the sensitivity of ranking algorithms can help identify potential manipulations.

5.1 Facebook Ranking with PageRank

In social network analysis, ranking members based on the graph structure is essential to tasks such as advertising [19], social link prediction [15], and recommendation [16]. Perturbations in rankings may have a significant influence on the related business strategies. As such, it is important to audit the sensitivity of such rankings as this may help uncover malicious accounts, or reveal unintended biases in the algorithm. For example, analysts employ graph-based ranking methods to provide recommendations within a social network. These recommendations are based on the ranking results which predict who in the social network might be interested in the recommended products. Here, it may be important to understand if one sub-population is being over-

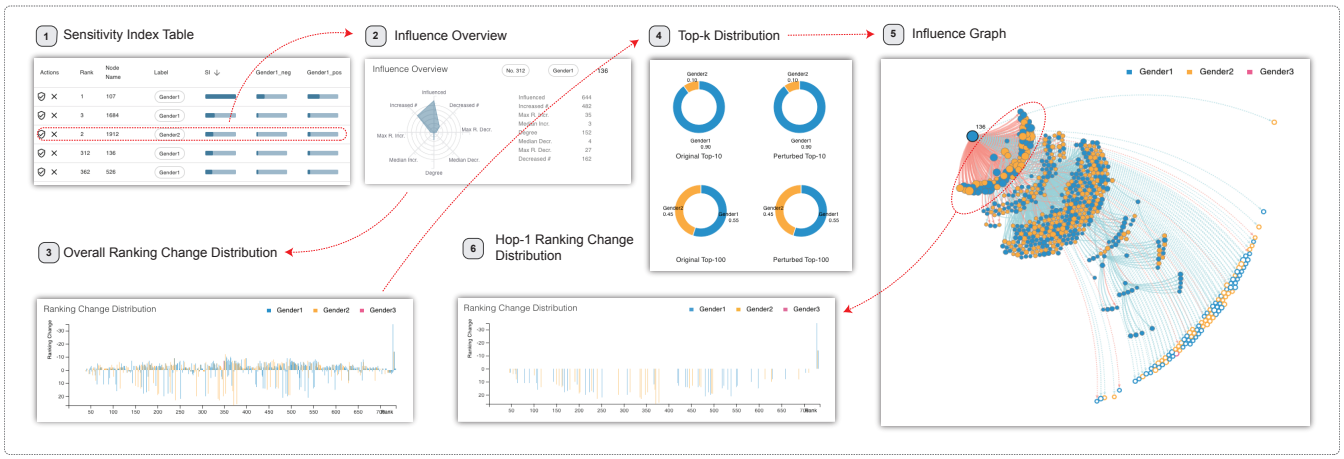

Fig. 5. Facebook sensitivity analysis on PageRank. (1) The sensitivity index list shows that *user 136* has the 4th largest influence on the Sensitivity Index while its ranking score is 312 out of 734 nodes, which is considerably low for such a large influence. (2) The influence overview shows that the removal of *user 136* influences 644 of the 734 nodes, with 482 of them being positively influenced (ranking increased) while 162 are negatively influenced (ranking decreased). (3) The ranking change distribution view shows that negatively influenced nodes see a larger ranking decrease in comparison to positively influence nodes. (4) The top-k distribution view shows that as k increases from 10 to 100, the proportion of gender 2 increases, which means nodes of gender 1 are ranked relatively higher than nodes of gender 2. (5) The influence graph view further explains that most of those nodes who have large ranking changes are the neighbors of the removed *user 136*, and (6) the corresponding distribution view shows that those neighbors' rankings are evenly distributed.

targeted with particular advertisements. In this case study, we analyze ranking sensitivity in the Facebook social network dataset [29]. For demonstration purposes, the social network is down-sampled into a graph with 734 nodes and 74254 edges. We use the gender of each network member (where each member is a node in the graph) as the class label and explore if perturbations in the network reveal ranking bias with respect to gender.

Identifying the Instance-level Sensitivity (T1): The sensitivity index table is displayed after loading the graph data and choosing the ranking method, Figure 5. We sort the column *SI (Sensitivity Index)* in order to explore which node (network member) is likely to have the largest influence on the ranking result. After sorting by SI, it can be observed that the nodes with the highest SI are also the highest ranked nodes. This phenomenon matches the explanation in Kang et al. [22] that the nodes with high influence are also often highly ranked. Interestingly, though, we see that the 4th node, *user 136*, has a high sensitivity index. However, the network member is ranked 312th out of 734 nodes. This particular case is of keen interest to our analyst.

Diagnosing the Perturbation Effects (T2): By clicking 'diagnose the perturbation effect' on *user 136*, the details of the influence is depicted in the influence overview. In Figure 5 (2), the influence overview shows that the perturbation caused by removing *user 136* has influenced 644 out of 734 nodes, where 482 nodes' rankings increased and 162 nodes' rankings decreased. By further exploring the ranking change distribution view, we find that although positively influenced nodes are 3 times more common than the negatively influenced nodes, the negatively influenced nodes are subject to larger ranking fluctuations than the positively influenced nodes. While we expect that the nodes that were previously ranked lower than *user 136* would see a rise in ranking to fill the gap created by *user 136*, it is surprising to also find large negative changes occurring. In the ranking distribution view and the top-*k* distribution view, Figure 5 (3), we observe that the perturbation does not result in drastic changes in the ranking distributions. Furthermore, if we split the top-*k* rankings by gender, the perturbation is not observed to impact one gender class's rankings more than another.

In addition, our analyst is interested in the relationships between the ranking changes and the topological structure. Specifically, the analyst wonders how the removed *user 136* is connected to the influenced nodes given the stronger impacts for decreased ranking. In the local influence graph view, the nodes who have significant ranking changes in the perturbation are directly connected to *user 136* since all the large circles are hop-1 node, Figure 5 (4). After selecting the range of influence distances, we can see that the decreases of rankings for most of the nodes are caused by being the immediate successor of *user 136*. Furthermore, most of the nodes outside the hop-1 circle have their ranks slightly increased. This may be due to the fact that those nodes are not heavily influenced by the perturbation.

5.2 Subreddit Ranking with PageRank

In the second case study, we audit the sensitivity of PageRank results on the subreddit community interaction graph dataset [32]. Kumar et al. [25] studied the community interactions and conflicts between communities in Reddit to show that a community can be mobilized by negative sentiment comments from another community. Such conflicts between communities can potentially reduce the activities among community members and may lead to people leaving the platform. In such cases, it is also possible that other communities will be influenced due to chain effects in the network, which may cause member churn and increased complaints about the platform. Thus, the Reddit community managers and data analysts may be interested in inspecting the activities of communities to make sure the content environment is benign and also be aware of any perturbations (deletion of subreddit posts/activity reduction) that might influence ranking results during any possible recommendation processes.

In this case, we sub-sampled the subreddit community interaction dataset [25] down to 464 nodes and 6676 edges. Each node represents a subreddit, and a topic label is assigned to the nodes to identify their main categories, such as `r/basketball`, `r/soccer` and `r/nfl` under the topic "Sport". An edge between two nodes indicates there is a comment in one subreddit which referred to the other subreddit in the content of the comment. We select three topics to explore: Sports, General, and Others. In this case, comments with a high page rank would receive more views, and if negative comments generate more clicks, this could be of concern. Community administrators could utilize our framework to explore the interactions between communities and identify potential issues of flaming, karma farming, etc.

Identifying the Instance-level Sensitivity (T1): After the ranking results are loaded into the system, we want to know which subreddit in the "Sports" topic has the largest influence on the reputation of other subreddits. By sorting the sensitivity index column, the ranking result is listed in the sensitivity index table, and the subreddits are ordered by their sensitivity values in descending order. We believe that concrete entities such as sports players and teams can have more

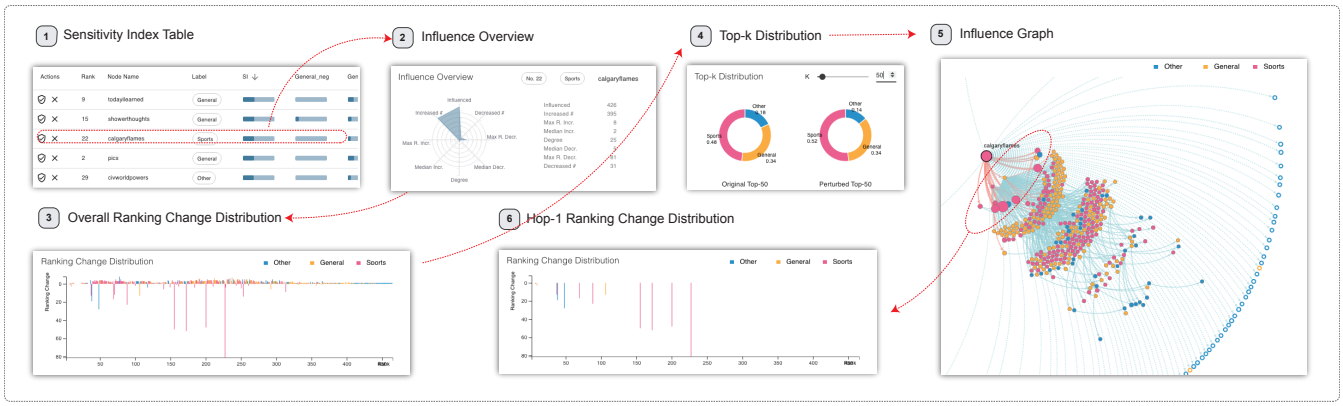

Fig. 6. Subreddit sensitivity analysis on PageRank. (1) The sensitivity index list shows that the `r/CalgaryFlames` node belongs to the sports topics and has a large sensitivity index. (2) The influence view shows that removal influenced 426 out of 464 nodes, and 395 of them are positively influenced while only 31 are negatively influenced. (3) The ranking change distribution view shows that a few nodes experience a large ranking decrease due to the removal of `r/CalgaryFlames`, the majority of which are sports topics. (4) The top-k distribution view shows that the subreddits for sports occupy more of the top-50 ranks than other subreddit topics. (5) (6) The removed node has relatively few neighbors that are negatively influenced, and there are more direct influences in the 'General' nodes than 'Other' nodes.

controversial topics, but are less noticeable by community members who are not interested in them. Thus, we skip the general subreddits such as `r/hockey, r/baseball, and r/basketball` and focus on the `r/CalgaryFlames`, which has a relatively large influence on the reputation rankings of the subreddits.

Diagnosing the Perturbation Effects (T2): The impact of removing the `r/CalgaryFlames` node are shown on the right side of Figure 6. In the influence overview, the removal has influenced 426 out of 464 nodes. Among the 426 nodes, 395 of them have their ranks increased while 31 decreased. Specifically, the node `r/thebeach` has increased by 8 positions, and the rank of node `r/coloradoavalanche` has dropped by 81 positions. This indicates that the perturbation has triggered massive declines even though the average increase is relatively low. In the ranking change distribution view, we observe that the ranking declines occur primarily in the Sports subreddits whose original ranks are relatively low (around 255 out of 464). Compared with the decrease, the overall distribution of the increased nodes covers a broader range on the original rankings; however, a much smaller climbing effect in the ranking positions is observed. This may be due to the fact that the original rank of the `r/CalgaryFlames` is very high (22nd), which could possibly bump many lower-ranked subreddits into higher positions. In the distribution view, all three topics (sports, general and others) receive slight increases in the median values. However, the overall distribution remains the same. We further query the top-50 since there are considerable ranking changes depicted in the ranking change distribution view. This suggests that after removing the `r/CalgaryFlames`, the proportion of subreddits in the category of Sports among the top-50 has increased from 48% to 52%, while the proportion of "Other" subreddits has dropped from 18% to 14%. In the influence graph view, we find that the removal results in large drops the to ranks of the hop-1 nodes, which matches the patterns shown in the ranking change distribution view. That is to say, the removal of `r/CalgaryFlames` only influences its neighbors with significant ranking changes. By filtering on the influence distance values, we find that the perturbation significantly influences the "Sports" subreddits.

5.3 Political Blogs Ranking with HITS

Graph-based ranking methods are widely used to rank webpages. Here, we consider a scenario where removing certain pages from a website (either intentionally by the website owner, or maliciously through shadow bans by an external party) can significantly change the rankings of other pages. Consider a political web forum where members post views and opinions on certain topics and issues. The search result rankings are determined by a graph ranking algorithm, and higher-ranked opinions are more likely to be read and shared. Here, one could imagine a nation-state actor that would want to promote biased content. The nation-state actor can create webpages to add various links in the graph structure, or even identify websites to shadow ban, which could manipulate the ranking results so that certain political opinions are more exposed to the public. By using the articles in the forum as the nodes and the hyperlinks between different articles as edges, the graph ranking methods can recommend popular articles in the forum based on the graph structure. However, there could be some nodes that are vulnerable and have high sensitivity indices with respect to the graph ranking method. They become the target for the attackers who wish to manipulate the article rankings and promote their own content. As such, blog managers and social scientists may wish to collaborate with ranking algorithm developers to make sure such ranking results are fair and stable with respect to potential perturbations (deletion of blogs).

In this case study, we explore the sensitivity of the HITS algorithm on the political blog dataset [24]. The dataset includes a topic citation graph between liberal and conservative blogs prior to the 2004 U.S. Presidential Election. We subsampled 397 nodes (i.e, blogs) and 12,365 edges (i.e., hyperlinks between blogs), removing all nodes with degree less than 30. Our goal was to explore the structural changes in the network that result in drastic ranking changes in the HITS algorithm.

Identifying the Instance-level Sensitivity (T1, T3): Before the analysis, we made three assumptions about ranking manipulation: 1) the top-k items in the ranking are much more important than other nodes since readers typically only view the top results provided by the search engine. 2) It is riskier to manipulate a node with a higher rank since the readers may notice the changes. 3) To avoid having manipulations discovered, the attacker would assume a posture of minimum risk. As such, our goal is to discover how we can manipulate the ranking results by removing a node, while working under these constraints.

Based on our constraints, we apply selection rules to filter the sensitivity index table. We first click the ranking column to sort the ranking order. Then, we add the top-5 nodes to the protected nodes with constraints of *protect selected nodes from their ranking decrease by 0%*, which excludes all perturbations that would cause the rankings of these selected nodes to decrease. The constrained sensitivity index table is shown on the left, which contains 1) the ranking positions, 2) node names, 3) overall sensitivities, and 4) sensitivity details including positive/negative influence to liberal/conservative blogs. After sorting the rows by the sensitivity index column in decreasing order, a liberal blog, `liberaloasis.com` appears in the second row of the table. By observing the other columns, we find that `liberaloasis.com` is not in the top-10 rank; however, its removal can increase the rankings of the conservative blogs while decreasing the rankings of the liberal blogs.

Diagnosing the Perturbation Effects (T2): Next, we explore why this blog is so influential. After selecting "Explore the Perturbation in Detail"

by clicking the shield button in the first column, all the details are listed on the right side of the interface (Figure 1). In the influence view, the radar chart indicates that there are 368 out of 397 nodes influenced by the removal of `liberaloasis.com`. 232 of the 368 nodes have their ranks increased while 136 decreased. The largest increase in the ranking positions is 16, and the largest decrease is 30. From the ranking change distribution (Figure 1 (6)), we find that most of the ranking changes happen in the range between 50 and 150. Since only mid-tier ranks are impacted, the effects of removing this node are subtle, meaning this change is not easily observable. However, the impact is significant. In the distribution changes view, we can observe that the median of the liberal blog ranking distribution decreases, while the conservative blog ranking distribution increases. By further exploring the proportions of both liberal and conservative blogs in the top-100 results, Figure 1 (5), we can see that the proportion of liberal blogs in the top-100 results has decreased by 5% (from 82% to 77%), thus subtlety shifting the site's content.

In the ranking change distribution view, we also find that there are three liberal blogs with considerable ranking decreases after the perturbation. We further check the detailed view for information on the influenced nodes. We sort the original column to locate the exact ranking position and notice that the first three liberal blogs, `sununes`, `boloboffin`, and `elemming2`, have a large ranking decrease after the perturbation, which corresponds to the three liberal blogs in the ranking change distribution view mentioned above. We want to further explore the relationship between the ranking changes with the topological structure. The influence graph view (Figure 1 (8)) shows that the removed node influences a majority of liberal nodes, and as the influence distance increases, conservative nodes are indirectly influenced.

5.4 Expert Review

Along with our case studies, we have conducted a group interview with our collaborator (E0) and three additional domain experts in graph mining (E1, E2, and E3) to provide feedback on our framework. We began the interview by introducing our visual analytics framework, and the functionalities supported by each module. Then, we presented a demo of the analytical flow across the three previously described case studies. After this, the experts are allowed to freely explore the perturbation results of the three datasets (Facebook, Reddit, and Polblogs) over two ranking methods (Pagerank and HITS) in our system. The interview lasted approximately 90 minutes, and we collected free-form responses to the following questions:

1. Does the system meet the design requirements and address the analytical tasks proposed in our work?
2. Does our analytical pipeline match your daily workflow?
3. How is the information delivered through our system?
4. How would you perform the same tasks in conventional graph mining methods?

Framework: We received positive feedback from the experts in terms of our proposed framework. The experts found the framework to be practical with respect to the proposed problems. E1 appreciated that the framework is capable of handling the sensitivity issues that are related to nodes' attributes, and noted that such a framework can support not only graph-ranking developers but also experts in other fields evaluating whether the ranking algorithm is suitable for real-world ranking tasks. E2 appreciated our constraint filtering functionality since such an iterative analysis is one of his preferred approaches.

Visualization: The overall visualization techniques were also well received. E1 mentioned: *"With the knowledge of understanding what the system is capable of doing, I found most of the visualizations are straightforward and easy to understand. The newly designed influence graph is also intuitive once I learned what each encoding means."* E2 also appreciated that the interactive effect is helpful for understanding the ranking change effects after the perturbation. E3 suggested that we could add question-mark icons that link to descriptions for each view.

Limitations: The experts also identified several limitations of our current framework. E1 and E2 noticed that there is only one perturbation method supported in our system, which is the node removal. However, they all understood that the perturbation spaces (edge removal, node/edge addition) are far larger than the node removal space. E3 also mentioned that the visualizations may become crowded when the size of the influence graph is large.

6 Conclusions and Future Work

In this work, we propose a visual analytics framework for auditing the sensitivity of graph-based ranking methods. By analyzing the influence of the ranking method due to a node's removal, analysts can diagnose the perturbation effects in terms of ranking changes. The system is targeted for graph ranking developers and graph-related domain experts. The framework is implemented using D3 for the visual components and Python 3 (the NetworkX library [18]) for data processing. All source code is provided in Github [4].

Scalability: The computation of the initial sensitivity check is the major bottleneck of our framework. This process requires re-computation of ranking results, calculating the ranking changes of all nodes, and pre-computing the statistical data for each perturbation. Even though we have pre-computed the required data for the visualization to mitigate the real-time computation burden, the overall preparation time of the initial sensitivity check is $O(n^2m)$, where n is the number of nodes and m is the number of edges. In our case studies, it takes approximately 2 minutes 30 seconds to pre-compute the Reddit dataset with 464 nodes and 6676 edges, 3 minutes for the Polblogs dataset with 397 nodes and 12365 edges and 25 minutes 30 seconds for the Facebook dataset with 734 nodes and 74254 edges. When a large graph is loaded, the computation time for processing the data can be prohibitively long on commodity hardware. This is the reason that we decided to subsample our two datasets instead of exploring the whole dataset. The corresponding effect is that the ranking result and sensitivity result may be different from the real dataset. Future work will explore parallel solutions and novel methods for further sensitivity calculations.

Visual Design: The visual design for the sensitivity index list is a bar representation for each sensitivity column. The advantage is that the analyst can sort every column, which allows them to quickly identify sensitivity aspects of interest. However, when a node has multiple labels, the sensitivity check for those aspects must also be displayed as columns, which consumes more space and reduces the ability of the analyst to quickly browse the results. Alternatives include re-designing the sensitivity check into glyphs so that the analyst can compare patterns and we will save display real estate. Another design issue occurs in the influence graph view, where the edge length is defined as the influence distance. Here we only consider a fixed number of influence distances, 9. This means that for large graphs, the display space will be limited.

Future work: In this work, we only audit sensitivity based on one perturbation strategy, node removal. In reality, there are also other types of perturbations of the graph, including adding/removing edges and adding nodes, where manipulating edges can be considered as establishing a connection/follow/like/cite other nodes, and adding nodes is a more subtle way of perturbation since it is often easier to create an account/website/user, etc. However, adding graph elements is computationally expensive as additions make the perturbation space infinite. Possible solutions would be limiting the perturbation space to a constrained budget. However, it is important to note that the addition of graph elements could be supported within the proposed framework. Our future work will target those types of perturbations.

Acknowledgments

This work was supported by the U.S. Department of Homeland Security under Grant Award 2017-ST-061-QA0001 and 17STQAC00001-03-03, and the National Science Foundation Program on Fairness in AI in collaboration with Amazon under award No. 1939725. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

[4] https://github.com/VADERASU/auditing-sensitivity-graph-ranking

REFERENCES

[1] How valuable is the first page of google? https://yourwebedge.com/valuable-first-page-google/. (Accessed on 02/28/2020).

[2] We analyzed 5 million google search results. here's what we learned about organic ctr. https://backlinko.com/google-ctr-stats. (Accessed on 02/28/2020).

[3] L. A. Adamic and N. Glance. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pp. 36–43. ACM, 2005.

[4] B. Alsallakh, L. Micallef, W. Aigner, H. Hauser, S. Miksch, and P. J. Rodgers. Visualizing sets and set-typed data: State-of-the-art and future challenges. In *Eurographics Conf. Visualization, EuroVis 2014 - State of the Art Reports, STARs*, 2014. doi: 10.2312/eurovisstar.20141170

[5] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. The state of the art in visualizing dynamic graphs. In *EuroVis (STARs)*. Citeseer, 2014.

[6] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. A taxonomy and survey of dynamic graph visualization. In *Computer Graphics Forum*, vol. 36, pp. 133–159. Wiley Online Library, 2017.

[7] M. Behrisch, T. Schreck, and H. Pfister. Guiro: User-guided matrix reordering. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):184–194, Jan 2020. doi: 10.1109/TVCG.2019.2934300

[8] A. Bougouin, F. Boudin, and B. Daille. TopicRank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pp. 543–551. Asian Federation of Natural Language Processing, Nagoya, Japan, Oct. 2013.

[9] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. 1998.

[10] M. Cavallo and . Demiralp. Clustrophile 2: Guided visual clustering analysis. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):267–276, 2019.

[11] T. P. Chartier, E. Kreutzer, A. N. Langville, and K. E. Pedings. Sensitivity and Stability of Ranking Vectors. *SIAM Journal on Scientific Computing*, 33(3):1077–1102, 2011. doi: 10.1137/090772745

[12] J. Chen, Z. Shi, Y. Wu, X. Xu, and H. Zheng. Link prediction adversarial attack, 2018.

[13] J. Chen, Y. Wu, X. Xu, Y. Chen, H. Zheng, and Q. Xuan. Fast gradient attack on network embedding, 2018.

[14] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song. Adversarial attack on graph structured data, 2018.

[15] D. F. Gleich. Pagerank beyond the web. *SIAM Review*, 57(3):321–363, 2015.

[16] M. Gori and A. Pucci. Itemrank: A random-walk based scoring algorithm for recommender engines. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pp. 2766–2771. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[17] S. Gratzl, A. Lex, N. Gehlenborg, H. Pfister, and M. Streit. LineUp: Visual analysis of multi-attribute rankings. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2277–2286, 2013. doi: 10.1109/TVCG.2013.173

[18] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using networkx. In G. Varoquaux, T. Vaught, and J. Millman, eds., *Proceedings of the 7th Python in Science Conference*, pp. 11 – 15. Pasadena, CA USA, 2008.

[19] J. Heidemann, M. Klier, and F. Probst. Identifying key users in online social networks: A pagerank based approach. 2010.

[20] J. Jia, B. Wang, X. Cao, and N. Z. Gong. Certified robustness of community detection against adversarial structural perturbation via randomized smoothing, 2020.

[21] J. Kang and H. Tong. N2n: Network derivative mining. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, pp. 861–870. ACM, New York, NY, USA, 2019. doi: 10.1145/3357384.3357910

[22] J. Kang, M. Wang, N. Cao, Y. Xia, W. Fan, and H. Tong. Aurora: Auditing pagerank on large graphs, 2018.

[23] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *JOURNAL OF THE ACM*, 46(5):604–632, 1999.

[24] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604632, Sept. 1999. doi: 10.1145/324133.324140

[25] S. Kumar, W. L. Hamilton, J. Leskovec, and D. Jurafsky. Community Interaction and Conflict on the Web. pp. 933–943, 2018. doi: 10.1145/3178876.3186141

[26] B. C. Kwon, B. Eysenbach, J. Verma, K. Ng, C. De Filippi, W. F. Stewart, and A. Perer. Clustervision: Visual Supervision of Unsupervised Clustering. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):142–151, 2018. doi: 10.1109/TVCG.2017.2745085

[27] O. Kwon, T. Crnovrsanin, and K. Ma. What would a graph look like in this layout? a machine learning approach to large graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):478–488, 2018.

[28] O.-H. Kwon and K.-L. Ma. A deep generative model for graph layout. *IEEE Transactions on Visualization and Computer Graphics*, p. 11, 2019. doi: 10.1109/tvcg.2019.2934396

[29] J. Leskovec and J. J. Mcauley. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*, pp. 539–547, 2012.

[30] M. Lu, Z. Wang, and X. Yuan. TrajRank: Exploring travel behaviour on a route by trajectory ranking. *IEEE Pacific Visualization Symposium*, 2015-July:311–318, 2015. doi: 10.1109/PACIFICVIS.2015.7156392

[31] A. Y. Ng, A. X. Zheng, and M. I. Jordan. Link analysis, eigenvectors and stability. In *International Joint Conference on Artificial Intelligence*, vol. 17, pp. 903–910. Lawrence Erlbaum Associates Ltd, 2001.

[32] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[33] C. Shi, W. Cui, S. Liu, P. Xu, W. Chen, and H. Qu. Rankexplorer: Visualization of ranking changes in large time series data. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2669–2678, 2012. doi: 10.1109/TVCG.2012.253

[34] A. Singh and T. Joachims. Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD 18, p. 22192228. Association for Computing Machinery, New York, NY, USA, 2018. doi: 10.1145/3219819.3220088

[35] R. Singh, J. Xu, and B. Berger. Pairwise global alignment of protein interaction networks by matching neighborhood topology. In *Annual International Conference on Research in Computational Molecular Biology*, pp. 16–31. Springer, 2007.

[36] M. Sun, J. Tang, H. Li, B. Li, C. Xiao, Y. Chen, and D. Song. Data Poisoning Attack against Unsupervised Node Embedding Methods. 2018.

[37] M. Sun, J. Zhao, H. Wu, K. Luther, C. North, and N. Ramakrishnan. The effect of edge bundling and seriation on sensemaking of biclusters in bipartite graphs. *IEEE Transactions on Visualization and Computer Graphics*, 25(10):2983–2998, 2019. doi: 10.1109/TVCG.2018.2861397

[38] T. Von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. J. van Wijk, J.-D. Fekete, and D. W. Fellner. Visual analysis of large graphs: state-of-the-art and future research challenges. In *Computer graphics forum*, vol. 30, pp. 1719–1749. Wiley Online Library, 2011.

[39] R. Vuillemot and C. Perin. Investigating the Direct Manipulation of Ranking Tables for Time Navigation. pp. 2703–2706, 4 2015. doi: 10.1145/2702123.2702237

[40] E. Wall, S. Das, R. Chawla, B. Kalidindi, E. T. Brown, and A. Endert. Podium: Ranking Data Using Mixed-Initiative Visual Analytics. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):288–297, 2018. doi: 10.1109/TVCG.2017.2745078

[41] X. Wang, M. Cheng, J. Eaton, C.-J. Hsieh, and F. Wu. Attack graph convolutional networks by adding fake nodes, 2018.

[42] Y. Wang, Z. Jin, Q. Wang, W. Cui, T. Ma, and H. Qu. DeepDrawing: A Deep Learning Approach to Graph Drawing. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2019. doi: 10.1109/tvcg.2019.2934798

[43] Y. Wang, M. Xue, Y. Wang, X. Yan, B. Chen, C.-W. Fu, and C. Hurter. Interactive Structure-aware Blending of Diverse Edge Bundling Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1–1, 2019. doi: 10.1109/tvcg.2019.2934805

[44] D. Weng, R. Chen, Z. Deng, F. Wu, J. Chen, and Y. Wu. SRVis: Towards Better Spatial Integration in Ranking Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):459–469, 2019. doi: 10.1109/TVCG.2018.2865126

[45] J. Weng, E.-P. Lim, J. Jiang, and Q. He. Twitterrank: Finding topic-sensitive influential twitterers. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pp. 261–270. ACM, New York, NY, USA, 2010. doi: 10.1145/1718487.1718520

[46] J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Viegas, and J. Wilson. The What-If Tool: Interactive Probing of Machine Learning

Models. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):56–65, 8 2019. doi: 10.1109/TVCG.2019.2934619

[47] J. Xia, Y. Hou, Y. V. Chen, Z. C. Qian, D. S. Ebert, and W. Chen. Visualizing rank time series of wikipedia top-viewed pages. *IEEE computer graphics and applications*, 37(2):42–53, 2017.

[48] K. Xu, S. Guo, N. Cao, D. Gotz, A. Xu, H. Qu, Z. Yao, and Y. Chen. Ecglens: Interactive visual exploration of large scale ecg data for arrhythmia detection. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI 18. Association for Computing Machinery, New York, NY, USA, 2018. doi: 10.1145/3173574.3174237

[49] Q. Zhang, Y. Nian Wu, and S.-C. Zhu. Interpretable convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[50] J. X. Zheng, S. Pawar, and D. F. Goodman. Graph drawing by stochastic gradient descent. *IEEE Transactions on Visualization and Computer Graphics*, 25(9):2738–2748, 2019. doi: 10.1109/TVCG.2018.2859997

[51] D. Zügner, A. Akbarnejad, and S. Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD 18, p. 28472856. Association for Computing Machinery, New York, NY, USA, 2018. doi: 10.1145/3219819.3220078