SHORT-PAPER

Locator: Local Stability for Rankings

FELIX S CAMPBELL, Ben-Gurion University of the Negev, Beer Sheba, Southern District, Israel

YUVAL MOSKOVITCH, Ben-Gurion University of the Negev, Beer Sheba, Southern District, Israel

Published: 22 June 2025

SIGMOD/PODS '25: International Conference on Management of Data
June 22 - 27, 2025
Berlin, Germany

Conference Sponsors:
SIGMOD

https://doi.org/10.1145/3722212.3725113
ISBN: 9798400715648

Locator: Local Stability for Rankings

Felix S. Campbell
Ben-Gurion University of the Negev
Beersheba, Israel
felixsal@post.bgu.ac.il

Yuval Moskovitch
Ben-Gurion University of the Negev
Beersheba, Israel
yuvalmos@bgu.ac.il

Abstract

Rankings play a crucial role in decision-making. However, if minor changes to items significantly alter their rankings, the quality of the decisions being made can be compromised. The stability of ranking is a measure used to assess how modifications to the ranking algorithm or data affect results. We introduce a novel measure we refer to as local stability, indicating the effect of minor changes to the values of an item in the ranking on its rank. Our proposed definition takes into account the presence of multiple items with similar qualities in the ranking, called dense regions, permitting minor modifications to swap the positions of items within the region.

We present Locator, a system utilizing local stability to provide ranking stakeholders–producers, participants, and consumers–with insights into the ranking. Ranking producers can use Locator to adjust their ranking functions, enhancing item stability. They can then share the local stability information with ranking stakeholders to provide a meaningful understanding of how uncertainty in an item's attributes affects its rank without exposing the ranking methodology.

CCS Concepts

• **Information systems → Data management systems**; • **Social and professional topics → Socio-technical systems**.

Keywords

ranking stability, ranking explanation, sensitivity analysis

ACM Reference Format:
Felix S. Campbell and Yuval Moskovitch. 2025. Locator: Local Stability for Rankings. In *Companion of the 2025 International Conference on Management of Data (SIGMOD-Companion '25), June 22–27, 2025, Berlin, Germany*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3722212.3725113

1 Introduction

Ranking items or individuals by their quantitative attributes often plays a key role in a wide range of domain applications, *e.g.,* in academia [7] or e-commerce [4]. Ideally, a higher ranking should reflect a meaningful improvement in quality. However, if minor adjustments to the data result in significant shifts in an item's position, this fundamental assumption of ranking is undermined.

Example 1.1. When choosing a university to enroll in, students often consult a ranking of potential universities compiled by third parties (*e.g.,* CSRankings[1]) in order to pick a program that can best help them achieve their goals. As a running example, we consider a ranking of fictitious universities based on a simplified version of CSRankings for the sake of exposition. Table 1 shows the top-10 universities in descending order of their adjusted average publication counts, computed by

$$\varphi(t) = \sqrt[17]{(t.\texttt{AI Pubs.}+1)^5(t.\texttt{Systems Pubs.}+1)^{12}}$$

Intuitively, this score is the geometric mean of adjusted publication counts for the 5 and 12 subfields of AI and systems respectively. Lakefront University is ranked 1st according to the ranking function. However, with just four fewer publications in systems, it would be 3rd. This raises the question of how different the three top-ranked universities really are or how *stable* a given item in the ranking is.

Table 1: Top-10 computer science departments ranked according to adjusted average publication count. Each group of rows highlighted by a single color represents a dense region in the ranking in which the departments are similar.

	University	AI Pubs.	Systems Pubs.	Score ↓
t_1	Lakefront University	44	36	39.2
t_2	Dempster University	42	35	37.9
t_3	Western Polytechnic	43	33	36.7
t_4	Prairie University	23	25	25.4
t_5	Ogden University	22	24	24.4
t_6	Kedzie Institute	20	24	23.8
t_7	University of Blue Island	21	22	22.7
t_8	Plainfield College	7	13	11.9
t_9	Irving University	8	11	11.0
t_{10}	Kimball College	6	10	9.6

The notion of *ranking stability* has been studied in previous works [1, 2]. In particular, [1] defined the stability of a ranking as a measure of the robustness of a ranking to changes in its methodology. This definition gives a *global* ranking stability score and allows for verifying the stability of the methodology. However, it treats all changes in the ranking equally—for instance, a transposition of a single pair of items is as significant as a complete reversal of the ranking. Rankings often include items with similar qualities, such as universities of comparable standing. This can create *dense regions* within the ranking, where small changes can reasonably lead to position swaps among the items.

Example 1.2. The universities in Table 1 are divided into three dense regions. Each dense region corresponds to a similar group of universities, closely related in their number of AI and systems publications. As shown in Example 1.1, with small modifications

[1] https://csrankings.org

SIGMOD-Companion '25, Berlin, Germany

ACM ISBN 979-8-4007-1564-8/2025/06
https://doi.org/10.1145/3722212.3725113

to the number of publications (4 in total), the 1st ranked university becomes 3rd. In contrast, the change in the number of publications that would shift Lakefront University to be ranked 4th is 17.

A global ranking stability measure may overlook these dense regions. Therefore, we propose the concept of *local stability,* which considers stability as a property of individual items within the ranking rather than the ranking as a whole. This approach evaluates changes to individual item attributes and assesses the magnitude of changes required to shift an item's position in the ranking. By focusing on data changes rather than methodology, we can treat the ranking process as a black-box, allowing our definition and system to accommodate any ranking methodology, including complex learning-to-rank models.

Our definition of local stability measures the stability of an item in the ranking while taking into account the region of similar items. Intuitively, we say that a tuple is locally stable if minor changes to its values do not alter its position beyond its region, though they may lead to swaps within the region. Understanding the local stability of items can help build trust among ranking stakeholders–producers, participants, and consumers–by providing insights into items of similar quality and assuring that small changes do not result in significant position shifts, thereby reinforcing the core assumption of ranking.

In this paper, we present Locator (LOCAl sTability fOr Rankings), a system enabling ranking producers to explore and tune their ranking methodology to improve the local stability of the items in the ranking through a dedicated web interface. The local stability information is computed by Locator by adapting counterfactual explanation methods. After tuning, Locator empowers ranking producers to share their ranking and the accompanying local-stability information with all stakeholders in the ranking, giving participants and consumers of the ranking a meaningful understanding of the effect of uncertainty in an item's attributes on its rank without exposing the producer's (potentially private) methodology.

We will demonstrate Locator using real-life datasets and invite the audience to play the role of ranking producer, in which they will interact with Locator to explore and improve the local stability of items in the ranked data and then demonstrate the resulting ranking with the local stability information.

Related work. **Ranking stability** has been studied previously either in a global context [1] or with regard to local changes to tuples' values for *specific* sets of ranking functions [2]. However, these works are not model-agnostic with respect to the ranking function, nor are dense regions explicitly handled. **Ranking explanations** (*e.g.,* [3, 8]) are an emerging body of work focused on providing insights into various aspects of the ranking process, *e.g.,* the importance of individual features [8]. Locator may be viewed as providing ranking explanations, though for an aspect not covered by prior work. In our solution, we adapt **counterfactual explanations** (*e.g.,* [6, 12]) which provide concrete examples of how to achieve *recourse* for an undesirable classification, a more general setting than ranking.

2 Technical Background

We next present the notion of local stability. We consider a *ranking* to be a permutation of a set of tuples as in [8]. A *ranking function* is a function f mapping a database D to a permutation of its tuples. We use $f(D)$ to denote the ranking resulting when applying f on a dataset D, and $f(D)[t]$ for the position of t in the ranking. When working with any database D, we consider only its numerical attributes.

Refinements. We aim to understand how changes in the attributes of tuples affect their positions in the ranking. To do this, we first define the terms *refinement* and *refinement set* for a tuple, which describe possible modifications to the tuple and their impact on its values.

Definition 2.1. Given a database D with numeric attributes $a_1, \ldots, a_n$, a *refinement* is a vector $\varepsilon = \langle \varepsilon_1, \ldots, \varepsilon_n \rangle \in \mathbb{R}^n_{\geq 0}$. Given a refinement ε, the *refinement set* of a tuple t is

$$\varepsilon(t) = \{t' \mid \forall i \in [n] \quad |t.a_i - t'.a_i| = \varepsilon_i\}$$

Intuitively, the refinement set of a tuple $\varepsilon(t)$ is the set of tuples that can be obtained by applying modifications to the values of t by the given refinement ε.

Example 2.2. Let us continue from Example 1.1. Considering a refinement $\varepsilon = \langle 10, 5 \rangle$, we have that $\varepsilon(t_1)$ contains, e.g., of the tuples $(54, 41)$, obtained by adding the maximal values in ε to each one of the attributes of t. It also contains, $(54, 31)$, $(34, 41)$, and $(34, 31)$.

Recall that we want to allow small shifts in the ranking position of a tuple when multiple tuples have a similar quality, namely, in dense regions. To account for these dense regions in the ranking, we introduce a parameter k, which represents the size of the area around a tuple (i.e., tuples that are ranked at most k position above or below) with approximately equal quality. We first define the notion of *position change* and then use it to define *k-unstable* and minimal unstable refinements.

Definition 2.3 (Position change). For a database D and a tuple t, let $D_{t \to t'}$ be the database obtained by replacing t with t'. We then define $\Delta(t, \varepsilon)$ as the maximum change of positions attained by any refined tuple $t' \in \varepsilon(t)$ in $f(D_{t \to t'})$

$$\Delta(t, \varepsilon) = \max_{t' \in \varepsilon(t)} |f(D)[t] - f(D_{t \to t'})[t']|$$

Example 2.4. Consider the refinement $\varepsilon = \langle 10, 5 \rangle$ from Example 2.2 and the refinement set $\varepsilon(t_1)$. Plugging each refined tuple $t' \in \varepsilon(t)$ into the score function, we find that the maximum change in positions is attained by the refined tuple $t' = (34, 31)$, which has a score of 32.9. Specifically, we have $|f(D)[t] - f(D_{t \to t'})[t']| = 2$ since t' ranks 3rd in $f(D_{t \to t'})$, so $\Delta(t, \varepsilon) = 2$.

Given a database D, a ranking function f, a refinement ε, a maximum change in positions k, and a tuple $t \in D$, we say a refinement set $\varepsilon(t)$ is k-stable for $f(D)$ if and only if $\Delta(t, \varepsilon) \leq k$, i.e., the difference in positions between t in the original ranking and t' in the ranking obtained by replacing t by t' is at most k for every tuple $t' \in \varepsilon(t)$. If this condition does not hold for $\varepsilon(t)$, we say that $\varepsilon(t)$ is *k-unstable* for $f(D)$.

Next, we define the set of minimal k-unstable refinements for a tuple t. We say a refinement ε is *contained* in a refinement ε' and denote it by $\varepsilon \preceq \varepsilon'$ if for every $i \in [n]$, $\varepsilon_i \leq \varepsilon'_i$. For example, the refinement $\varepsilon' = \langle 10, 6\rangle$ contains the refinement $\varepsilon = \langle 10, 5\rangle$. Given tuple $t \in D$, a ranking function f and a value k, we say that $\varepsilon(t)$ is a *minimal k-unstable refinement* if there is no distinct refinement ε' such that $\varepsilon'(t)$ is *k-unstable* with respect to $f(D)$ and $\varepsilon' \preceq \varepsilon$. Note that there might be many different minimal k-unstable refinements for a tuple $t \in D$ and $f(D)$. For instance, considering t_1 in our example, $\varepsilon_1 = \langle 1, 3\rangle$ and $\varepsilon_2 = \langle 0, 4\rangle$ are both minimal 2-unstable refinements. We use $\mathcal{M}_{f(D),t}$ to denote the set of minimal k-unstable refinements for $t \in D$ and $f(D)$.

The set $\mathcal{M}_{f(D),t}$ of minimal k-unstable refinements for $t \in D$ and $f(D)$ gives us insight into how difficult it is to change the position of t in $f(D)$. Intuitively, high values in $\varepsilon \in \mathcal{M}_{f(D),t}$ refer to larger modifications required for a significant position change (more than k positions), thus indicating higher local stability.

Local stability. We define the local stability measure as the probability of applying a refinement to t that is not greater than the minimal refinements required to shift t more than k positions in the ranking. Local stability is defined with respect to a set of reasonable changes $\mathcal{R}$, which bound the values of possible refinements. Namely, $\mathcal{R}$ is a vector $\langle \varepsilon_1, \ldots, \varepsilon_n\rangle \in \mathbb{R}^n_{\geq 0}$ and we consider only ε such that $\varepsilon \preceq \mathcal{R}$.

Example 2.5. In the context of Example 1.1, the set of reasonable changes may depend on the amount of activity in the research area. For example, if we assume AI is a more active area than systems at these universities, we may assume that there is a larger variance in the number of AI papers than systems papers. Therefore, we may choose, *e.g.,* the set of reasonable changes $\mathcal{R} = \langle 10, 5\rangle$, encoding the assumption that each university might have ±10 AI publications, and ±5 systems publications.

Given $\mathcal{R}$ and $\mathcal{M}_{f(D),t}$ we define the set $\mathcal{E}_{f(D),t}$ as follows.

$$\mathcal{E}_{f(D),t} = \{\varepsilon \mid \exists \varepsilon' \in \mathcal{M}_{f(D),t} \text{ such that } \varepsilon' \preceq \varepsilon \preceq \mathcal{R}\}$$

The set $\mathcal{E}_{f(D),t}$ consists of all refinements contained in the set of reasonable changes $\mathcal{R}$ which contain a minimal k-unstable refinement for $f(D)$. We are now ready to formally define local stability.

Definition 2.6. The **local stability** of a tuple t with respect to a database D, a ranking function f, a set of reasonable changes $\mathcal{R}$, and a region-defining parameter k is defined as

$$\text{Stability}(t, k \mid f, D, \mathcal{R}) = 1 - \mathbb{P}_{\varepsilon \sim \text{Unif}(\mathcal{R})}(\varepsilon \in \mathcal{E}_{f(D),t})$$

We can interpret local stability as the probability of *not* making a refinement that is sufficient to move the tuple t at least k positions. Intuitively, a refinement is *sufficient* to move the tuple t at least k positions if it contains a refinement that k-unstable for t and $f(D)$.

Computing local stability. In order to compute the local stability of a tuple t, we need the set of minimal k-unstable refinements $\mathcal{M}_{f(D),t}$. As our stability definition focuses on data perturbation, we assume nothing about the ranking function and support general ranking functions f. For this reason, we consider sampling-based approaches to estimate the local stability. Such approaches are used in model-agnostic counterfactual explanation methods, such as DiCE [6]. We adapt its random sampling method to find refinements of t which differ by more than k positions, and post-process the set to estimate the minimal unstable refinements. Applying Monte-Carlo estimation methods [11], we estimate the local stability as $1 - (|\widehat{\mathcal{E}}_{f(D),t}|/n)$, where $\widehat{\mathcal{E}}_{f(D),t}$ is our estimate of $\mathcal{E}_{f(D),t}$, and n is the sample budget.

3 System Overview

Our framework is implemented in Locator, a web application that enables the exploration and release of rankings with local stability information. Ranking producers can upload their data, define a ranking function, and interactively explore the local stability of tuples based on various values of k and different ranges of reasonable changes. Once the ranking producer is satisfied with the resulting ranking function, the ranking can be released with the local stability information. The front-end of Locator, shown in Figures 1 and 2, is implemented with React[2], and the back-end is implemented in a Python application wrapping our heuristic estimator in a REST API.

Local stability exploration. Ranking producers can interact with Locator to fine-tune their ranking function and assess the local stability of various items in their ranking using a dedicated interface shown in Figure 1. Users upload their datasets and specify a ranking function, which can be either manually designed or created using a learning-to-rank model. For demonstration purposes, we have implemented a learning-to-rank pipeline interface, allowing users to easily generate a ranking model by selecting a subset of attributes used to rank. The pipeline trains a LightGBM [5] learning-to-rank model on the user's selected attributes. Once the dataset and ranking function are set, the user defines the set of reasonable changes. This could be done either by specifying constant values or choosing the attribute's maximum reasonable change to be proportional to another attribute. Finally, users select a tuple and the region-defining parameter k by dragging the vertical slider to the left of the table, highlighting the appropriate region around the selected tuple as it changes. After all the parameters are set, users ask Locator for local stability information for the selected tuple.

Local stability overview. After setting all the parameters, Locator computes the local stability for the selected tuple t and k value. The system presents the user with an overview of the local stability information of the tuple as depicted in Figure 2. The local stability value is displayed using an intuitive color-coding, where a ● red light indicates a low stability value of at most 0.2, a ● yellow light for a value between 0.2 and 0.5, and a ● green light for a value greater than 0.5.

In addition to the stability value, Locator presents the users with possible modifications to the attributes of t that are required to alter its ranking by more than k positions. These changes are displayed to the user in the form of a bar chart, such that the change in each dimension may be easily visualized and compared. The chart also includes the set of reasonable changes highlighted in gray, allowing the user to compare these refinements with the given set of reasonable changes. Intuitively, modifications that are close to the highlighted area are harder to achieve, thus indicating a higher local stability. The user can switch between refinements,

[2] https://react.dev

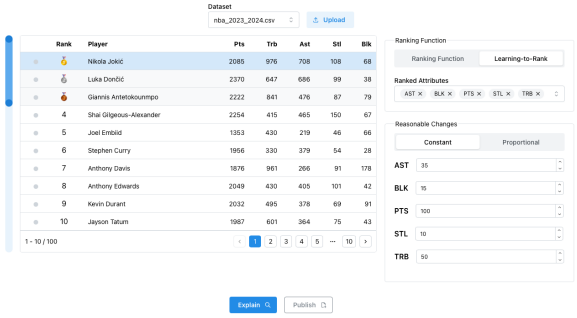

Figure 1: Ranking Producer View

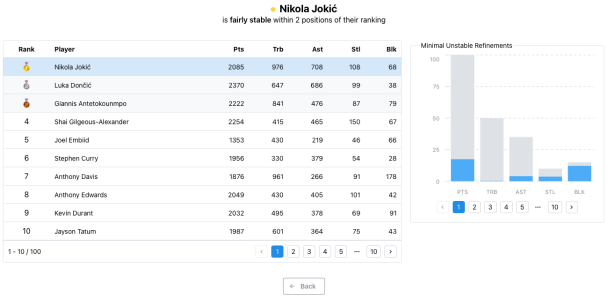

Figure 2: Single Tuple Explanation View

which are ordered by their (ℓ_2 norm) distance from the original tuple. The local stability information can be released alongside the ranking, offering consumers a deeper understanding of it.

4 Demonstration

We will demonstrate Locator using three real-life datasets:*(i)* **NBA**, a dataset of 100 athletes in the NBA, ranked according to [10]. Since the reference ranking provides no statistics, we obtain 5 statistics–points (PTS), total rebounds (TRB), assists (AST), steals (STL), and blocks (BLK)–from [9]. These 5 statistics are the same statistics considered by [3]; *(ii)* **CSRankings** [8], a dataset consisting of 188 universities, including counts of publications and faculty members for four computer science subfields; and *(iii)* **Cars**[3], a dataset of 11,914 real cars with information about their make, model, year, fuel efficiency, horsepower, cylinders, and number of doors.

We let the audience select a dataset and define a ranking algorithm by setting their own function or using the learning-to-rank pipeline. After choosing the dataset and ranking functions, the user sets the reasonable changes, chooses a tuple, and defines k. Participants may then explore the output and interactively refine their ranking function and parameters by returning to the previous step.

Demonstration scenario. Let us consider the NBA dataset and play the role of an analyst designing a ranking of NBA players, aiming to publish a list of this season's top-10 players. We begin with a learning-to-rank running function that incorporates all five attributes of the dataset, generating the ranking shown in Figure 1. We next set the reasonable changes $\mathcal{R}$. We assume a change of at most 5% of the largest difference in each statistic to be realistic. Next, we turn to explore the local stability of different players. Assuming we believe Nikola Jokić, who is ranked 1st according to the ranking function, is this season's MVP, we want to make sure his rank is locally stable within the top-3. We select Nikola Jokić and use the vertical slider to set k to include the two players following him in the ranking and submit our request.

The result, presented in Figure 2, shows a ● yellow light for Jokić, indicating he is only *fairly stable* within the top-3. That is, it requires relatively few changes for Nikola Jokić to be ranked outside of the top-3. Locator also presents examples of such refinements in the chart shown to the right of the table. The changes sufficient to rank Jokić outside of the top-3 are shown in blue, and are relatively small for each attribute compared to the reasonable changes shown in gray. In fact, it requires virtually no change in rebounds (TRB), and relatively small changes all others except blocks (BLK).

Therefore, we return to the first step and tune our learned ranking function to consider only the players' points (PTS) and total rebounds (TRB). The top-10 players ranked by this tuned ranking function share 9 of the top-10 in the previous ranking and have the same players ranked 1st–3rd. However, when requesting local stability information again for Jokić, we find that he is stable not only in the top-3 for this ranking function, but also in 1st.

We can then iteratively explore the local stability for other players, changing k until a ● green light is shown for each, giving us the range in which they are reliably ranked. This information can then be released with the ranking, providing fans with deeper insight into the ranking without revealing the ranking methodology.

[3]https://www.kaggle.com/datasets/CooperUnion/cardataset

Acknowledgments

This research was supported by the Israel Science Foundation grant 2121/22, and the Frankel Center for Computer Science at BGU.

References

[1] Abolfazl Asudeh, H. V. Jagadish, Gerome Miklau, and Julia Stoyanovich. 2018. On Obtaining Stable Rankings. *Proc. VLDB Endow.* 12, 3 (2018), 237–250.
[2] Timothy P. Chartier, Erich Kreutzer, Amy N. Langville, and Kathryn E. Pedings. 2011. Sensitivity and Stability of Ranking Vectors. *SIAM Journal on Scientific Computing* 33, 3 (2011), 1077–1102.
[3] Zixuan Chen, Panagiotis Manolios, and Mirek Riedewald. 2024. Finding Linear Explanations for a Given Ranking. arXiv:2406.11797
[4] Mahsa Derakhshan, Negin Golrezaei, Vahideh Manshadi, and Vahab Mirrokni. 2022. Product Ranking on Online Platforms. *Management Science* 68, 6 (2022).
[5] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *NeurIPS 2017*. 3146–3154.
[6] Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. 2020. Explaining machine learning classifiers through diverse counterfactual explanations. In *FAT* 2020*. 607–617.
[7] Jeremiah P Ostriker, Charlotte V Kuh, and James A Voytuk. 2011. *A data-based assessment of research-doctorate programs in the United States.* Natl. Acad. Press.
[8] Venetia Pliatsika, Joao Fonseca, Kateryna Akhynko, Ivan Shevchenko, and Julia Stoyanovich. 2024. ShaRP: A Novel Feature Importance Framework for Ranking. arXiv:2401.16744
[9] Sports Reference. 2024. *2023-24 NBA Player Stats: Totals.* Retrieved July 29, 2024 from http://basketball-reference.com/leagues/NBA_2024_totals.html
[10] The Ringer. 2024. *The NBA, Ranked.* Retrieved July 29, 2024 from https://nbarankings.theringer.com
[11] Christian P. Robert and George Casella. 2004. *Monte Carlo Statistical Methods.* Springer. doi:10.1007/978-1-4757-4145-2
[12] Peter M. VanNostrand, Huayi Zhang, Dennis M. Hofmann, and Elke A. Rundensteiner. 2023. FACET: Robust Counterfactual Explanation Analytics. *Proc. ACM Manag. Data* 1, 4 (2023), 242:1–242:27.