Supplementary Material for User-Centric Property Graph Repairs

Anonymous Author

ACM Reference Format:
Anonymous Author. 2024. Supplementary Material for User-Centric Property Graph Repairs. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 SURVEY FOR THE USER STUDY

To investigate how the findings from the synthetic experiments translate to a real use case, we tested our approach with real users. For this qualitative experiment, we focus on the `Star Wars`[1] dataset, which is sufficiently small and understandable by real users. We created a small web application using Flask[2] that shows the users the violations and asks them to choose an appropriate repair. We recruited 20 users involving people from our laboratory (including Master, PhD students, and postdocs) and from our connections, aiming for a mix of people with and without a database background. Before the experiment, they were briefly instructed on what their task was. We collected their expertise on databases, graphs, and Star Wars (since it's the domain of the dataset we used) using a 10-level Likert scale. The average knowledge about databases is 5.65, on constraints is 4.35, on graphs is 4.65, and on Star Wars is 5. The questions with their relative answers were:

- How would you rate your knowledge about Databases?

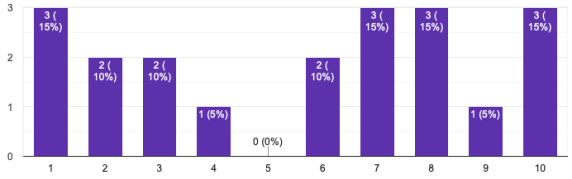


Figure 1: Histogram for the 10-level Likert scale: 10 means expert.

- How would you rate your knowledge about Database constraints?

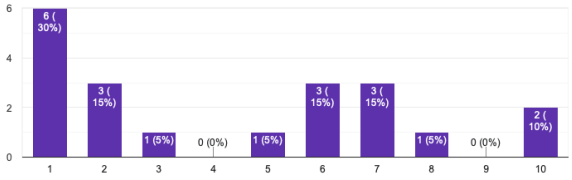


Figure 2: Histogram for the 10-level Likert scale: 10 means expert.

- How would you rate your knowledge about Graphs?

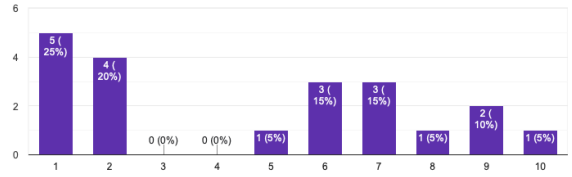


Figure 3: Histogram for the 10-level Likert scale: 10 means expert.

- How would you rate your the Star Wars universe?

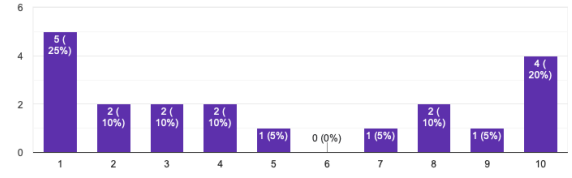


Figure 4: Histogram for the 10-level Likert scale: 10 means expert.

- Which of the following Star Wars movies have you seen?

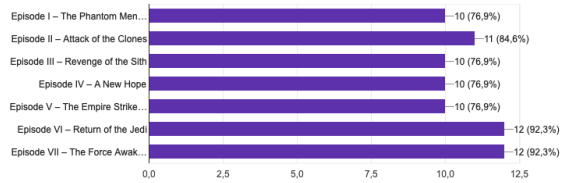


Figure 5: Barplot for the movies count.

2 TABLE SHAPES

Table 1 reports the information about the number of graph pattern shapes that we use in the experiments and affects the construction of the Graph Repair Dependency Graph. For instance, let us consider a star pattern having a central node linked with multiple other nodes with outgoing edges, like *Star 3* shown in Figure **??**. An inconsistency in the central node could potentially be shared with all the nodes of the star, and as a consequence, the effect of its repair can impact all the neighbors. Notice also that, as opposed to the sizes of the datasets, the number of patterns - that is the element on which the constraints are built - grows exponentially

[1]https://github.com/neo4j-graph-examples/star-wars
[2]https://flask.palletsprojects.com/en/3.0.x/

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA
© 2024 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

with respect to the size of the graph, exploding to the order of billions of elements.

Dataset	Chain 2	Chain 3	Star 2	Star 3	Flower	Cycle
StarWars	646	160	2156	10122	0	0
StackOverflow	869	869	15192	∼1.5M	0	0
wwc_2019	43548	43548	55520	∼499k	0	0
Fincen	85158	∼116k	∼12M	∼40B	5113	0
Synthea	36504	51681	∼263k	∼2.5B	439	153
Synthea50	60974	83035	∼2.5M	∼1.8B	1452	415
Synthea75	∼123K	∼172K	∼6.7M	∼3.3B	2489	665
Synthea100	∼144K	∼199K	∼7M	∼4.3B	2639	757
Synthea200	∼294K	∼403K	∼15B	∼36B	4431	1264

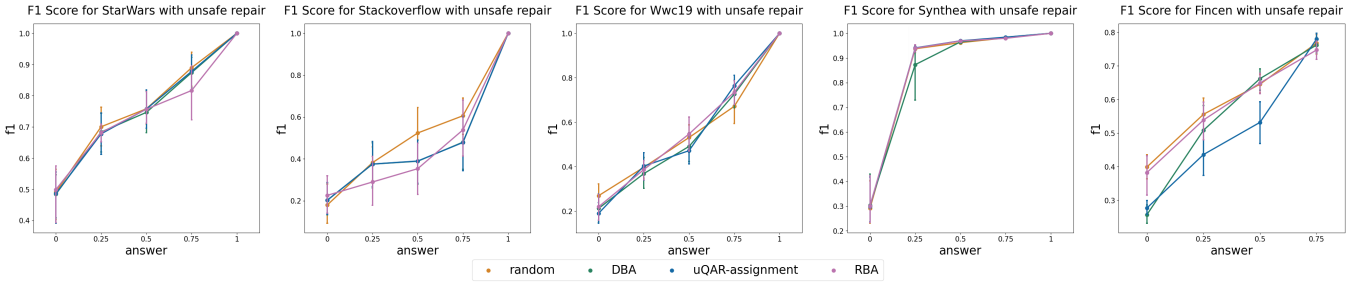
Table 1: Distribution of the number of patterns present in the different datasets

3 EVALUATION OF DIFFERENT ASSIGNMENT POLICIES WITH UNSAFE REPAIRS

To evaluate the effectiveness of the assignment algorithm, we compare it with two alternative policies: Density-Based Assignment (DBA) and Relevance-Based Assignment (RBA). The DBA policy prioritizes violations that are more interconnected in the graph, assigning those with the highest degree first, ensuring that violations affecting many nodes are addressed early. The RBA policy assigns violations based on their importance within the GRDG, with those having the highest PageRank score prioritized. The PageRank reflects the relevance of a node, so addressing more influential violations first may increase the propagation of the repair early on. As shown in in Figure 6 in our experiment with the unsafe configuration we didn't find any significant difference in term of quality of the results.

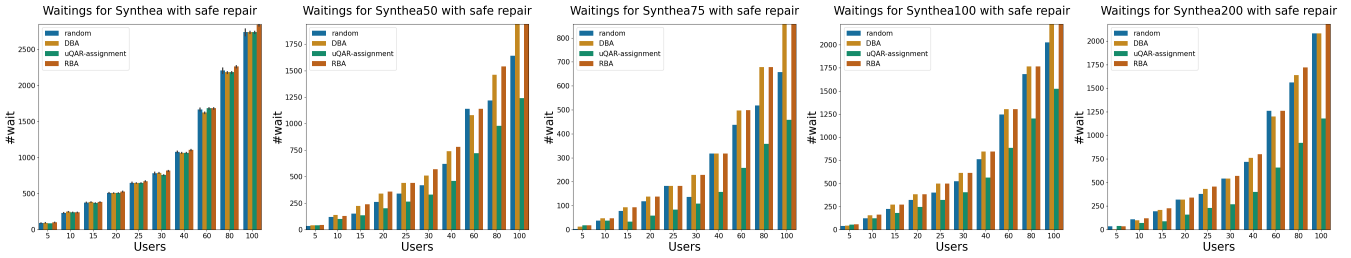
4 NUMBER OF WAITINGS WITH UNSAFE REPAIRS

To assess the scalability of our approach, we tested our repair framework in its safe version (because we wanted to ensure the termination property) on the Synthea dataset generated with an increasing number of patients (15, 50, 75, 100, and 200). Additionally, we considered different numbers of (oracle) users, from 5 to 100. Figure 7 shows that once a plateau is reached, adding more users does not reduce the iterations required, but it increases the number of times the users need to wait. We also compared our algorithm with assigning the violations at random and with the policies described in Section 3. As shown in Figure 7, when the size of the dataset increases, as well as the number of users (when the plateau is not reached), our assignment algorithm takes fewer iterations and produces fewer waits.



(a) (b) (c) (d) (e)

Figure 6: Average F1-measure of our approach using different assignment policies. The error bars represent the standard deviations.



(a) (b) (c) (d) (e)

Figure 7: Number of waitings for each dataset vs users number and the function used to rank the nodes in the GDRG. The y axes have different scale.