Evaluating Query Strategies for Different Feedback Types in Interactive View Recommendation

Xiaozhong Zhang Xiaoyu Ge Panos K. Chrysanthis
Department of Computer Science, University of Pittsburgh
xiz151@pitt.edu, xig34@pitt.edu, panos@cs.pitt.edu

2020 24th International Conference Information Visualisation (IV) | 978-1-7281-9134-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/IV51561.2020.00070

***Abstract*—Existing visualization recommendation approaches introduced an overwhelmingly large number of utility functions (UFs) to rank the visualizations (i.e., views). In order to discover the ideal UF and their tunable parameters that are most suited for a particular analysis context, recent works have proposed Interactive View Recommendation (IVR). IVR identifies the ideal UF by learning from the interactions with the user during the analysis process. We claim that the user is usually unable to provide an accurate real number between 0 and 1, indicating the view interestingness, and in this work, propose three alternative feedback types: binary, Likert scale, and pairwise comparison. We further developed ranking algorithms for the three feedback types. Finally, we propose different example selection strategies, and experimentally evaluate them on the three feedback types. We found that uncertainty-based and hybrid-based strategies usually outperforms the random strategy in terms of recommendation accuracy across all feedback types, while the interestingness-based strategy usually performs worse than the random strategy.**

I. Introduction

The ubiquitously available information sources and the advancements in data storage and acquisition techniques have led to an unprecedented increase in the data volumes available for data analysis tasks. One major challenge in utilizing these abundantly available data is discovering insights from them effectively and efficiently. Examples of an "insight" include the structure, patterns, and causal relationships. To explore these massive and structurally complicated datasets, data analysts often utilize visual data analysis tools, such as Tableau, Qlik, Lyra, Amazon Quicksight, Google Fusion Tables etc. [17]. However, the effectiveness of these tools depends on the user's expertise and experience. Coming up with a visualization that shows interesting trends/patterns is a non-trivial issue.

To address the shortcomings of the current visual analysis tools, several methods for recommending visualizations have recently been proposed (e.g., [21], [10], [16], [9], [4], [14], [12], [22]). These methods automatically generate all possible views of data, and *recommend* the *top-k interesting* views, according to some utility function (UF) (e.g., deviations, data variance, usability) that measures the interestingness of data. Even though each UF might be suitable for specific scenarios, identifying the ones that are most suitable for the current analysis context, is still a challenge for both expert and non-expert users.

A new paradigm in view recommendation, called *Interactive View Recommendation* (IVR) [25] aims to discover the *ideal* UF, i.e., the combination of UFs and their tunable parameters that are most suitable for the current analysis context by learning from the interactions with the user during the analysis process. Current IVR works [25], [23], [24] usually require the user to provide feedback for selected example views in the form of a real number between 0 and 1, indicating the view interestingness and use machine learning models to learn the UF based on the feedback.

In this paper, we claim that the user is usually unable to provide an accurate real number feedback between 0 and 1, due to the infinite number of choices that the user faces in this kind of feedback type. Motivated by the above issue, we propose three alternative feedback types in this work, to which the user is more likely to provide accurate answers. We also design and experimentally evaluate different example selection strategies (i.e., query strategies) for the three feedback types.

Below are the main contributions of this paper:

1) We propose three new feedback types, namely, *binary, Likert-scale*, and *pairwise comparison* to be used in the interactive view recommendation, and design their corresponding ranking algorithms. (§III)
2) We propose different query strategies to effectively leverage each feedback type for interactive view exploration. (§IV)
3) We experimentally identify the effectiveness of our query strategies with different feedback types using real-world datasets. (§V-VI)

II. Background

In this section, we present the necessary background details of our work. Specifically, we discussed view construction in the database context, the traditional view recommendation problem, and the interactive view recommendation problem.

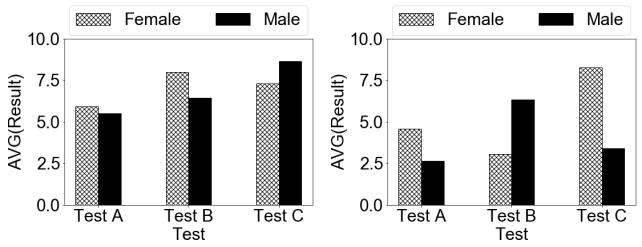
A. Views & Data Visualization

In the context of structural databases, a view (i.e., histogram or bar chart) essentially represents an SQL query with a group-by clause over a database D [21], [4].

Under the typical multi-dimensional data models, data can be modeled as a set of measure attributes (i.e., numerical attributes) $M = \{m_1, m_2, m_3, ...\}$ and a set of dimension attributes (i.e., categorical attributes) $A = \{a_1, a_2, a_3, ...\}$. The measured attributes are a set of attributes that contain measurable values and can be aggregated. The dimension attributes are the set of attributes on which measured attributes are viewed. An SQL query with a group-by clause need to have a set of aggregation functions $F = \{f_1, f_2, f_3, ...\}$. Thus, each view v_i is the visualization of the query result of applying an aggregation function.

Two example views are shown in Figure 1. The two views are both comparing the average test results of female and male patients, and are constructed by applying the *Average* function on the *Result* attribute, and grouping the aggregate values by the *Test* attribute for female and male patients.

2375-0138/20/$31.00 ©2020 IEEE
DOI 10.1109/IV51561.2020.00070



(a) View A (b) View B

Fig. 1: Example Views

It can be seen that, the View Space (VS), i.e., the total number of possible views is:

$$VS = |A| \times |M| \times |F| \quad (1)$$

where $|\cdot|$ is the attribute count. Clearly, View Space can be very large, especially for high-dimensional data.

B. Traditional View Recommendation

In order to recommend the set of k most interesting views from a large number of views, traditional view recommendation approaches [21], [4], [14], [22], [12] have proposed a variety of utility functions (UFs) to rank the views. A UF maps a view to a real number indicating the interestingness of the view.

Definition 1: (*Traditional View Recommendation Problem*) Given a database D, a user-specified query Q, a set of results R produced by Q, a UF $u()$, and the size of the preferred view recommendations k. Find the top-k views v_1, v_2, ..., v_k constructed from R that have the highest interestingness according to $u()$ among all possible views.

From the above definition, one can clearly see that in the traditional view recommendation approaches, the UF $u()$ is defined a priori and cannot adapt to different analysis contexts such as the user, and the analysis task.

C. Interactive View Recommendation

A new paradigm in view recommendation called *Interactive View Recommendation* (IVR) was proposed to support analysis context adaptability [25].

Definition 2: (*The IVR Problem*) Given a database D, a user-specified query Q, a set of results R produced by Q, a set of n possible UFs $U = \{u_1(), u_2(), ..., u_n()\}$, and the size of the preferred view recommendations k. Find the top-k views v_1, v_2, ..., v_k constructed from R that have the highest interestingness according to $u()$ among all possible views, where $u()$ can be any combination of the functions in U, and is most suitable for the current analysis context.

In IVR paradigm, the view recommendation system interacts with the user during the analysis process to automatically learn the UF $u()$ that is most suitable for the current analysis context. For example, current IVR approaches ask the user to label selected example views with a score indicating the interestingness of the views and use different learning methods to learn the UF based on user labels.

The labels for the example views in current IVR approaches are usually in the form of a real number between 0 and 1, with 0 being not interesting and 1 being most interesting. However, we claim that the user is usually unable to provide accurate real number labels as the feedback for the example views, and propose three alternative feedback types (i.e., binary, Likert-scale, and pairwise comparison), which are much easier for the user to label, and more likely to have accurate labels.

Algorithm 1 The *ViewSeeker*

Require: The raw data set D and subsets R specified by user queries
Ensure: The view utility estimator UE
1: Unlabeled view set $U \leftarrow generateViews(D, R)$
2: Labeled view set $L \leftarrow$ obtain initial set of view labels
3: $InfoEst \leftarrow$ initialize view informativeness estimator $InfoEst$ using L
4: $InterestEst \leftarrow$ initialize view utility estimator $InterestEst$ using L
5: **loop**
6: Choose one x from U using $InfoEst$
7: Solicit user's label on x
8: $L \leftarrow L \cup \{x\}$
9: $U \leftarrow U - \{x\}$
10: $InfoEst \leftarrow$ refine $InfoEst$ using L
11: $InterestEst \leftarrow$ refine $InterestEst$ using L
12: $T \leftarrow$ recommend top views using $InterestEst$
13: **if** the user is satisfied with T or the user wants to stop **then**
14: Break
15: **end if**
16: **end loop**
17: Return the most recent $InterestEst$

III. View Feedback Types

In this section, we will first introduce our IVR framework *ViewSeeker* [25] in terms of how it selects example views, learns from user feedback, and generates view recommendations. Then we will introduce the three proposed feedback types and how they can be used in *ViewSeeker*.

A. The ViewSeeker Framework

As shown in Algorithm 1, *ViewSeeker* takes in a raw dataset and user queries, then generates all possible views (line 1) in two steps: *ViewSeeker* first generates the grouped aggregated values for each view as mentioned in Section II-A. *ViewSeeker* then uses the scores from the set of individual utility functions $u_1(), u_2(), ..., u_n()$ on each view as the learning representation for it. Next, *ViewSeeker* acquires a couple of initial user labels to initialize the view informativeness estimator $InfoEst$ and the view interestingness estimator $InterestEst$ (line 2-4). Then, in each user-interaction iteration, *ViewSeeker* uses $InfoEst$ to select example views with the highest informativeness scores, solicits user labels on the example views, and refines $InfoEst$ and $InterestEst$ using the updated labeled set (line 6-11). At the end of the iteration, *ViewSeeker* uses the latest $InterestEst$ to estimate the interestingness of all the views to provide recommendations (line 12). *ViewSeeker* repeats the above steps until the user is satisfied with the recommendation, and then returns the latest $InterestEst$ as the exploration terminates.

B. Binary Feedback

The first feedback type is the *binary*. Given an example view, the binary feedback asks the user to indicate if the view is *interesting* or *not interesting*. For example, for the two views in Figure 1, if the user is interested in the test result difference

between the genders, then she will likely mark View A as not interesting and View B as interesting.

The labeled views are used to train and refine the $InterestEst$ in the form of a binary classifier. For view recommendation, we propose to use the positive class probability of the classifier as the interestingness estimation, because the positive class probability reflects the model's belief of the example's relevance (i.e., interestingness).

C. Likert-Scale Feedback

The second type of feedback is the Likert-scale feedback. We propose to use the scale from 1 to 5, which is a commonly used scale, and is also recommended by [18]. Given an example view, the Likert-scale feedback asks the user to indicate the interestingness of the view on a scale of 1 to 5, with 1 being *not interesting* and 5 being *most interesting*. For example, for the two views in Figure 1, the user could give a score of 2 to View A, because there is a small difference between the test results of the two genders; and a score of 4 to View B, because the difference between the genders is large but not extreme, which prevents View B from receiving a score of 5.

Similar to the binary feedback, the user's labels are used to train and refine the $InterestEst$ in the form of a 5-class classifier. For view recommendation, we propose to use the weighted sum of the model's predicted probability and the class weight for each class as the interestingness estimation as shown in Equation 2:

$$I = \sum_{i=1}^{5} (\alpha_i \times p_i) \tag{2}$$

where I is the interestingness estimation, α_i and p_i are the weights and predicted probability for each class.

We use linear spacing to define the class weights, such that the 5 classes have weights 0, 0.25, 0.5, 0.75, and 1.0 for class 1 to 5, respectively. It can be seen that the interestingness estimation will be a real number between 0.0 and 1.0. Similar to binary feedback, our definition of the interestingness estimation changes in the same direction as the model's belief of the example's relevance. For example, if the probability of class 3 decreases and the probability of class 4 increases, it means that the model's belief of the example's relevance increases, and so does our interestingness estimation.

D. Pairwise Comparison Feedback

The third proposed feedback type is the pairwise comparison between a pair of views. Given a pair of example views, the pairwise comparison feedback asks the user to indicate which of the two is more interesting. For example, for the two views in Figure 1, the user could label View B to be more interesting because there is a larger difference between the test results of the two genders.

The partial order information obtained from the comparison feedback can be used to train and refine the $InterestEst$ in the form of a learning-to-rank (L2R) model [11]. An L2R can provide predicted ranking of a list of views, and we use this ranking as the view recommendation for this feedback type.

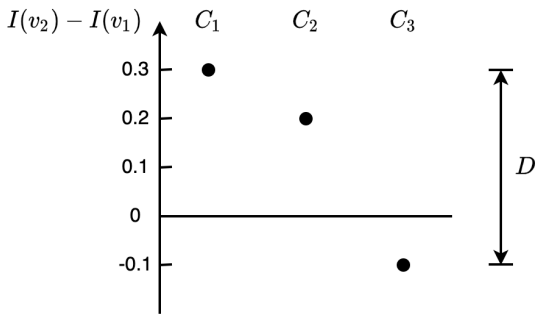


Fig. 2: Committee disagreement calculation for pairwise comparison feedback

IV. Query Strategies

We refer to the example selection strategy in *ViewSeeker* as *query strategy* in accordance with active learning literature [19]. Below we will discuss the details of each query strategy.

Random Strategy This is the vanilla query strategy. When using this strategy, *ViewSeeker* will randomly select example views based on a uniform distribution. In other words, each unlabeled view will have a probability of $1/|U|$ to be selected, where $|U|$ is the size of the unlabeled view set.

Uncertainty-based Strategy This strategy from active learning selects the example views whose labels the current model is most uncertain about, and uses the uncertainty of an example view as its informativeness. We have adopted the query-by-committee (QBC) algorithm [19] as the uncertainty-based strategy. QBC builds a committee of learners and selects the example views on which the committee has the largest disagreement. The disagreement among the committee members is the informativeness measure for the $InfoEst$ under this query strategy.

We define different disagreement metrics for different feedback types. For binary and Likert-scale feedback, we use the standard deviation of the interestingness estimations from the committee members as the disagreement measure. For pairwise comparison feedback, we use the difference in the relative interestingness estimation among the committee members as the disagreement measure. An example of the disagreement calculation is shown in Figure 2. For a pair of view v_1 and v_2, $I(v_2) - I(v_1)$ is the relative interestingness estimation between them. Assume that we have a committee of three members C_1, C_2, and C_3, and the three points represent the estimations from them. The disagreement D is measured as the distance between the highest point and the lowest point. Besides, if the three points are on the same side of the zero line, we set the disagreement to zero, because in such a case, there would be no ranking inversions among the committee members.

Interestingness-based Strategy A different kind of query strategy selects the example views with the highest estimated interestingness. This strategy is also called *active search* [6].

For this query strategy, the $InterestEst$ of *ViewSeeker* also serves as the $InfoEst$, such that the views with the highest estimated interestingness by the $InterestEst$ are regarded as the most informative and are selected. Specifically, for binary and Likert-scale feedback, the informativeness score is the

TABLE I: Ideal Utility Functions
(D: Deviation, V: Diversity, C: Conciseness)

#	Ideal Utility Function
1	$0.6 \times D + 0.2 \times V + 0.2 \times C$
2	$0.2 \times D + 0.6 \times V + 0.2 \times C$
3	$(D + 0) \times (V + 1) \times (C + 1)$
4	$(D + 1) \times (V + 0) \times (C + 1)$

estimated interestingness of the view; and for the pairwise comparison feedback, the informative score is the sum of the estimated interestingness of the views in the pair.

Hybrid-based Strategy This query strategy takes into account both the uncertainty and interestingness of the example. For this query strategy, we first calculate the uncertainty score (i.e., committee disagreement) and the interestingness score (i.e., estimated interestingness) of the example, and multiply the two scores to form the final informativeness score.

V. Experimental Testbed

Dataset: We use two datasets in our experiments. The DIAB dataset contains clinical care data of patients with diabetes. The CENSUS dataset [5] contains microdata from the U.S. labor force survey.

Analysis Task: We created two analysis tasks. For the DIAB dataset, we perform analysis by comparing clinical data between female and male patients, and for the CENSUS dataset, we compare the microdata between female and male labor force. To generate views, we create a data subset for each gender, generate all possible views for each subset, and combine each view with the corresponding view from the opposite gender to form a comparison view.

Ideal Utility Functions: There are three individual utility functions involved in our experiments, each measuring a different aspect of the views. They are *Deviation* [21], *Diversity* [8], and *Conciseness* [8]. From the individual utility functions, we create composite UFs that are most suitable for each analysis context, which are called *Ideal Utility Functions* (IUFs). We have simulated four IUF's in our experiments, as shown in Table I. It can be seen that IUFs 1 and 2 are in linear form as suggested by the current IVR work [25], and IUFs 3 and 4 are in non-linear form as suggested by work [24]. Besides, IUFs 1 and 3 simulate the situation when the user has more interest on *Deviation* in the current analysis context, while IUFs 2 and 4 simulate the situation when the user has more interest on *Diversity* in the current analysis context.

Query Strategies: We experimented with all three query strategies proposed in this work and compared them with a random baseline. We use the following abbreviations for the query strategies: Uncertainty-based (**U-Based**), Interestingness-based (**I-Based**), and Hybrid-based (**UI-Based**).

User Simulation: The simulated user gives her feedback according to the IUF. For binary feedback, we use 70 percentile of the interestingness of all views calculated under the current IUF as the decision boundary for the user. We assume that 70 percentile as the decision boundary can achieve a balance between the quantity and quality of the relevant views. For Likert-scale feedback, we use 5 bins with boundaries at 20, 40, 60, 80 percentiles of the interestingness of all views calculated under the current IUF, and map the view's score into the corresponding bin as the user's choice. For the pairwise comparison feedback, we calculate the scores of the two views based on the current IUF, and use the comparison result of the two scores as the user's feedback.

TABLE II: Testbed Parameters

Total number of records	100K (DIAB), 100K (CENSUS)
Dimension attribute count	6 (DIAB), 4 (CENSUS)
Measure attribute count	8 (DIAB), 7 (CENSUS)
Aggregation function count	5
All possible view count	240 (DIAB), 140 (CENSUS)
Individual utility measure count	3
Query strategy count	4
Query view count per iteration	1
Performance measurement	Top-k accuracy
Recommend view count (k)	5,10,15,20
Runs for each configuration	5

Learning Methods: We use XGBoost [2] with linear boosters for all three feedback types. Specifically, we use XGBoost Classifier for the binary and Likert-scale feedback types and XGBoost Ranker for the pairwise comparison feedback type.

Recommendation Accuracy: Assuming that the top-k views ranked by the IUF is V^* and the top-k views ranked by the learner are V, then the recommendation accuracy (or accuracy for short) is $|V^* \cap V|/k$.

VI. Experimental Results

Binary Feedback The experimental results for the DIAB dataset for the binary feedback are shown in Figures 3–6. We can see all four query strategies have similar accuracy at the early stage of the exploration at 10 labels. U-Based and UI-Based strategies outperform the baseline at 20 labels and more, with an average accuracy improvement of 15.5% for U-Based and 17.2% for UI-Based against the Random strategy. In contrast, I-Based performs consistently worse than the baseline.

Similar observations are made for the CENSUS dataset, whose results are shown in Figures 7–10.

Likert-scale Feedback The experimental results for the DIAB dataset for the Likert-scale feedback are shown in Figures 11–14. These are similar to Binary feedback: all four query strategies have similar accuracy at the early stage of the exploration at 10 labels; U-Based and UI-Based strategies outperform the baseline at 20 labels and more, with an average accuracy improvement of 14.1% for U-Based and 10.4% for UI-Based against the Random strategy; and I-Based consistently performs worse than the baseline.

The corresponding results for the CENSUS dataset are shown in Figures 7–10. We have made similar observations for the CENSUS dataset, with the difference that the advantage of the U-Based strategy and the disadvantage of the I-Based strategy against the baseline only shows after 20 examples.

Pairwise Comparison Feedback The experimental results for the DIAB dataset for the pairwise comparison feedback are shown in Figures 19–22. Similar to other feedback types, U-Based and UI-Based strategies outperform the baseline at 20 labels and more, while the I-Based strategy consistently performs worse than the baseline. However, the advantage of the former two and the disadvantage of the later against the baseline are smaller compared to those of the other two

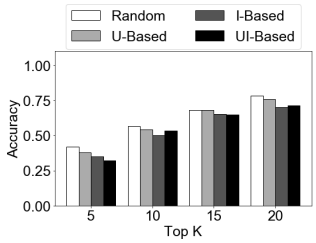


Fig. 3: DIAB, binary feedback, 10 labeled examples.

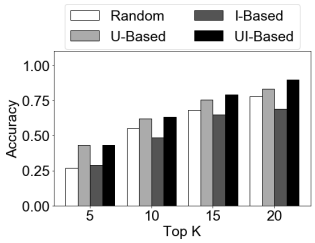


Fig. 4: DIAB, binary feedback, 20 labeled examples.

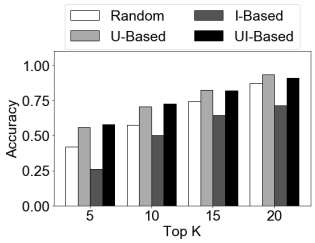


Fig. 5: DIAB, binary feedback, 30 labeled examples.

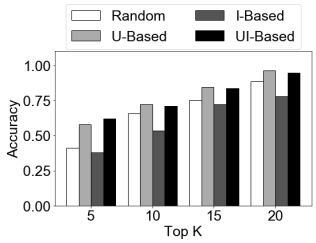


Fig. 6: DIAB, binary feedback, 40 labeled examples.

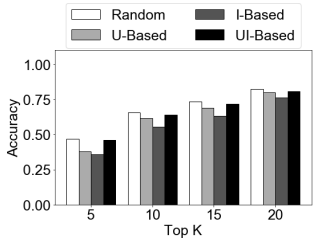


Fig. 7: CENSUS, binary feedback, 10 labeled examples.

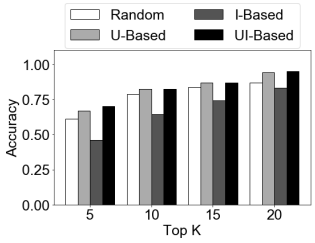


Fig. 8: CENSUS, binary feedback, 20 labeled examples.

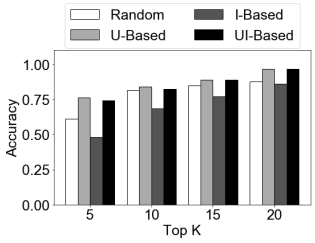


Fig. 9: CENSUS, binary feedback, 30 labeled examples.

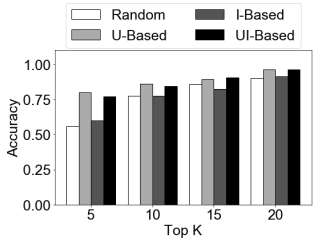


Fig. 10: CENSUS, binary feedback, 40 labeled examples.

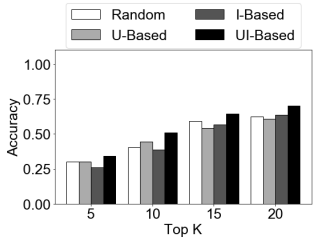


Fig. 11: DIAB, Likert feedback, 10 labeled examples.

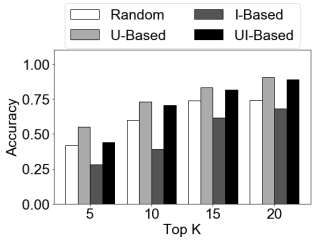


Fig. 12: DIAB, Likert feedback, 20 labeled examples.

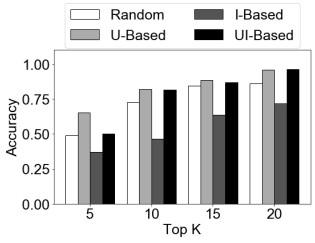


Fig. 13: DIAB, Likert feedback, 30 labeled examples.

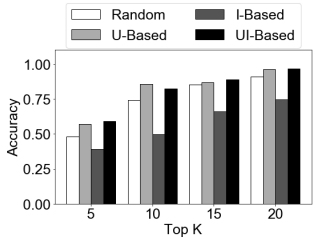


Fig. 14: DIAB, Likert feedback, 40 labeled examples.

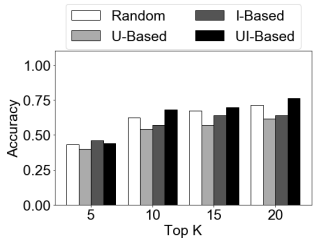


Fig. 15: CENSUS, Likert feedback, 10 labeled examples.

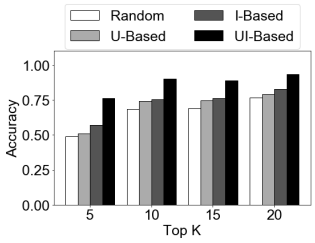


Fig. 16: CENSUS, Likert feedback, 20 labeled examples.

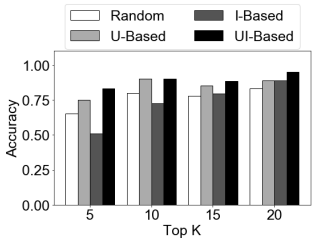


Fig. 17: CENSUS, Likert feedback, 30 labeled examples.

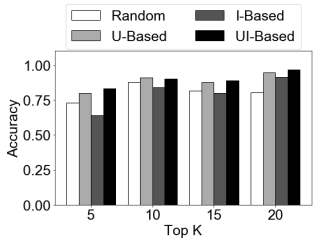


Fig. 18: CENSUS, Likert feedback, 40 labeled examples.

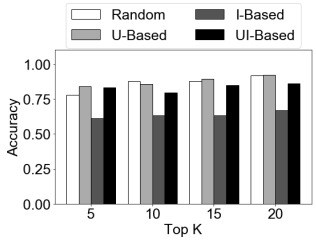


Fig. 19: DIAB, pairwise comparison, 10 labeled examples.

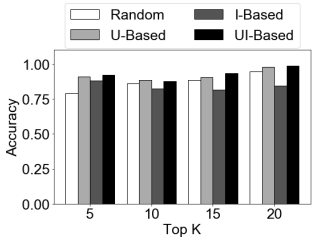


Fig. 20: DIAB, pairwise comparison, 20 labeled examples.

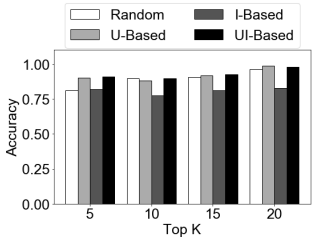


Fig. 21: DIAB, pairwise comparison, 30 labeled examples.

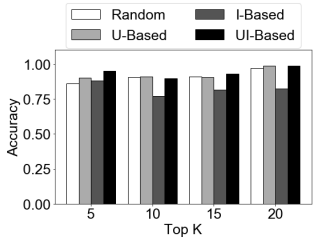


Fig. 22: DIAB, pairwise comparison, 40 labeled examples.

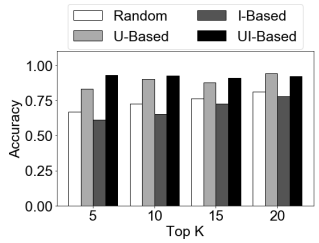


Fig. 23: CENSUS, pairwise comparison, 10 labeled examples.

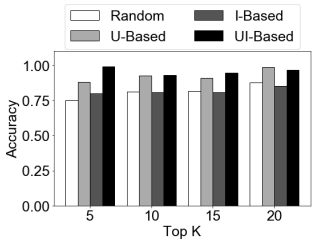


Fig. 24: CENSUS, pairwise comparison, 20 labeled examples.

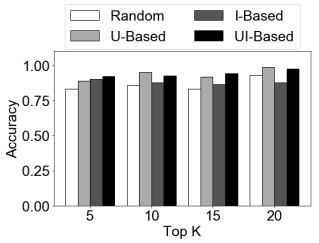


Fig. 25: CENSUS, pairwise comparison, 30 labeled examples.

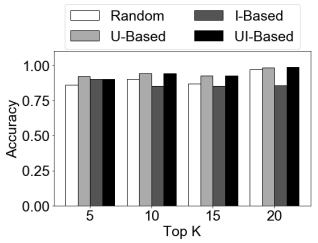


Fig. 26: CENSUS, pairwise comparison, 40 labeled examples.

feedback types. Similar trends are found for the CENSUS dataset, whose results are shown in Figures 23–26.

VII. Related Works

View Recommendation techniques automatically generate all possible views of data, and recommend the top-k interesting views, according to some utility function (UF) (e.g., [21], [4], [14], [12], [22], [16]). The key difference between our work and all prior work is that all previous works use predefined view UFs and do not discover the UF that best matches an individual user's intention and exploration task.

Interactive Visualization Tools have been studied extensively for the past few years [9], [13], [20], [10], [1]. Unlike visualization recommendation tools such as *ViewSeeker* that recommend visualization automatically by searching through the entire views spaces, traditional interactive visualization tools require the user to manually specify the views to be generated. Recently, a few interactive visualization tools have attempted to automate part of the data analysis and visualization process. The work [1] also uses user feedback to steer the view exploration. However, the user feedback is only used to make binary relevance decisions, which is not capable to estimate the ideal UF and get the top-k view rankings.

Data Exploration techniques that aim to efficiently extract knowledge from data [15] are complementary to our work. In particular, *example-driven* data exploration approaches [7], [3] assume minimum prior knowledge of the data and share the same underlying approach as *ViewSeeker*. These works aim to iteratively construct the exploratory query through user interactions as *ViewSeeker* iteratively discovers the UF using user feedback. *ViewSeeker* is well suited to such situations and can enhance example-driven data exploration by creating visualizations that illustrate interesting patterns during the construction of the exploratory queries.

VIII. Conclusion

In this work, we first claim that, in interactive view recommendation (IVR), the user is usually unable to provide accurate real number feedback, and propose three alternative feedback types for an IVR setting: binary, Likert-scale, and pairwise comparison. We further developed ranking algorithms for these three feedback types. Finally, we propose four different example selection strategies, and experimentally evaluate them on the three feedback types.

Our experimental results using two real datasets suggest the use of Uncertainty-based or Hybrid-based query strategies, and discourage the use of Interestingness-based query strategy in IVR, especially for the binary and Likert-scale feedback types.

References

[1] M. Behrisch, F. Korkmaz, L. Shao, and T. Schreck. Feedback-driven interactive exploration of large multidimensional data supported by visual classifier. In *IEEE VAST*, pp. 43–52, 2014.

[2] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *ACM SIGKDD*, pp. 785–794, 2016.

[3] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Explore-by-example: an automatic query steering framework for interactive data exploration. In *ACM SIGMOD*, pp. 517–528, 2014.

[4] H. Ehsan, M. A. Sharaf, P. K. Chrysanthis. Efficient recommendation of aggregate data visualizations. *IEEE TKDE*, 30(2):263–277, 2018.

[5] S. Flood, M. King, R. Rodgers, S. Ruggles, and R. Warren. Integrated public use microdata series, current population survey: Version 7.0 [dataset]. In *Minneapolis, MN: IPUMS*, 2020.

[6] R. Garnett, Y. Krishnamurthy, X. Xiong, J. G. Schneider, and R. P. Mann. Bayesian optimal active search and surveying. In *ICML*, 2012.

[7] X. Ge, Y. Xue, Z. Luo, M. A. Sharaf, and P. K. Chrysanthis. REQUEST: A scalable framework for interactive construction of exploratory queries. In *IEEE Big Data*, pp. 646–655, 2016.

[8] L. Geng and H. J. Hamilton. Interestingness measures for data mining: A survey. *ACM Comput. Surv.*, 38(3):9, 2006.

[9] S. Kandel, R. Parikh, A. Paepcke, J. M. Hellerstein, and J. Heer. Profiler: integrated statistical analysis and visualization for data quality assessment. In *ACM AVI*, pp. 547–554, 2012.

[10] A. Key, B. Howe, D. Perry, and C. R. Aragon. Vizdeck: self-organizing dashboards for visual analytics. In *ACM SIGMOD*, pp. 681–684, 2012.

[11] T. Liu. *Learning to Rank for Information Retrieval*. Springer, 2011.

[12] Y. Luo, X. Qin, N. Tang, and G. Li. Deepeye: Towards automatic data visualization. In *IEEE ICDE*, pp. 101–112, 2018.

[13] J. D. Mackinlay, P. Hanrahan, and C. Stolte. Show me: Automatic presentation for visual analysis. *IEEE TVCG*, 13(6):1137–1144, 2007.

[14] R. Mafrur, M. A. Sharaf, and H. A. Khan. Dive: Diversifying view recommendation for visual data exploration. In *ACM CIKM*, pp. 1123–1132, 2018.

[15] D. Mottin, M. Lissandrini, Y. Velegrakis, and T. Palpanas. New trends on exploratory methods for data analytics. *PVLDB*, 10(12):1977–1980, 2017.

[16] B. Mutlu, E. E. Veas, C. Trattner. Vizrec: Recommending personalized visualizations. *ACM Trans. Interact. Intell. Syst.*, 6(4):31:1–31:39, 2016.

[17] X. Qin, Y. Luo, N. Tang, and G. Li. Making data visualization more efficient and effective: a survey. *VLDB J.*, 29(1):93–117, 2020.

[18] M. A. Revilla, W. E. Saris, and J. A. Krosnick. Choosing the number of categories in agree–disagree scales. *Sociological Methods & Research*, 43(1):73–97, 2014.

[19] B. Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[20] C. Stolte and P. Hanrahan. Polaris: A system for query, analysis and visualization of multi-dimensional relational databases. In *IEEE INFOVIS*, pp. 5–14, 2000.

[21] M. Vartak, S. Rahman, S. Madden, A. G. Parameswaran, and N. Polyzotis. SEEDB: efficient data-driven visualization recommendations to support visual analytics. *PVLDB*, 8(13):2182–2193, 2015.

[22] K. Wongsuphasawat, D. Moritz, A. Anand, J. D. Mackinlay, B. Howe, and J. Heer. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE TVCG*, 22(1):649–658, 2016.

[23] X. Zhang, X. Ge, and P. K. Chrysanthis. Leveraging data-analysis session logs for efficient, personalized, interactive view recommendation. In *IEEE CIC*, pp. 110–119, 2019.

[24] X. Zhang, X. Ge, and P. K. Chrysanthis. Interactive view recommendation with a utility function of a general form. In *HILDA*, 2020.

[25] X. Zhang, X. Ge, P. K. Chrysanthis, and M. A. Sharaf. Viewseeker: An interactive view recommendation tool. In *BigVis Workshop*, 2019.