Navigating Data Repositories: Utilizing Line Charts to Discover Relevant Datasets

Daomin Ji
RMIT University
daomin.ji@student.rmit.edu.au

Hui Luo
University of Wollongong
huil@uow.edu.au

Zhifeng Bao*
RMIT University
zhifeng.bao@rmit.edu.au

Shane Culpepper
The University of Queensland
s.culpepper@uq.edu.au

ABSTRACT

Line charts are fundamental to data analysis and exploration, offering concise visual representations of trends. However, gaining access to the underlying data used to construct these charts is often challenging. In this paper, we describe DDLC (short for Dataset discovery via line charts), an automatic dataset discovery tool that is able to not only identify datasets (from a dataset repository) that are "relevant" to the information depicted from a line chart provided by the users, but also empower users to refine search results based on specific visual elements extracted from the line chart. Moreover, DDLC offers multiple avenues for users to validate search outcomes: 1) Providing explanations on how a similar line chart could be generated from the identified dataset; 2) enabling comparison of line charts generated from different datasets via different ways (e.g., the aggregation vs. non-aggregation operator); 3) facilitating fine-grained examination of the correspondence between the line chart and the identified dataset. By seamlessly combining dataset retrieval with visual refinement and validation mechanisms, DDLC offers a comprehensive solution for the data-driven exploration and analysis.

PVLDB Reference Format:
Daomin Ji, Hui Luo, Zhifeng Bao, Shane Culpepper. Navigating Data Repositories: Utilizing Line Charts to Discover Relevant Datasets. PVLDB, 17(12): 4289 - 4292, 2024.
doi:10.14778/3685800.3685857

1 INTRODUCTION

Dataset discovery [2, 3, 10, 12] is a common activity in data exploration and analysis, which aims to identify and select a set of datasets from a large repository that can meet user needs. It serves as a cornerstone in the data science pipeline, influencing the effectiveness of subsequent analysis tasks. Existing dataset discovery methods fall into two main categories: 1) keywords-based dataset discovery [11], which finds datasets whose contents are the most relevant to specified keywords; 2) unionable or joinable dataset discovery [3, 8, 12], which finds datasets that can be joined or unified with a user-provided dataset.

In this work, we introduce an novel method for users to find relevant datasets through line charts. Line charts, being visual representations of data, are essential tools in various domains for decision-making, forecasting, and communicating statistical insights. Nonetheless, the underlying data that generates the line chart is usually unavailable. Thus, one may ask, given a line chart, whether there exists a tool to help users identify a set of relevant datasets that can generate line charts similar to the given one? For instance, in stock market analysis, it enables analysts to delve into historical data associated with stock trend line charts for further exploration, leading to more accurate forecasting.

To bridge this gap, we present a novel dataset discovery tool, called DDLC (short for Dataset discovery via line charts), enabling users to find relevant datasets for a given line chart. At first, DDLC employs a learned cross-modal relevance function to identify an initial set of relevant datasets and returns them to the users. Upon these datasets, users are allowed to refine them by specifying further information that they think should be relevant to the desirable datasets, such as the title and labels of the line chart. Considering that data exploration purpose varies from one user to another and from one stage to another, we additionally provide the following functions to cater for a range of typical purposes below: (1) *Explanation.* DDLC provides explanations for users why such a dataset presents in the result by illustrating how a similar line chart can be generated by the dataset; (2) *Comparison.* Users are also allowed to compare the line charts generated from different datasets in different ways (e.g., the same line chart could be generated from different aggregation operators upon different datasets), such that users can further pick those that truly meet their information needs; (3) *Locating.* Users can check finer-grained correspondence between a line chart and a relevant dataset by specifying a region in the line chart. Accordingly, DDLC will locate the corresponding data segments of the dataset that matches the region best.

In order to achieve the above functions, DDLC is designed to encompass the following components: 1) *Extractor* aims to extract informative visual elements from the line chart that can guide the dataset discovery search, such as lines, labels, titles and ticks; 2) *Matcher* aims to estimate the relevance score between a candidate dataset and a line chart; 3) *Filter* aims to filter out the datasets that are irrelevant to the visual elements specified by the users; 4) *Explainer* provides the explanation on why such a relevant dataset

Proceedings of the VLDB Endowment, Vol. 17, No. 12 ISSN 2150-8097.
doi:10.14778/3685800.3685857

* corresponding author.

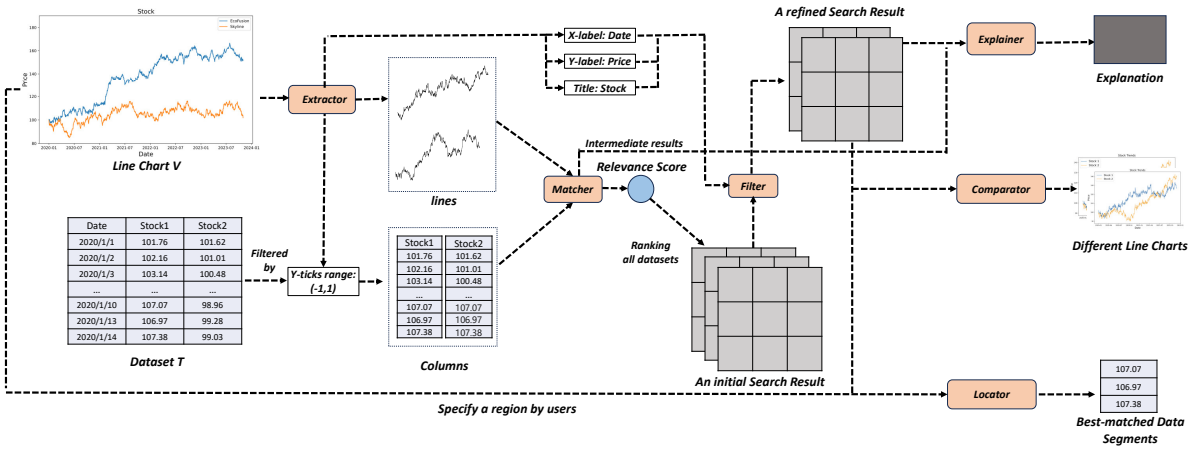


Figure 1: The architecture of DDLC.

is presented in the search result. 5) *Comparator* allows users to compare the line charts generated from different datasets; 5) *Locator* aims to find the data segment that matches the user-specific line chart region best.

Regarding efficiency, DDLC employs a hybrid indexing strategy, combining interval trees and the locality-sensitive hashing, to expedite the search for relevant datasets. This approach significantly reduces the search time compared to a basic linear scan algorithm.

2 TOOL DESIGN

DDLC is mainly composed of the following components, extractor, matcher, filter, explainer, and locator as illustrated in Fig. 1.

Extractor aims to extract key visual elements from a line chart, which is informative about the relevant datasets. Specifically, DDLC considers the following visual elements in a line chart:

- *Lines* are the most essential visual elements in a line chart that demonstrate how the underlying data of the line chart will change within a given period.
- *Ticks* are the markings or values displayed along each axis, indicating the range corresponding to the points of each line.
- *Title* describes the primary topic of the line chart.
- *Labels* delineate the attribute of the data represented along each axis.

Specifically, in DDLC, lines and ticks are the primary visual elements for identifying relevant datasets due to two reasons: 1) they provide key information about the data values and data trend of the datasets; 2) other visual elements may not show up in a line chart or convey misleading information if users arbitrarily specified them during line chart generation. Therefore, DDLC retrieves relevant results in two stages: 1) in the first stage, it leverages lines and ticks to identify an initial set of relevant datasets; 2) in the second stage, it allows users to specify other visual elements that a desirable dataset should be relevant to, thereby refining the search result.

To implement the extractor, we resort to image segmentation methods, which have been widely used in the field of computer vision to extract objects or instances from an input image. Specifically, we train a line chart segmentation model based on Mask R-CNN [5] from scratch. To this end, we automatically label and collect the first training dataset for the line chart segmentation, *LineChartSeg*, with the help of the python visualization library *matplotlib* [6].

Matcher adopts a machine learning model to learn a cross-modal relevance function between a line chart and a candidate dataset, which is achieved in two steps:

- *Representation Learning.* In the first stage, the matcher encodes both the line chart and the dataset and learns their fine-grained representations that can preserve locality semantics. Here, lines, as the most essential visual elements of the line chart, are leveraged to learn the representation of the line chart, and for datasets, DDLC first employs the extracted y-tick ranges to exclude the columns whose range does not have an overlap with the y-tick range. Specifically, for each line of the line chart or each column in the dataset, we first divide it into a sequence of line segments or data segments. Specifically, for a line that can be expressed as a 2-D image with $H \times W$, where H and W represent the height and width of the image, respectively, we divide it into a sequence of small images with $H \times P_1$, where P_1 denotes the width of the line segment. For a column that can be expressed as a data series with length N, we directly divide it into $\frac{N}{P_2}$ data segments with length P_2, where P_2 denotes the length of each data segment. Then we employ Transformer [4, 13] to learn their representations by capturing the relations among the line segments or data segments. Finally, we assemble the representations of all the line segments and data segments to obtain the representation of the line chart and dataset, respectively.
- *Relevance Calculation.* In the second stage, the matcher calculates the relevance between the line chart and the dataset based on their learned representations. To this end, matcher employs a hierarchical attention network to capture the fine-grained alignment between the line chart and the dataset. Specifically, in the low-level attention network, the alignment between each line segment and each data segment is performed, while in the high-level attention network, the alignment between each line and each column is performed. Through the above alignment, the representations of the line chart and the dataset are reconstructed,

which are finally sent into a multi-layer perceptron (MLP) to estimate a relevance score between the line chart and the dataset.

In practice, when employing a line chart to visualize a dataset, people may apply some data aggregation (DA) operations (i.e., *min*, *max*, *mean*, and *sum*) to obtain the statistics within a period. For instance, retailers typically calculate weekly revenue by summing up the daily sales figures throughout the week. However, such data aggregation operations may shift the data distribution of the original data, thus posing a great challenge in accurately calculating the relevance between the line chart and the dataset. DDLC supports handling DA-based line chart by introducing three DA-related layers in the process of the dataset representation learning: 1) The transformation layer to bridge the distribution gap between the aggregated data and the original data. 2) The hierarchical representation learning layer to learn a comprehensive representation for the dataset by jointly considering different aggregation window sizes. 2) The mixture-of-experts layer to infer the most likely data aggregation operator.

Filter aims to judge whether a dataset is relevant to user-specific visual elements, i.e., labels and titles. To this end, the filter also adopts a machine learning model to estimate a relevance score between the visual elements specified by the users and the dataset. To this end, DDLC employs TURL [1], a large language model (LLM) for the tabular data, to obtain the representation of the dataset, and employs a pre-trained LLM, BERT [7], to obtain the representation(s) of the specified visual element(s). If the number of specified visual elements is more than one, DDLC averages the representations to obtain the final representation for them. Finally, a cosine similarity is used to calculate the relevance score between the dataset and the specified visual elements. The similarity score less than a user-specific threshold will be filtered out.

Explainer aims to provide useful explanations for users to help them understand why a dataset presents in the search result. Specifically, explainer tells users how they can leverage a relevant dataset to generate a line chart similar to the given one from two aspects: 1) which column in the dataset is mostly likely to produce a line similar to the one in the line chart; 2) whether a data aggregation operation is involved in the generation of the line chart. To this end, the explainer mainly employs the intermediate result from the matcher to generate the corresponding explanation. Specifically, the alignment information from the hierarchical attention network is leveraged to address the first question, and the distribution of different data aggregation operators inferred by the mixture-of-experts layer is leveraged to address the second question.

Comparator enables users to compare different line charts generated from different datasets in different ways, such that they can further choose the ideal dataset for the subsequent data analysis task. In DDLC, we mainly adopt the common python visualization library *matplotlib* [6] to visualize the selected datasets.

Locator allows users to check fine-grained correspondence between a line chart and a dataset, by identifing the data segment in the dataset that best matches the region specified by the users. To achieve this goal, DDLC also employs the extractor to extract the line segments from the specified region, and then for each line segment, DDLC adopts the matching algorithm proposed in Qetch [9] to locate the data segment that best matches the line segment.

Hybrid Indexing Strategy. Besides the above main components, DDLC also adopts a hybrid indexing strategy to accelerate the search process, which is constructed based on the interval tree and locality-sensitive hashing. The former aims to quickly locate datasets whose values have overlaps with those of the line chart, and the latter aims to reduce the number of candidate datasets in the dataset repository by only considering datasets having the same binary code-based representations with the line chart.

3 DEMONSTRATION OVERVIEW

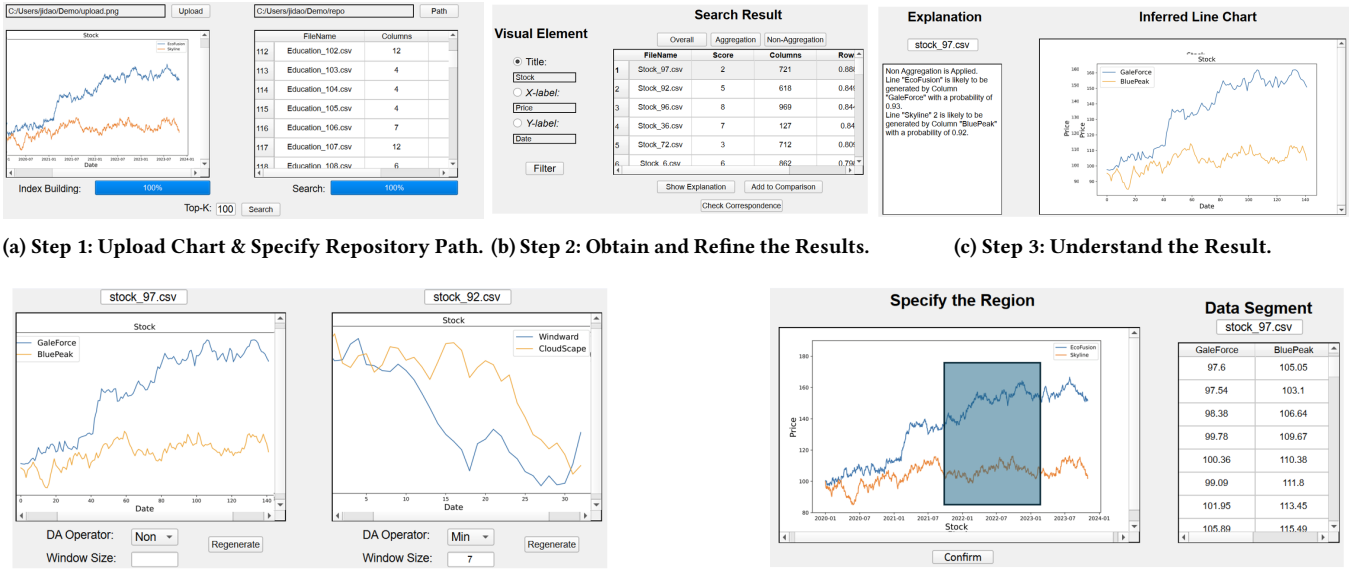
DDLC is implemented based on Python 3.9 and PyQt6. In this demonstration, we illustrate how DDLC assists users in discovering relevant datasets for a given line chart and understanding the relationship between these datasets and the line chart.

Step 1: Upload Chart & Specify Repository Path. At first, users need to upload a line chart and specify the path of dataset repository through the "Upload" and "Path" button, respectively. Then the overview of the line chart and contents of dataset repository will be displayed, as illustrated in Fig. 2a. At the same time, DDLC builds the index for the dataset repository and extracts the visual elements from the line chart in the background.

Step 2: Obtain and Refine the Result. Once the index has been built, DDLC starts the search process on the dataset repository, and the top-k relevant datasets will be displayed as the search result associated with the relevance score estimated by DDLC, where k can be specified by the users (Shown in Fig. 2a). Specifically, DDLC arranges the results in three different lists, *aggregation*, *non-aggregation*, and *overall*, where aggregation and non-aggregation represent whether the line chart is more likely to be generated by the dataset based on a data aggregation operator or not. For example, in Fig. 2b, if users think a desirable table should be relevant to "stock", the title of the line chart, they can tick the corresponding radio button and click the "Filter" button to refine the results. As a result, the datasets whose content does not fall in the stock domain are further excluded from the result.

Step 3: Explain the Result. DDLC also provides useful explanations why each dataset shows up in the search result. Users only need to select the dataset in the search list by clicking one of them, and then click the "Show Explanation" button (shown in Fig. 2b). The corresponding explanation will be displayed from two aspects, as shown in Fig. 2c. The explanation tells users how to leverage the selected dataset to generate a line chart similar to the given one, such as which lines can be derived from specific columns of the dataset. Furthermore, a line chart based on the above inference will be generated for users to compare with the given one. For example, in Fig. 2c, based on the explanation generated by DDLC, the users understand that two stocks , "GaleForce" and "BluePeak" from the Stock_97 dataset have similar trends with "EcoFusion" and "Skyline" from the line chart, and no data aggregation operator is involved in this process. Furthermore, the high similarity between the inferred line chart and the given one also demonstrates the relevance of the selected dataset, Stock_97, to the given line chart.

Step 4: Compare Different Relevant Datasets. DDLC also enables users to compare the line charts generated from different relevant datasets in different ways (e.g., with or without data aggregation), which will help them further choose the ideal dataset for



(a) Step 1: Upload Chart & Specify Repository Path. (b) Step 2: Obtain and Refine the Results. (c) Step 3: Understand the Result.

(d) Step 4: Compare Different Relevant Datasets. (e) Step 5: Check Fine-grained Correspondence.

Figure 2: The illustration of Workflow of DDLC.

their data analysis task. Specifically, users can select the relevant datasets in the result list and then click "Add to Comparison" button (shown in Fig. 2b), then those datasets along with the corresponding line charts will be displayed, as shown in Fig. 2d. Additionally, for each dataset, users can specify the way to generate the line chart by defining the data aggregation operator and aggregation window size in the combo box and input line, respectively. For example, in Fig. 2d, users compare the line charts of two datasets, Stock_97 and Stock_92, and may find that the the former one is better than the latter one, since the line charts of Stock_97 are more similar to the given one. Thus, users may opt for the dataset Stock_97 on the subsequent task.

Step 5: Check Fine-grained Correspondence. DDLC also allows users to check fine-grained correspondence between the dataset and the given line chart, as shown in Fig. 2e. Similarly, users can select a dataset in the result list, and then click the "Check Correspondence" button to activate this process (shown in Fig. 2b). Furthermore, users can specify a region in the line chart that contains several line segments and click the "Confirm" button, then DDLC will conduct a search on the dataset and return the data segment that best matches the specified region. For example, in Fig. 2e, users want to check the correspondence between the dataset Stock_97 and the given line chart. As a result, users may find that for any specified region, there always exists a data segment from the chosen dataset that matches the region well. As a result, they may believe the dataset Stock_97 is the one that they really need for the subsequent data analysis task.

4 ACKNOWLEDGEMENT

This work is supported in part by ARC DP240101211 and DP220101434.

REFERENCES

[1] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2022. Turl: Table understanding through representation learning. *ACM SIGMOD Record* 51, 1 (2022), 33–40.

[2] Yuyang Dong, Kunihiro Takeoka, Chuan Xiao, and Masafumi Oyamada. 2021. Efficient joinable table discovery in data lakes: A high-dimensional similarity-based approach. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 456–467.

[3] Grace Fan, Jin Wang, Yuliang Li, Dan Zhang, and Renée J Miller. 2023. Semantics-Aware Dataset Discovery from Data Lakes with Contextualized Column-Based Representation Learning. *Proceedings of the VLDB Endowment* 16, 7 (2023), 1726–1739.

[4] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. 2022. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence* 45, 1 (2022), 87–110.

[5] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 2961–2969.

[6] J. D. Hunter. 2007. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* 9, 3 (2007), 90–95. https://doi.org/10.1109/MCSE.2007.55

[7] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*. 4171–4186.

[8] Aamod Khatiwada, Grace Fan, Roee Shraga, Zixuan Chen, Wolfgang Gatterbauer, Renée J Miller, and Mirek Riedewald. 2023. SANTOS: Relationship-based Semantic Table Union Search. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–25.

[9] Miro Mannino and Azza Abouzied. 2018. Expressive time series querying with hand-drawn scale-free sketches. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13.

[10] Fatemeh Nargesian, Erkang Zhu, Ken Q Pu, and Renée J Miller. 2018. Table union search on open data. *Proceedings of the VLDB Endowment* 11, 7 (2018), 813–825.

[11] Natasha Noy, Matthew Burgess, and Dan Brickley. 2019. Google Dataset Search: Building a search engine for datasets in an open Web ecosystem. In *28th Web Conference (WebConf 2019)*.

[12] Aécio Santos, Aline Bessa, Fernando Chirigati, Christopher Musco, and Juliana Freire. 2021. Correlation sketches for approximate join-correlation queries. In *Proceedings of the 2021 International Conference on Management of Data*. 1531–1544.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).