An Exploratory Interface for Dataset Repositories Using Cell-Centric Indexing

Drake Johnson
Computer Science, Info. Sys., and Engr.
California University of Pa, California, PA
joh0057@calu.edu

Keith Register
Computer Science
Princeton U., Princeton, NJ
kregister26@gmail.com

Brian D. Davison Jeff Heflin
Computer Science and Engineering
Lehigh University, Bethlehem, PA
{davison, heflin}@cse.lehigh.edu

***Abstract*—Large collections of datasets are being published on the Web at an increasing rate. This poses a problem to researchers and data journalists who must sift through these large quantities of data to find datasets that meet their needs. Our solution to this problem is cell-centric indexing, a novel approach which considers the individual cell of a dataset to be the fundamental unit of search, indexing the corresponding metadata to each individual cell. This facilitates a new style of user interface that allows users to explore the collection via histograms that show the distributions of various terms organized by how they are used in the dataset.**

I. Introduction

Given that the amount of data produced is exponentially growing and interest in datasets is increasing, there is a need for a user-friendly dataset search tool for journalists, researchers, and the public. Existing tools and data repositories tend to focus on indexing metadata, but we argue that tools should be able to consider data values, and their context, as well. We propose the concept of cell-centric indexing [7] to address this problem, where a traditional inverted index is used, but a dataset cell takes the place of a document, and this item has fields for its value, its column heading, its dataset metadata, and the values of other cells in the same row.

In this poster paper, we describe a novel interface that allows users to explore dataset repositories without being familiar with its contents, the structure of the datasets or domain terminology. The tool issues queries to an underlying Elasticsearch[1] server in order to build histograms that summarize the distribution of results for any user query. Similar to faceted-browsing, users can narrow their query by adding more search terms directly from the histograms.

II. Related Work

There is currently a multitude of dataset search tools available for use online. However, there are issues when determining how to rank and present data. These issues are discussed by Chapman et al. in their study of dataset search [3]. Many of these tools use interfaces that resemble textual search engines. A prime example is Google's dataset search tool. The pre-query interface is nearly identical to that of Google's search engine. Once a query is entered the tool returns a list of datasets accompanied by some metadata and domains to show where the dataset was found [2]. Other dataset search tools including Kaggle [6], Data.world [4], and NYU's Auctus Datamart [9] employ similar interfaces. The Auctus Datamart and Google closely resemble one another when presenting results, however the Auctus Datamart provides more information about the dataset when selected. Kaggle and Data.world also have similar interfaces that provide a list of datasets that when selected take you to a page dedicated to the selected dataset. Another interface presented by Maier, Megler, and Tufte [8] is similar to the advanced search feature used by some search engines. Bozzon et al. [1] propose an interface referred to as Liquid Query. When results are provided, the user can interact by changing the query, the content of the results, and the results layout. We present our results based on result frequency in the repository. We discuss this further in Section III-B.

Others have also considered exploratory search interfaces in an effort to help users find data in an area with which they are unfamiliar. Derthick, Kolojejchick, and Roth [5] discussed using a visual query language that dynamically links queries and visualizations to allow for reuse and modification of exploration sessions. Yogev et al. [10] also offer an approach for exploratory search. Their approach combines an expressive query language, faceted search, and ER graph navigation. This allows both knowledgeable users and users unfamiliar with the query language to query entities and their relationships.

III. Interface Overview

An example of a typical use case is demonstrated using the figures provided in this paper. For this specific case, the user wants to find a dataset containing data on Kenya's performance in the 2004 Athens Olympics. Initially, the user is presented with the graphs in Fig. 1a. However, the histograms do not initially show anything regarding the Olympics. By using the "More Items" button at the bottom of the title histogram the user can find the term Olympics and add it directly to their query. After this term is added the screen changes to that shown in Fig. 1b. The user can now look through all 4 histograms and decide which term best helps them get to their desired data. Once again using the "More Items" button,

[1] https://www.elastic.co/elasticsearch/

978-1-7281-6251-5/20/$31.00 ©2020 IEEE

the term Athens can be found in the content histogram. Once this is added the user might direct themselves to the full title histogram shown in Fig. 1c. There the user can find a dataset titled "Kenya at the Olympics Medalists". To gain access to the dataset the user must add the full title to their query. Once a full title is in the query a button appears that performs a Google query of the full title. Since this specific dataset is from WikiTables, the Google query will provide a link to the Wikipedia page containing the table. We now discuss specific interface components in more detail.

A. *Pre-query Histograms*

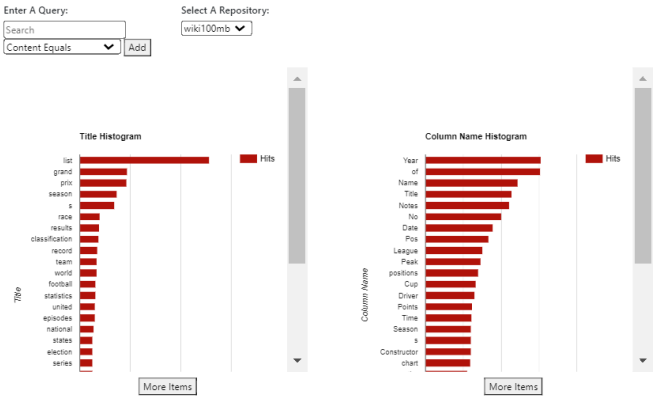
Before any search parameters are set, the user is shown two pre-query histograms that return up to the 50 most frequent title and column name tokens within the current repository (see Fig. 1a). Column name and title histograms provide a good overview and are vital in allowing the user to explore the datasets without prior knowledge of the contents. The pre-query histograms are presented to the user when there are no active queries, such as when the page is initially loaded or when all queries have been deleted. Clicking on a histogram bar will automatically add the corresponding term to the query and generate the standard set of histograms.

B. *Results Histograms*

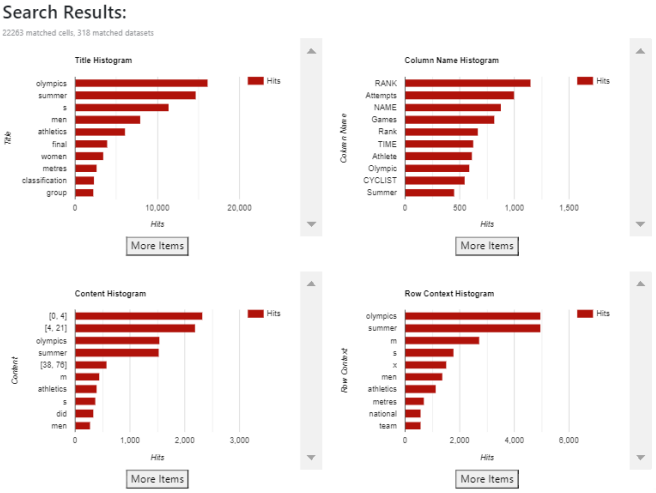
The standard screen displays the user's current query and five histograms. Each histogram is associated with a field, and tokens are sorted in descending frequency of co-occurrence with the query. The length of a bar indicates how many cells match the query. As with the pre-query histograms, clicking on a bar adds the associated term to the query, and generates a new result histogram. Below each histogram is an option to provide more results on the histogram. Initially, each histogram presents the top 10 results, however, the top 25 results are pre-fetched which allows the newly requested results to be automatically added to the histogram.

Due to the connection between matched cells and bar length, there is the possibility that the first bar will be significantly larger than all remaining bars, making them difficult to see or select. To combat this, we compare the results of the two most frequent results. If the first result contains 10 times more results than the second most frequent we change the scale of the histograms to logarithmic, thus making it easier to visualize distinctions in skewed distributions.

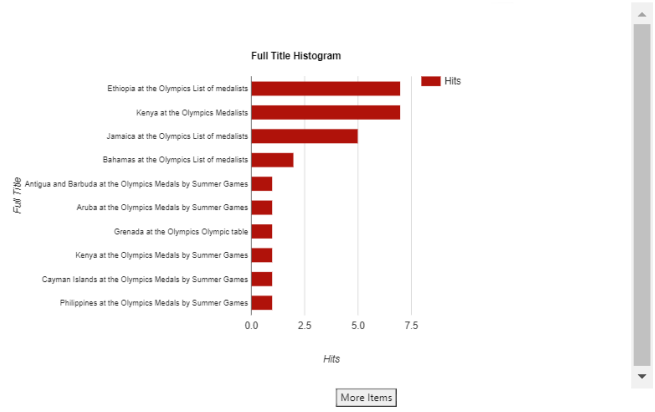
The first four histograms, represented in Fig. 1b, are Title, which contains individual title tokens, Column Name, which contains tokens from the column headings of each dataset, Content, which contains tokenized values from cells, and Row Context, which contains tokens of values in all cells in the same row as the matching cell. In order to identify which datasets best match the query, the final histogram provided is the Full Title histogram (see Fig. 1c). In this histogram, the bars represent the number of cells in a data set that match the user's query. The user can add this bar to the query to get specific information about the distribution of terms in the chosen dataset. Additionally, this enables the option to search for the dataset, which is accomplished using a Google query of the dataset's full title.[2]



(a) Initial pre-query histograms



(b) Search results with query: title:olympics



(c) Results of full title histogram with query: title:olympics, content:athens

Fig. 1: Screenshots from interface.

[2]Many of our dataset collections did not have a URL recorded, which is why we do not just open a new window containing the dataset.

C. Numeric Ranges

To handle numeric values we generate ranges to represent how the numeric content in the datasets is distributed. We generate these ranges by performing an initial percentile Elasticsearch query to generate bounds that exclude outliers by discarding the top and bottom 5 percent of the data. Using these bounds we calculate a mean and standard deviation of the remaining data. Typically, the lowest range and highest range are calculated as follows [min, mean – 1.5 * std dev] and [mean + 1.5 * std dev, max] while there are 3 ranges in between of uniform size with the middle range being [mean – 0.5 * std dev, mean + 0.5 * std dev]. However, when the mean is close to the min or max the data is likely to be distributed with longer tails. To remedy this, we generate more ranges that contain them. For the cases when mean-1.5*std dev is less than the min, there is no need to generate the typical lowest range. Instead, we shrink the ranges above the mean to maintain 5 ranges. In Fig. 1b, the first two bars of the content histogram are generated from our numeric range processing.

IV. System Architecture

The architecture of the system, depicted in Fig. 2, is centered around an Elasticsearch server. Elasticsearch is an open source search engine that supports the creation of inverted indices. One of the functions of Elasticsearch is to index the datasets when they are being loaded into the repository, and because the system uses cell-centric indexing, each cell is the primary focus when the datasets are going through the indexing process. This occurs after the data and metadata from the datasets are initially parsed.

The interactive portion of the architecture is along the bottom of the figure. Queries are issued to Elasticsearch which returns results that allow the creation of the histograms. The front end of this system is generated using a Java Spark server. This is where the user is creating their queries and receiving all the generated histograms. Between the interface and Elasticsearch, there is a query processor, also run on the Java Spark server, that sends the queries to Elasticsearch and generates the histograms once the results are returned. A typical case for the system begins with the user inputting a query into the web interface. This query is then sent to the query processor which then sends the finalized query to Elasticsearch. The results from Elasticsearch are sent back to the query processor that then generates the histograms and sends them back to the web interface that shows the user the results and allows them to interact with the histograms.

Currently, our dataset search tool indexes datasets held in either CSV files or JSON files. Some experiments were done to determine the scalability of the system. For these tests, different repositories were loaded ranging from 26 MB to 3,940 MB. The smallest repository, gathered from DataGov and containing CSV files, took 2.6 minutes to load while the largest repository, gathered from WikiTables and containing JSON files, took 106.3 minutes to load. Beyond just loading times, query times were tested for varying sizes of repositories. One test, done on the largest repository with random fields,

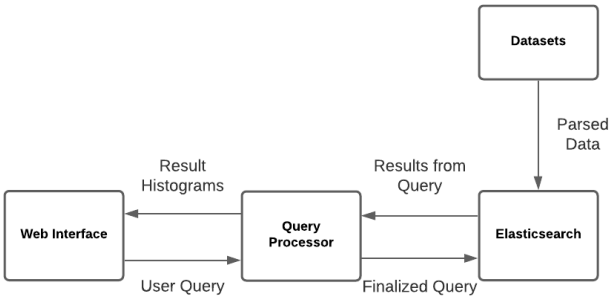


Fig. 2: Basic System Architecture Diagram

resulted in an average query time of 244.2 ms for 1-term queries. This query time went down as more terms were added to the query resulting in an average time of 60.1 ms for 5-term queries. The tests conducted are discussed in greater depth by Qiu et al. [7] and provide further explanation on the results from the tests and the conclusions drawn from them.

V. Conclusion

Histograms allow the user to quickly understand the contents of a repository and to explore simply by adding or removing terms from the query. Therefore, our novel interface is useful for exploring repositories regardless of prior knowledge. Future work includes scaling up the system, adding new features, and a formal user evaluation of the system to determine if our design truly allows novice searchers to be more effective.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. CNS-1757787.

References

[1] Bozzon, A., Brambilla, M., Ceri, S., & Fraternali, P. (2010, April). Liquid query: multi-domain exploratory search on the web. In Proceedings of the 19th international conference on World wide web (pp. 161-170).
[2] Brickley, D., Burgess, M., & Noy, N. (2019, May). Google Dataset Search: Building a search engine for datasets in an open Web ecosystem. In *Proceedings of The Web Conference*, (pp. 1365-1375).
[3] Chapman, A., Simperl, E., Koesten, L., Konstantinidis, G., Ibáñez, L. D., Kacprzak, E., & Groth, P. (2020). Dataset search: a survey. The VLDB Journal, 29(1), 251-272.
[4] Data.world Open Data Datasets. https://data.world/datasets/open-data
[5] Derthick, M., Kolojejchick, J., & Roth, S. F. (1997, October). An interactive visual query environment for exploring data. In Proceedings of the 10th annual ACM symposium on User interface software and technology (pp. 189-198).
[6] Kaggle Datasets. https://www.kaggle.com/datasets.
[7] L. Qiu, H. Jia, B.D. Davison, & J. Heflin. An Architecture for Cell-Centric Indexing of Datasets. International Workshop on Profiling and Searching Data on the Web (PROFILES 2020), ISWC 2020.
[8] Maier, D., Megler, V. M., & Tufte, K. (2014, April). Challenges for dataset search. In International Conference on Database Systems for Advanced Applications (pp. 1-15). Springer, Cham.
[9] NYU Auctus Datamart. https://auctus.vida-nyu.org/
[10] Yogev, S., Roitman, H., Carmel, D., & Zwerdling, N. (2012, April). Towards expressive exploratory search over entity-relationship data. In Proceedings of the 21st International Conference on World Wide Web (pp. 83-92).