DiagramFlyer: A Search Engine for Data-Driven Diagrams

Zhe Chen
University of Michigan
Ann Arbor, MI 48109-2121
chenzhe@umich.edu

Michael Cafarella
University of Michigan
Ann Arbor, MI 48109-2121
michjc@umich.edu

Eytan Adar
University of Michigan
Ann Arbor, MI 48109-2121
eadar@umich.edu

ABSTRACT

A large amount of data is available only through data-driven diagrams such as bar charts and scatterplots. These diagrams are stylized mixtures of graphics and text and are the result of complicated data-centric production pipelines. Unfortunately, neither text nor image search engines exploit these diagram-specific properties, making it difficult for users to find relevant diagrams in a large corpus. In response, we propose DiagramFlyer, a search engine for finding data-driven diagrams on the web. By recovering the semantic roles of diagram components (e.g., axes, labels, etc.), we provide faceted indexing and retrieval for various statistical diagrams. A unique feature of DiagramFlyer is that it is able to "expand" queries to include not only exactly matching diagrams, but also diagrams that are likely to be related in terms of their production pipelines. We demonstrate the resulting search system by indexing over 300k images pulled from over 150k PDF documents.

Categories and Subject Descriptors

H.4.0 [**Information Systems**]: Information systems applications—*General*

Keywords

Web search; diagrams; information extraction

1. INTRODUCTION

Data-driven diagrams (or statistical graphics) are an important method for communicating complex information. Diagrams, a stylized mixture of graphics and text, offer succinct quantitative summaries of data that motivate the overall document's content. Indeed, for many technical documents, the diagrams may be readers' *only* access to the raw data underlying the documents' conclusions. Especially for quantitative disciplines such as finance, public policy, and the sciences, certain diagrams could be even more valuable than the surrounding text.

WWW 2015 Companion, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3473-0/15/05.
http://dx.doi.org/10.1145/2740908.2742831.

Consider a chemist who wants to find all published experiments about a class of compounds called *metal-organic frameworks* [10]. Such materials are promising candidates for a range of applications, including carbon dioxide absorption and hydrogen storage. A common experiment for these materials is to examine the relationship between *temperature* and, say, H_2 *uptake*. Unfortunately there is no reliable shared database of experimental results on this topic: the most authoritative source of data on this topic is the set of relevant diagrams published in scientific papers. The chemist may be uninterested in the scientific paper text: the compelling information is embedded in the diagram alone. It is easy to imagine similar diagram-driven information needs for demographers, public health experts, and other quantitative professionals. It would thus be useful to be able to search for diagrams *per se*, not just relevant documents.

Beyond domain experts, the ability to find similar (or alternative) diagrams may even be highly useful for average citizens. There is an increasing amount of information given to the public through visualizations. In some situations, this presentation can be highly misleading. For example, the original "Obamacare Enrollment" diagram from Fox News displayed a "false baseline" (see Figure 1(a)). Here, we may be interested in identifying corrected versions of the diagram (Figure 1(b)) or alternative representations (Figure 1(c)). Finding these diagrams automatically can correct misconceptions and stimulate balanced discussions.

Of course, we could build such a diagram search engine on top of existing tools. Standard text-based search may be able to retrieve the diagrams' enclosing documents. Image-based search engines, which generally work by examining textual content that surrounds images rather than the visual qualities of the images themselves, may retrieve some diagrams [2, 4]. More recently, some commercial search systems such as Zanran [14] and others [7, 8, 9] can also be used to query data-driven diagrams.

However, searching systems to date have ignored one distinct quality of data-driven diagrams: a diagram is the final product of a multistep generation pipeline. First, the diagram author must choose a dataset to visualize, which is often just a small fraction of the total available data. Second, the author defines a "specification" of what they want displayed either programmatically (perhaps using a set of known rules, such as the grammar of graphics [13]) or through direct manipulation. Finally, a program takes the data and specification and renders a graphical display. These steps are potentially *lossy:* the visualized dataset is likely smaller than the total available database and both

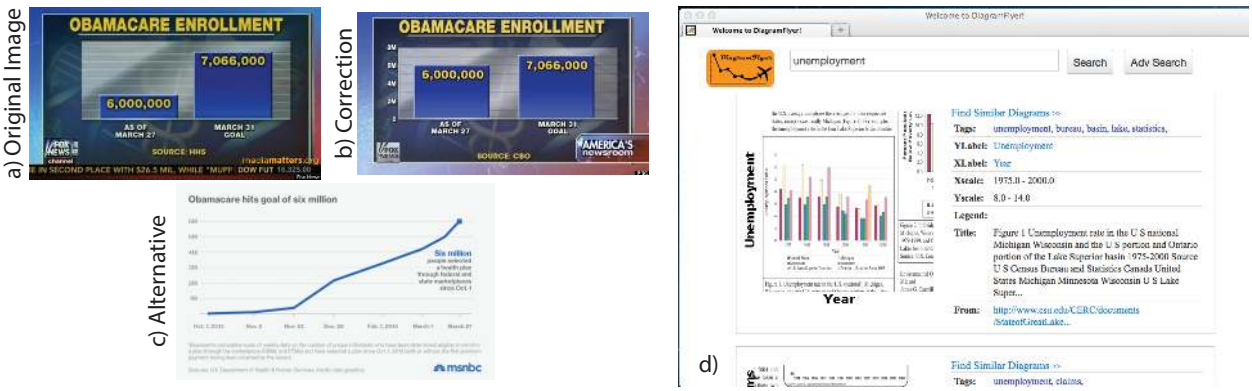

Figure 1: Example diagrams with false baselines (a), a similar "corrected" diagram (b), and an alternative representation (c). On the right is a a screenshot of the DIAGRAMFLYER search system (d).

the author and rendering system may make decisions about what marks to display (*e.g.,* as when only the largest categories are drawn cleanly in a stacked bar chart).

Unfortunately, this latent pipeline information is likely useful to those searching for data-centric diagrams. Consider the materials scientist who wants to find diagrams about metal-organic frameworks under a given certain temperature range; it would ideally work even when the visualization tool does not include a label for each diagram axis. Consider also a demographer who has chosen to generate a small handful of diagrams and likely used a large database that contains more data than any of the distinct diagrams might imply; it could be useful for the searcher to find diagrams that are derived from the same dataset. (Figure 1(d) shows the unemployment statistics in only three states, but this image was likely generated from a much larger dataset.)

In response, we present DIAGRAMFLYER, a web-based search engine for data-driven diagrams. It has two main components. First, for each data-centric diagram discoverable in a large corpus of documents, the DIAGRAMFLYER extractor recovers as much of the underlying diagram production process as possible (*e.g.*, y-axis label, *etc.*). Second, DIAGRAMFLYER gives searchers the ability to search this recovered information, via query tools, ranking methods, and snippet generators. One distinctive feature of DIAGRAMFLYER is its ability to expand queries with a *lexicon generator* to find a broader range of diagrams with similar semantics; for example, if a diagram's x-axis contains a few US States as labels, the lexicon generator could be used to infer the other possible x-axis labels (*e.g.,* other U.S. states) for diagrams that are generated from the same underlying database.

We demonstrate that DIAGRAMFLYER is a working search engine that provides search services for a corpus of 319k diagrams extracted from 153K web-crawled PDFs. First, we have implemented the software architecture and set of algorithms for implementing DIAGRAMFLYER's diagram pipeline extractor, as well as query tools that perform diagram relevance ranking, similar item finding via lexicons, and snippet generation. Second, we demonstrate that the system is able to perform many interesting applications, including searching diagrams via keywords, advanced faceted queries to allow highly targeted searches, and searching for similar diagrams.

In the rest of the paper, we will introduce the system interface and query language (Section 2), give an overview of the system architecture (Section 3), describe how the end-user can interact with DIAGRAMFLYER during the demonstration with an available online video (Section 4), and conclude with a brief summary of the technical problem the system addresses (Section 5).

2. QUERY INTERFACE AND LANGUAGE

Prior to any search engine activity, the web's population of users produces diagrams. Diagram generation is a unique process. First, an underlying database has to be collected ahead of time, then a customized graphical specification has to be designed (either through a specification or direct manipulation), then finally the specification is "compiled" to render the diagram images. The graphical specification describes how to visualize the contained elements that define the structure of the diagram. For example, as mentioned in [13], given a database, a user must specify the data variables for both x- and y- axis, the transformation of the variables, the scale of axes (log or linear), and other characteristics, in order to generate a two-dimensional scatterplot.

DIAGRAMFLYER attempts to extract all the necessary elements of the graph generation specification for the diagrams. We call this specification a *diagram template* or *diagram metadata*. To be more specific, we try to find all textual and visual elements that are necessary to render the final images. In our prototype we focus on 8 key fields that can be used to generate a unique diagram image: `x-label`, `x-scale`, `y-label`, `y-scale`, `title`, `legend`, `caption`, `scale` and `type`. (`Type` identifies what kind of two-dimensional chart it is: bar, line, scatter or other.) For example, Figure 2 shows two sample data-driven diagrams and the diagram metadata that DIAGRAMFLYER found in each. These diagrams, plus the accompanying diagram metadata, form the corpus our search engine will index.

2.1 Query Interface

DIAGRAMFLYER's interface is similar in appearance to traditional web search engines, accepting input into a search box (or boxes in the faceted "advanced" mode) and presenting the results using a top-10-style Search Engine Results Page (SERP). Figure 1(d) shows the current DIAGRAMFLYER prototype SERP, with a query for `unemployment` and one visible hit. A score for each retrieved image is calculated by combining the similarity of each individual fields (*e.g.*, how well do the x-labels match? how much does the x-scale

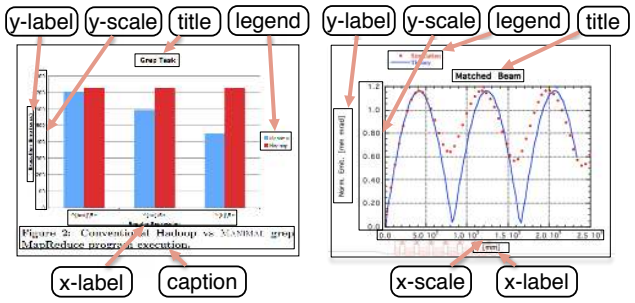

Figure 2: A diagram contains several characteristic regions of text: the `title`, `x-label`, `y-label`, `legend`, and so on.

overlap?, and so on). We will discuss the scoring mechanism in detail in Section 3.

2.2 Query Language

DiagramFlyer's query language supports complex, faceted queries which allows end-users to create highly targeted searches. Thus, DiagramFlyer is able to support querying on features that are part of the descriptive pipeline that generated the diagram. DiagramFlyer's query language is composed of the 8 *field operators* (based on the fields described above) and a *fuzzy expansion* function. Field operators operate against a faceted index of the diagrams (each field is stored separately). For example,

Example 1. If a user wants to get diagrams about population statistics over the year 1990 to 2014, she can formulate the query as follows:

```
x-label: year AND
x-scale: from: 1990 to: 2014 AND
y-label: population
```

DiagramFlyer's query language also supports a variety of diagram search applications including "similar diagram" search. DiagramFlyer supports fuzzy matching. When processing a search query, a unique component of DiagramFlyer is its *query expander*, which is able to expand the query by generating semantically similar terms. The goal of lexicon expansion is to retrieve diagrams using information that may have been removed as a side effect (either intentionally or not) in the diagram production process. For example, given a term "Michigan", the *query expander* could recognize that terms such as "California" and "Wisconsin" are highly relevant terms belonging to the same category. This lexicon was built by analyzing hundreds of millions of web pages, which we will discuss in detail in Section 3.

Example 2. After finding a relevant diagram d_q, a user may want to retrieve all the relevant diagrams that could potentially be generated from the same underlying dataset. The query can be formulated as:

```
x-label:expand(d_q.x-label) AND
y-label:expand(d_q.y-label) AND
title: d_q.title AND
caption: d_q.caption
```

3. SOFTWARE ARCHITECTURE

The DiagramFlyer system proceeds in two stages. In an initial *offline* stage, it processes a corpus of diagrams and prepares them for search. In the subsequent *online* stage, DiagramFlyer offers three distinct methods for giving users access to the diagrams.

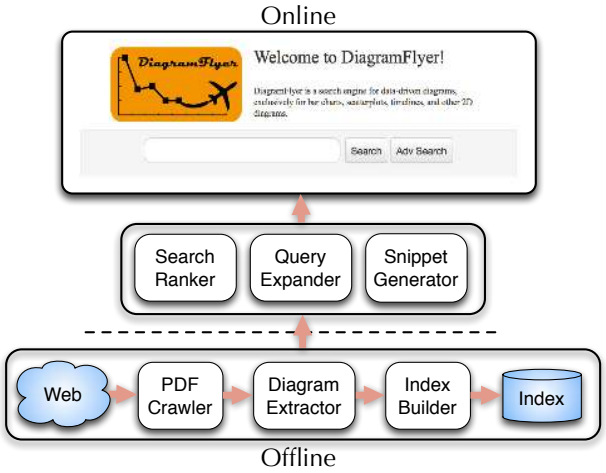

Figure 3: DiagramFlyer**'s data processing pipeline.**

The system architecture is seen in Figure 3. It employs a pipeline of offline corpus-processing steps that produce output then used by an online search query system. The ***offline*** pipeline has three components.

First, the ***PDF crawler***, which is based on the Nutch open-source crawler [3], downloads a large number of PDFs from public web pages on academic Internet domains. In our current testbed we concentrate on diagrams extracted from web-hosted scientific PDFs (found by targeting .edu websites). We found 153K documents. We focus on PDFs that contain diagrams with explicit text. Thus, we avoid the use of optical character recognition (OCR) software. Although the quality of many OCR systems is reasonably high for basic document types such as book pages and business cards, the significant modification they require to function on diagrams is beyond the scope of this paper.

These PDFs are then fed to the ***diagram extractor***. This extractor identifies all the diagrams in the corpus and extracts their metadata at the same time. The system extracted 319K diagrams (*i.e.*, slightly more than 2 diagrams per paper). For the testbed system we target two-dimensional data-driven plots (including scatter, time series, and bar plots) as these have been found to represent a large portion of data-driven diagrams (*e.g.*, 70% of diagrams found in news magazines [12] are time-series). More detail about the *diagram extractor* can be found in [5].

Finally, the ***index builder*** uses Lucene [1] to construct an inverted search index over the extracted and annotated diagrams. The index tracks each extracted field separately so that keyword matches on individual parts of the diagram can be identified.

All three parts of the ***online*** query system are implemented in Java running as a web application, using Lucene for query processing during inverted index retrieval. They are the *search ranker*, *query expander*, and *snippet generator*.

Search Ranker — Given a keyword query, the *search ranker* computes a relevance score for each diagram and presents a ranked list of diagrams as the results. We implemented the scoring mechanism, *weight-rank*, for assessing a diagram's relevance to a user's query using Lucene [1]. The *weight-*

rank mechanism looks for matches in each distinct metadata field of a searching diagram, then computes the standard TF-IDF relevance score of each metadata field. It allows each of the eight fields to have a different weight when computing the total diagram relevance score. We obtained the weights by using a Support Vector Machine to find the optimal weight assignment based on a supervised training set of more than 430 human-annotated *(query, diagram, relevance)* triples. By separately finding search hits among fields that are distinctive and meaningful, a ranking system has greater ability to assign useful (and different) weights to each field. The ranker then sorts diagrams according to their relevance scores and presents the top results to the user.

Query Expander — The *query expander* extends the query to retrieve a broader range of relevant diagrams. We aim to recover diagrams that are related to a target diagram but have a connection obscured by the lossy production pipeline.

The main component of the *query expander* is a lexicon generator built on 14 million HTML lists crawled from ClueWeb09 [6]. We used the lexicon generation algorithm proposed in [11]. Given a term, a lexicon generator produces a ranked list of terms belonging to the same category. For example, the lexicon generator might take "Michigan" as input and emit many other states in the US. The web list dataset is not domain specific, and thus we believe our lexicon generator can cover a large number of different topics.

The *query expander* chooses a single expansion for each user query term (to avoid the resulting search query to strongly favor one term over another). The lexicon entry we choose for the expansion of query term t_s will be the lexicon term that maximizes the lexicon similarity score the lexicon similarity score $S_{Lex}(t, t_s)$. Given a query term t_s, let $L = Lex(t_s)$ be its generated lexicon and t be a term in a searching document d. A direct way to measure the semantic similarity between t_s and t is to measure the probability of how often t_s and t co-occur in the same list, as $S_{co}(t_s, t)$. But web lists are noisy, so directly using $S_{co}(t_s, t)$ to represent how close the two terms are can be misleading. Thus we compute t's lexicon similarity to t_s based on two parts: similarity of the document term to the query term and to the query term's overall generated lexicon:

$$S_{lex}(t, t_s) = S_{co}(t, t_s) + \frac{1}{|L|} \sum_{t' \in L} S_{co}(t, t') \qquad (1)$$

Snippet Generator — Finally, the *snippet generator* generates a brief visual summary of each search hit in the SERP, as shown in Figure 1. Textual snippets in traditional web search are a query-relevant compact document representation that help users scan the result list quickly and find high-quality matches. To achieve these goals in DiagramFlyer, we annotate a thumbnail image (*i.e.*, a scaled image of the diagram) with diagram metadata. We found the annotation to be useful as the text in a scaled-down thumbnail image is often too difficult to read. By overlaying text in a larger font on top of the diagram thumbnail we allow the end-user to quickly identify good matches in the SERP list.

4. DEMONSTRATION

The online demo video is available on YouTube[1]. We demonstrate the working data-driven diagram search system DiagramFlyer via the following three functions.

Keyword Search — First, DiagramFlyer supports keywords queries. For example, when a user types the search query "birth rate" in the search box, she is able to browse a ranked list of diagram objects. The snippet (as shown in Figure 1) presents all the extracted elements of a diagram specification for fast browsing. In addition, the user can reach back to the original document by clicking its URL.

Advanced Facet Search — The DiagramFlyer also supports querying by the diagram generating process. By clicking the "Adv Search" button on the search interface (as shown in Figure 1), a user can query by the diagram template language (as shown in Example 1 & 2). For example, a user can easily obtain diagrams with "year" to be the x-axis from 1990 to 2000 and with "population" to be the y-axis using the "Adv Search" interface.

Find Similar Diagrams — The query language of DiagramFlyer makes it possible to support many interesting application, including finding similar diagrams. For example, when a user gets the initial results of querying for "birth rate" related diagram, she can click a resulting diagram's x-label to find all the diagrams with a similar x-label. The user can also click "Find similar diagrams" on the top of each resulting diagram snippet (as shown in Figure 1) to obtain a list of similar diagrams.

5. CONCLUSIONS

DiagramFlyer is a working search engine that searches 319k diagrams extracted from thousands of PDFs in the web. We have implemented the software architecture and algorithms for implementing DiagramFlyer's diagram pipeline extractor, as well as query tools that perform diagram relevance ranking, similar item finding via lexicons, and snippet generation. In addition, we have demonstrated that the system is able to perform many interesting applications, including traditional keywords search, advanced facet search, and searching for similar diagrams.

6. REFERENCES

[1] Apache Lucene, http://lucene.apache.org/java/docs/index.html.
[2] S. Bhatia, P. Mitra, and C. L. Giles. Finding algorithms in scientific articles. In *WWW*, 2010.
[3] M. Cafarella and D. Cutting. Building nutch: Open source search. *ACM Queue*, 2, 2004.
[4] S. Carberry, S. Elzer, and S. Demir. Information graphics: An untapped resource for digital libraries. In *SIGIR*, 2006.
[5] S. Z. Chen, M. Cafareela, and E. Adar. Searching for statistical diagrams. In *Frontiers of Engineering, National Academy of Engineering*, pages 69–78, 2011.
[6] ClueWeb09, http://lemurproject.org/clueweb09.php/.
[7] 2011. D8taplex, http://d8taplex.com/.
[8] 2011. DataMarket, http://datamarket.com/.
[9] 2011. EidoSearch, http://www.eidosearch.com/.
[10] J. Goldsmith, A. Wong-Foy, M. Cafarella, and D. Siegel. Theoretical limits of hydrogen storage in metal-organic frameworks. *Chemistry of Materials*, 2013.
[11] Y. He and D. Xin. Seisa: Set expansion by iterative similarity aggregation. In *WWW*, pages 427–436, 2011.
[12] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 2001.
[13] L. Wilkinson. The grammar of graphics. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2005.
[14] 2011. Zanran, http://www.zanran.com/q/.

[1] http://youtu.be/B7I1_o23N38