VizSnippets: Compressing Visualization Bundles Into Representative Previews for Browsing Visualization Collections

Michael Oppermann and Tamara Munzner, *Senior Member, IEEE*

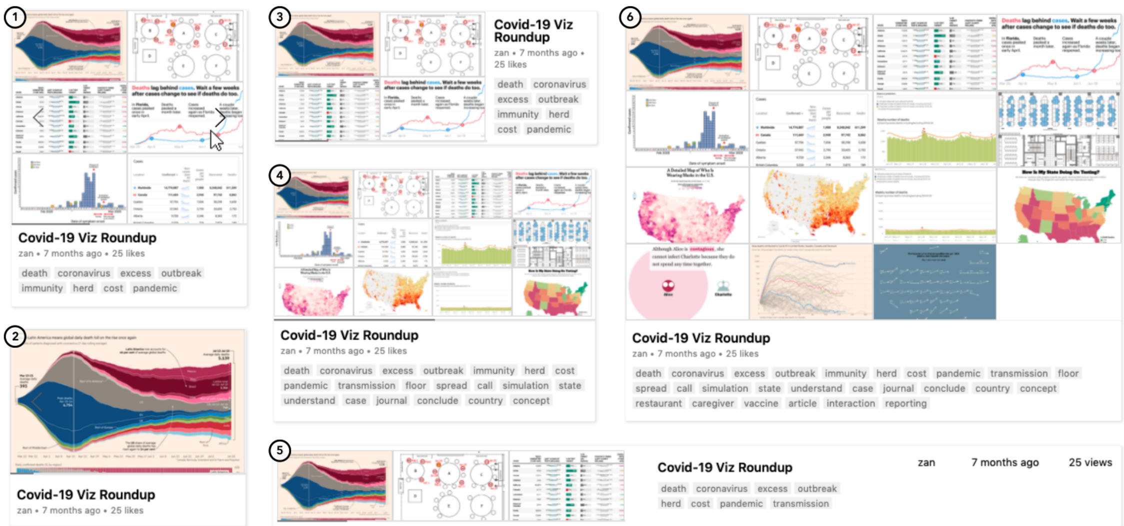

Fig. 1. Automatically generated snippets of the same Observable notebook [3]. (1) Vertical snippet layout with image collage of six images, and image carousel to scroll through additional content; (2) Existing visualization snippets in commercial products are limited to a single thumbnail; (3) Horizontal snippet layout; (4) Small minimum image size allows more images to fit in a collage; (5) Table layout; (6) Large preview snippet minimizes information loss.

Abstract— Visualization collections, accessed by platforms such as Tableau Online or Power BI, are used by millions of people to share and access diverse analytical knowledge in the form of interactive visualization bundles. Result snippets, compact previews of these bundles, are presented to users to help them identify relevant content when browsing collections. Our engagement with Tableau product teams and review of existing snippet designs on five platforms showed us that current practices fail to help people judge the relevance of bundles because they include only the title and one image. Users frequently need to undertake the time-consuming endeavour of opening a bundle within its visualization system to examine its many views and dashboards. In response, we contribute the first systematic approach to visualization snippet design. We propose a framework for snippet design that addresses eight key challenges that we identify. We present a computational pipeline to compress the visual and textual content of bundles into representative previews that is adaptive to a provided pixel budget and provides high information density with multiple images and carefully chosen keywords. We also reflect on the method of visual inspection through random sampling to gain confidence in model and parameter choices.

Index Terms—visualization collections, visualization bundles, result snippets, visual inspection

1 Introduction

People across organizations and sectors now have access to growing troves of data and collectively amass large collections of visualization content. Business intelligence (BI) tools, such as Tableau, Power BI, Qlik, and Looker, are used daily by millions of people. The primary artifacts created with these tools are *visualization bundles*, that combine multiple visualizations, dashboards, and data sources. Bundling content to organize, save, and share analysis results is a common practice even beyond BI tools, through computational notebooks such as Observable that encompass substantial amounts of diverse visualization content.

When users browse or search collections of visualization bundles, they regularly need to choose between multiple results and judge their relevance. To avoid the time cost of loading and examining in detail these results one by one, bundles are summarized as compact previews, which we call *snippets*. However, existing snippet designs yield poor information density: they fail to help people judge the relevance of bundles because they include only one image and a title, falling far short of communicating the full information content of a bundle that may include multiple views and dashboards. We engaged with Tableau product groups and end users to determine that this problem has substantial real-world impact, and reviewed snippet designs across five major tools to establish that it is pervasive.

Although guidance exists for snippet design with data types such as text, images, and video, we are the first to systematically approach visualization snippet design. We propose a computational pipeline to compress visualization bundles into representative snippets, providing visualization system designers with a set of adjustable algorithmic building blocks. It has many controllable features including the ability to adapt to a given pixel budget and to create flexible families of layouts with desired form factors. We also present practical guidance on how visualization snippet designers can use the pipeline to support many use cases through concrete examples.

• *Michael Oppermann and Tamara Munzner are with the University of British Columbia. E-mail: {opperman,tmm}@cs.ubc.ca*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

We contribute:

- A set of eight key challenges pertaining to visualization snippets, and a top-down design framework for visualization snippets with three levels (page, snippet, images+text) and two categories of design choices (size/number of elements, visual layout/encoding).
- A computational pipeline for the lossy compression of visualization bundles into snippets, consisting of a set of algorithmic building blocks to filter, rank, and display visual and textual content in accordance with an available pixel budget.

We validate the pipeline's algorithmic building blocks on two kinds of visualizations bundles, from Tableau Public and Observable, using quantitative measures, visual inspection, and A/B comparisons. For this purpose, we also labeled several thousand images to compare human judgements with model predictions.

As a secondary contribution, we promote a methodology for extensive visual inspection through random sampling. We reflect on the creation of multiple lightweight visual inspectors, each tuned for one question or algorithm, that sample a substantial corpus of real-world data to gain confidence in algorithm and parameter choices.

Inspired by the work of Rogers et al. [52], we include direct links to research artifacts (SUP1 to SUP7) throughout the paper to transparently provide an abundance of evidence for our claims and our process.

2 Visualization Collections and Snippets

We provide a data characterization and describe the collections we use to assess our work, define snippets in the context of visualization collections, survey current practices in snippet design, and discuss our investigation of real-world challenges through engagement with Tableau product groups. We identify key challenges and present a top-down framework for snippet generation to address them. We finally discuss the intended context of use by snippet designers for our computational pipeline.

2.1 Data Characterization and Sources

We define visualization bundles and collections in general, followed by a description of two sample collections that we acquired.

2.1.1 Bundles and Collections

Visualization bundles combine single views (charts), dashboards combining multiple views [53], and one or more datasets. We categorize the content into four groups, while noting that some bundle types contain only a subset of this information.

Bundle meta-data contains the bundle title, author name, date, and usage statistics, such as the number of views and likes.

Bundle text is bags of words in each of three categories: 1) text within views, including chart and axis titles, captions, and titles; 2) data column names; 3) longer prose text that may be attached.

Bundle images are stored as PNG files, each corresponding to a view or dashboard. All images conform to the same 4:3 aspect ratio, and may be cropped or padded with white space to fit those dimensions.

Bundle specification contains additional layout and chart type information, for example which views are embedded in a dashboard.

We refer to repositories of manually crafted bundles that can be browsed on dedicated platforms as *visualization collections*. The number of bundles in a collection ranges from hundreds to thousands or more when shared within an organization, or even millions in special cases that are open to larger communities, such as Tableau Public.

2.1.2 Data Sources

We used two collections of real-world visualization bundles to develop and evaluate the proposed computational pipeline.

Tableau workbooks are a widely used type of visualization bundle, containing all necessary components that are required for loading and displaying the fully interactive visualizations to users.

We build on our previous work on visualization recommendation [48], where we extracted bundle data from the XML files of a set of 2910 hand-crafted workbooks randomly sampled from Tableau Public, a sprawling collection of publicly shared workbooks. Other business intelligence tools, such as Looker, Qlik, and Power BI, use similar approaches to bundle visualizations and data sources into workbooks, reports, or apps, so our findings are generalizable beyond Tableau.

Observable is a reactive computational notebook for data analysis and visualization, used by a rapidly growing community. A notebook is made up of a series of cells containing JavaScript code, prose, interactive visualizations, and images. This structure follows the paradigm of literate programming [31] where explanations in natural language are interspersed with code snippets. Although the use cases related to computational notebooks may be very different to those facilitated by BI tools, they exhibit intriguing content similarities; thus we also consider those notebooks as visualization bundles.

We implemented a Chrome extension to scrape a diverse sample of 178 publicly accessible Observable notebooks. Our tool takes screenshots of Canvas and SVG areas, downloads embedded images, extracts meta-data information, and creates a bag of words from the text found in Markdown cells. All visualizations are treated as single views because inferring the composition of dashboards from source code is non-trivial, and the actual number of dashboards on Observable appears to be very low (0 in a sample of 40 notebooks). [SUP2] contains the source code of the scraper and the raw data of our Observable collection.

2.2 Visualization Snippet Usage

We define visualization *snippets* broadly as compact, representative summaries of visualization bundles that help users discriminate between relevant and irrelevant content in visualization collections. Use cases for snippets within visualization platforms include showing search results, supporting faceted browsing, and suggesting recommendations. The form factors vary: many medium-sized snippets can be shown within a large window as with search results, or a few snippets within a small region of the screen as with recommendation results, or a single large snippet may appear upon hover in a large popup window to preview bundle content with as much detail as possible.

We provide an example scenario to illustrate the search use case for snippets. A marketing analyst searches the company-wide visualization collection to check if any colleagues have previously created a report on product downloads. The collection does have some hierarchical organization into folders, but that structure does not help them scope their search. A query results in multiple pages with over a dozen snippets each. The analyst rules out some bundles as irrelevant from the title and single image visible from the snippet, but often must open the bundle to check whether it contains relevant materials. The analyst spends several minutes opening bundles and clicking through each of their views, only to realize that they are outdated or concern a different product, before ultimately locating the desired analysis report.

In the information retrieval process, snippets concern the *relevance prediction step* [20] where users predict which items summarized in a results display align with the current information need before they open the detailed versions [23]. Opening many visualization bundles can be a time-consuming and tedious endeavour because of substantial load, build, and explore times, especially when visualizations contain many views or are connected to live data streams. Snippets serve as previews to facilitate the relevance prediction step as surrogates that avoid loading entire bundles via visualization platforms.

2.3 Embedded Inquiry Within Tableau

This work grew out of collaboration with the *Recommendation* and *Search* product groups at Tableau, following up on our previous investigation on content-based recommendation models for visualizations [48]. Over a period of more than a year, we had conversations regularly with product managers, machine learning engineers, and UX designers, who relayed end-user feedback and pain points about many issues to us.

The key realization that motivated this research project is that the shortcomings of existing snippet designs constitute a significant pain point for Tableau end users. Specifically, we have substantial evidence that it is difficult and often impossible for users to judge a bundle's relevance based on the given snippet, without opening it. We heard this idea from multiple people within Tableau. We verified this common internal perception through direct interviews with external target users.

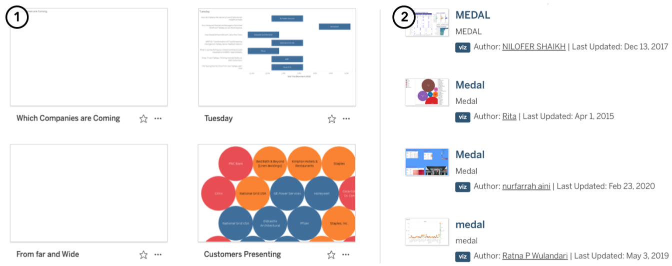

Fig. 2. Examples of existing snippets with low information content in their single images and titles: (1) Tableau Online, (2) Tableau Public.

The first author of this paper joined several interviews with target users to evaluate a new visualization recommendation model; despite the intent to focus on the back-end model, participants repeatedly commented on the presentation of the recommendations that were shown as visualization snippets, confirming the problem of insufficient visible information content.

We also observed that visualization authors commonly start from scratch instead of reusing existing content, even in contexts where re-use would seem to provide benefits; we conjecture that better snippet design might combat this tendency by helping authors find relevant content more easily.

Finally, we noted that when developing new product features, product groups either simply re-use existing snippet designs or end up making custom modifications based on intuition, without systematic reasoning of how snippets could be explicitly designed to support platform use cases or user analysis goals.

2.4 Review of Existing Snippet Designs

To ensure generality beyond the particular platform of Tableau, we informally surveyed the content and layout of existing snippet designs used in Tableau Online/Public, Power BI, Qlik, Looker, and Observable. All existing snippet designs show only the bundle title and one image, and sometimes a few meta-data attributes. In many cases these titles are not very descriptive and the single image has low information content, as shown in Fig. 2, so they fail to support users when assessing the relevance of bundles.

None of the snippet approaches that we surveyed incorporate multiple images; in all cases a single image thumbnail of a view or dashboard is selected and cropped automatically. For example, Observable chooses the first image, SVG, or Canvas element found in a notebook. Even if a bundle contains useful content, the automatically selected images may show incomplete draft views or appear empty when scaled down to a thumbnail size. Only a few tools allow creators to upload custom thumbnails that may include illustrations that reflect the bundle context but are not directly related to the data [30]. The thumbnail size varies substantially between platforms.

In terms of text content, the bundle title is always prioritized, although sometimes significantly truncated. The only approaches that include a textual description are Tableau Public's gallery view that presents human-curated bundles and Qlik's list view. We note that creators are frequently unwilling to invest the additional effort of crafting a description. None of the existing snippet designs include keywords or other textual information extracted from the bundles such as tags, despite tagging functionalities in some of the tools. Meta-data attributes such as author name, date, and number of likes, are included in some snippet designs but the selection of these attributes differs widely across platforms.

Snippets are arranged in a uniform grid layout by default in all tools, except Tableau Public which uses a list layout. The option to choose between two different layouts is provided in 4 of the 6 platforms. The snippet size is always the same independent of the bundle content, and most of the pixel space is devoted to the thumbnail.

Further details are included in [SUP1], and Sec. 5.3 presents A/B comparisons between existing product snippet designs and the results of our proposed VizSnippets pipeline.

2.5 Challenges

Our two-pronged investigation led us to identify eight challenges for snippet generation:

C-OneImage: Limited information content of a single image. A single image of one view or dashboard provides only a very narrow lens on a bundle and may not accurately capture the diverse visual content across multiple views. Blank thumbnail images are surprisingly frequent within our sample collections.

C-Titles: Limited information content of titles. We learned through interviews and by reviewing the data samples that short and cryptic bundle titles pose a challenge across all collections. Uninformative titles in similar forms such as "sub-2", "Project 8", "Final Vis", and "Untitled" are seen frequently. Vague titles impede the identification of relevant content for non-authors, and community-generated content is particularly messy.

C-Pixels: Limited pixel space. Pixels are a precious commodity that need to be used as effectively as possible, on the desktop and especially for mobile. The full content of a bundle can rarely be shown in its entirety, necessitating compression.

C-SparseText: Limited textual content. The text found in bundles that are created with BI tools generally consists of fragments of a few words, such as titles or labels. Although an author might sometimes include captions or annotations of lengthier text, the amount of available text is far less than in standard text documents [48]. This challenge is not directly applicable to Observable computational notebooks, which commonly include prose with typical grammatical sentences similar to text documents, for instance when created as instructional guides.

C-Form: Diversity of form factors. The wide variety of use cases for snippets leads to different form factor requirements, from many medium-sized snippets in a large window to support quick comparisons between bundles, to a few small snippets in a small region using minimal screen real estate, to a full-screen view of a single large snippet to provide a very detailed preview.

C-Complex: Diversity of bundle complexity. The content in the reviewed collections ranges from single bar charts to complex bundles with dozens of views and dashboards. Quantitatively, the bundles in our Tableau Public collection contain 2.19 dashboards and 4.11 views on average (the medians are 1 and 3 respectively). Views are often just building blocks for dashboards or contain auxiliary information.

C-Quality: Diversity of content quality. We found substantial differences in the quality of information content within bundles: the same bundle might have empty or incomplete drafts in addition to high quality finalized content, or many similar images alongside quite different ones. When summarizing bundles, high quality informative content should be elevated, while incomplete or empty or similar views and less informative text should be filtered or downgraded.

C-Goals: Diversity of end-user goals. We identified disparate end-user goals that could be supported by visualization platforms: *collaborate with colleagues*, *create portfolios to share work*, *find out how other people did something*, *get visual inspiration*, *look up information*, and *save all visualizations in one space*. These goals may lead to different snippet affordances. For example, images and chart types are paramount for visual inspiration, but information lookup might require more emphasis on the underlying data sources of a bundle.

2.6 Design Framework

To address these challenges, we propose a top-down design framework for snippet generation with three levels and two categories of choices, shown in Fig. 3. The top level pertains to the layout of a result *page*, and choices at this level affect options at the middle level of *snippet* and lowest level of *images+text* (see [SUP4] for all possible combinations). One category of design choices is the size and number of elements, and the other pertains to the layout and visual encoding. The computational pipeline proposed in Sec. 4 fully supports this framework.

We consider snippet generation through the lens of lossy compression, where the goal is to provide as much detail as possible within a given pixel budget. Although a very compact snippet that could be used within a grid of many snippets is a highly compressed representation of a bundle, another use case is a full-screen preview of a single bundle that could capture a substantial fraction of the salient content with limited loss.

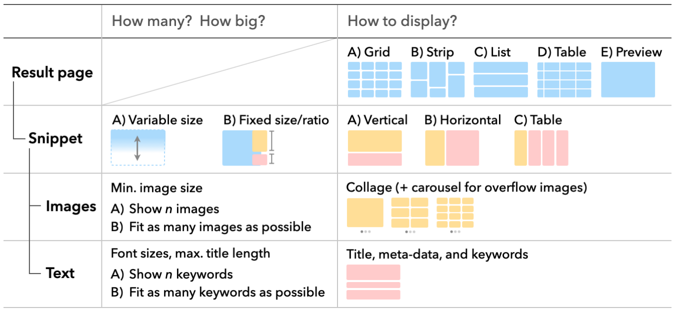

Fig. 3. Design framework for displaying visualization snippets, with three levels and two categories of choices.

2.7 Intended Framework and Pipeline Usage

We instantiate this design framework with a computational pipeline that delivers a powerful and flexible infrastructure to *visualization snippet designers* who want to automatically create snippets with greater information density than existing designs. We consider snippet designers to be a broad group, encompassing not only UX designers but others involved in creating or customizing snippets related to visualization collections, such as a team working on a recommendation system wanting to extend snippets to add explanations why something gets recommended [62].

We anticipate that snippet designers would use this pipeline after they ascertain the exact requirements for their particular use case. That characterization would probably require targeted user studies, which we leave to those designers rather than attempting to conduct any of them ourselves. Instead, we thoroughly validate the technical underpinnings and performance of the pipeline and its algorithmic building blocks on real-world data, and provide actionable recommendations on how it can be flexibly adapted for different scenarios.

We hope our work encourages the creation of representative snippets beyond single thumbnails with titles. The diversity of snippet form factors (C-Form) and end-user goals (C-Goals) reveals that different use cases would benefit from more specialized designs.

3 Related Work

We now discuss result snippets for a range of content types, and existing approaches to visualization compression.

3.1 Result Snippets

Our work is informed by previous studies that evaluated snippet representations. Result snippets have been investigated for a range of scenarios and content types, such as documents [21, 58, 60], source code [19], e-commerce products [44], books [40], music [72], videos [55, 68], images [14], and extensively for websites [5, 20, 28, 33, 39, 61] in the context of search engine result pages. Although the relative position of results remains one of the most influential factors for assessing their relevance [39], the back-end algorithms to find and rank bundle results are not the objective of this project; instead, our focus is specifically on the snippet content and its representation.

Several studies examined website snippets that contain thumbnails only and in combination with text summaries [5, 20, 34], and suggest that thumbnails used along with text summaries help in reducing errors when predicting relevance. Capra et al. [9] showed that images provide only a small benefit in judgment accuracy compared to text-only snippets. However, they emphasize that image-augmentation helped measurably in several scenarios when textual components are poor, which is the case for visualization bundles. Otherwise, the characteristics of a website are significantly different than visualization bundles.

Snippets for videos are particularly relevant for our work. Due to their intrinsic graphical and temporal nature, videos exacerbate the need for snippets that offer additional representation facets besides text. Wildemuth et al. [67] concluded that multi-image compositions or short video sequences are more expressive and favored by users over single images. Displaying multiple key frames of a video [24, 67] is similar in spirit, but different in detail, to the image collage technique that we propose in Sec. 4.3.5. Collages are composed of several selected visualizations and dashboards to provide a bird's eye view on the bundle content. However, due to completely different semantics, existing techniques for identifying highlight frames from videos cannot be directly applied to selecting visualization images from a bundle.

Natural language provides great richness in result snippets and is used beyond textual documents for other types of media, such as videos or photographs. Augmenting a snippet with a description or keywords has been proven effective in previous work [16, 22].

Recent approaches to automatically generate captions for visualizations are promising but are limited to single views and specific visual features [1, 35, 43]. In contrast, our goal is to summarize visualization bundles, with potentially dozens of views, on a high level. The automated creation of snippets for complex visualization bundles, as we propose in this paper, has not been studied previously.

3.2 Approaches to Visualization Compression

In the visualization literature, many approaches have been discussed to compress visual encodings into more compact representations, for example, to create overviews, to support comparisons across many items, or to allow consumption on mobile devices. This body of work spans across visualization techniques, including trees [37], graphs [4, 70], time series [13], small multiples [7], and infographics [8, 38]. In recent years, there has been a growing interest in responsive visualizations [26, 69]. In contrast, our goal is different from all these approaches. Rather than adapting visual encodings in response to different screen resolutions, we summarize complex visualization bundles containing existing images of multiple views and dashboards. The methods discussed in Sec. 4 are based on selecting from a given set of images; we do not propose new altered or derived visual encodings.

Most closely related is the work on visualization thumbnails. Heer et al. [25] proposed to include thumbnails of single views in graphical histories to support the visualization authoring process. Kim et al. [30] surveyed current practices in creating visualization thumbnails, specifically for data stories. Our work does not address generating new thumbnails, but rather provides computational methods for selecting from a set of existing thumbnails to generate snippets.

4 VizSnippets Computational Pipeline

We propose a pipeline to compress visualization bundles into representative snippets, that is adaptive to given space constraints (C-Pixels and C-Form) and other user-defined preferences. The pipeline, illustrated in Fig. 4, consists of multiple interchangeable algorithmic building blocks with minimal dependencies. The extraction (green box) and lossy compression of the bundle content (yellow boxes for images, pink for text) is completed in an offline preprocessing step. The creation and arrangement of the snippets (blue box) is determined at run time.

4.1 Visual Inspectors

We developed and evaluated the building blocks iteratively by using a suite of eight *visual inspectors*, leveraging an extensive corpus of real-world data. These lightweight inspectors *I1* to *I8*, shown in Fig. 5 (and [SUP3]), are targeted at specific algorithms or analysis questions and present results based on random data samples. This type of extensive visual inspection allowed us to gain further confidence in model choices and parameter settings, beyond standard quantitative analyses. We elucidate how inspectors informed our decisions throughout the following sections, and reflect on this approach as a generalizable method to provide guidance and to foster transferability in Sec. 6.

4.2 Page and Snippet Layouts

We support five general page layouts that control how snippets are arranged: the *grid* layout divides a page into rows and columns, and all snippets have a uniform size (fixed width, fixed height); the *strip*

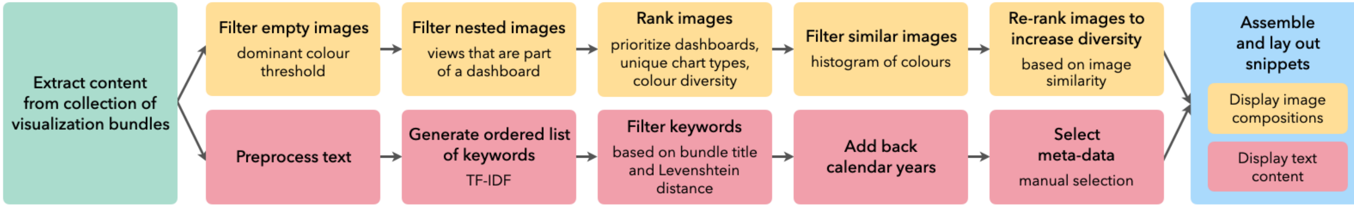

Fig. 4. VizSnippets computational pipeline to compress visual and textual content of visualization bundles into representative snippets.

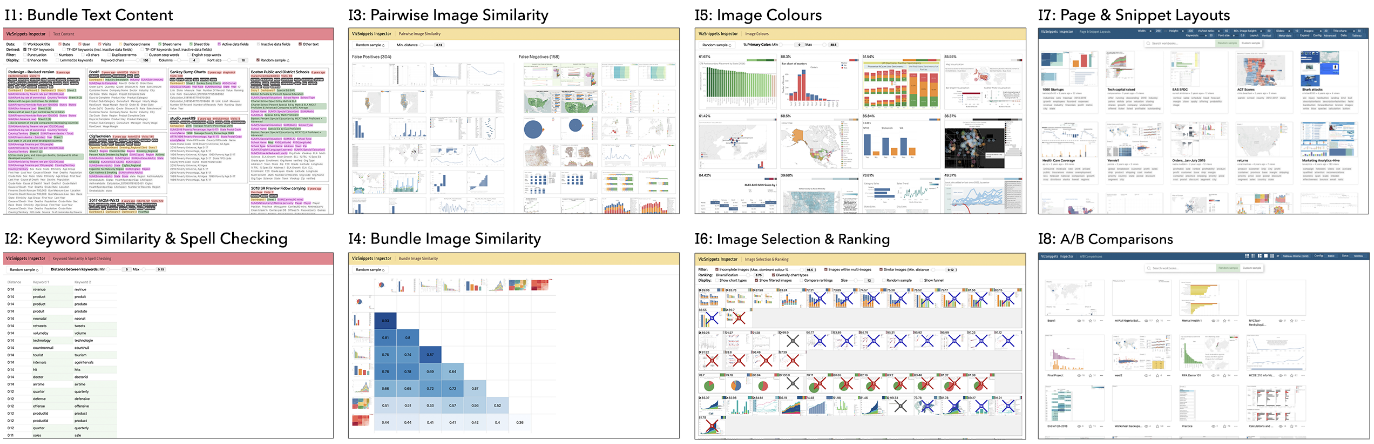

Fig. 5. Overview of visual inspectors that we used to analyze models and parameter settings. High-resolution screenshots are in [SUP3].

layout, also known as masonry layout, divides the page into equal sized columns (fixed width) with variable snippet heights; the *list* layout displays one snippet per row (full width, variable height); the *table* layout also shows one snippet per row but meta-data attributes are divided into separate sortable columns and the snippet height is fixed; the *preview* layout allocates all pixels of the page to a single large snippet. These five layouts support all snippet use cases and form factors (C-Form) that we identified, and, due to the modularity of the pipeline, more options can be flexibly added. To optimize the efficacy of snippets, layout choices should be made according to tasks and typical bundle characteristics in a visualization collection. We discuss design rationales and potential usage scenarios for different layouts in [SUP4]

The initial page layout selection has cascading effects on lower levels and dictates certain design decisions. For example, the page layout dictates if a snippet has a variable or a fixed size. With the grid layout, we can choose a specific snippet size and decide what ratio of pixel space should be devoted to visual versus textual content (C-Goals). With the strip layout, the snippet height is variable and depends on other parameters related to the text and image content (C-Complex). The actual number of snippets that are visible on screen depends on the snippet size and the chosen page layout.

We provide three options to arrange the visual and textual content within a snippet: a vertical, horizontal, or table layout. We deemed more complex compositions such as wrapping text around images to be less important for the use cases we identified, but these could be added in the future. The snippet layout defines the canvas that will be filled with text and image content. We describe the algorithms for compressing and displaying the bundle content in subsequent sections.

4.3 Snippet Images

Our investigation confirmed that images are particularly relevant for visualization snippets. Besides conveying the visual style and helping users to recognize previously seen content [33, 61], images capture semantic information. The chart type can indicate underlying data types or the visualization objective. For instance, maps imply geographic data and pie charts show part-to-whole relationships.

We now describe the algorithmic building blocks for selecting and displaying composite images to increase the information content of a snippet beyond what a single image can convey (C-OneImage). The yellow boxes in Fig. 4 show the image-related blocks: filter empty and embedded images, filter nested images, rank images, filter similar images, re-rank images to increase diversity, and display image compositions.

4.3.1 Filter Empty and Embedded Images

In a first step, we filter empty and nearly empty images based on a dominant colour threshold C_T. We convert each image to a list of RGB pixels (points in 3D space) and use the *k-means* clustering algorithm to extract the k dominant colors of an image [57], where k corresponds to the number of cluster centroids that are used to iteratively assign the pixels to clusters. We compute the relative size of each cluster, namely how many pixels of the original image are assigned to each dominant colour, to determine if a single colour predominates an image, signalling that it is less informative (C-Quality). A dominant colour threshold C_T of 98.2% led to the highest accuracy for the sample visualization collections, with $k = 5$ colours. Further details about determining these parameters are included in Sec. 5.2.

For visualization bundles that contain dashboards, we avoid repetition by filtering out images of the embedded views that were the building blocks used to create them. Prioritizing dashboards makes wise use of available pixel space (C-Pixels). This process is straightforward for Tableau data because the views that are embedded in a dashboard are defined in the bundle specification. All images extracted from Observable notebooks are treated as single views.

4.3.2 Rank Images

We create an initial ranking of all images within a bundle. In many cases, the available images do not fit within the allocated pixel budget so some images are hidden or only revealed after user interaction, as we will describe in Sec. 4.3.5. Consequently, the order of images is relevant to increase the explanatory power during the first impression. We divide the images into two groups: dashboard images and single-view images.

Dashboards are prioritized and sorted by the number of embedded views and colour diversity. The percentage of the most dominant colour that we computed earlier is used as a proxy measure for colour diversity. Through visual inspection with *I5* (see Sec. 5.2), we discovered that mostly single-colour images are less informative because they mostly show a background color.

We group all single-view images by chart type, sort the images within each group by colour diversity, and sort all groups by their maximum colour diversity. Then we iteratively withdraw one image from each group until all groups are empty. All the images get assigned a consecutive number that is used as an initial score. When chart types are not available, such as in Observable collections, the images are only sorted by colour diversity.

4.3.3 Filter Similar Images

Bundles frequently include very similar or identical views that should be filtered (C-Quality). We compute pairwise distances between all bundle images and filter out those where the distance is below a given threshold. For determining the distance between images, we compared three traditional approaches based on hand-crafted visual features, histogram of colours (HoC), histogram of oriented gradients (HoG) [18], and structural similarity index (SSIM) [64], and the more recent approach of learning visual features and a similarity function through a Siamese convolutional neural network (CNN) [63]. The performance of the CNN was comparable with the best hand-crafted feature approach for this particular task of identifying very similar or identical images, so we chose the less complex, unsupervised method (HoG). See further details in Sec. 5.2.

4.3.4 Re-Rank Images to Increase Diversity

After applying the previous steps, similar images might still be ranked high and close to each other. The chart type specification may not be available or cannot be detected, or views use the same base chart type.

We propose a re-ranking step using *maximal marginal relevance* (MMR) [10] to ensure that the visual diversity of a bundle is conveyed by a very small number of images. MMR is a greedy, iterative algorithm that linearly interpolates between the original ranking score of an image and the diversity to other, already selected images. The algorithm is typically used in diversity-based text retrieval scenarios [10,47] but has been also applied to image re-orderings [56].

The MMR algorithm hinges on the similarity model in Sec. 4.3.3 to accurately compare images. First, the top-ranked image is selected based on the initial score and all remaining images are considered as candidates. Then, the algorithm iteratively selects the image with the highest MMR score $\hat{s} := \lambda \cdot s - (1-\lambda) \cdot MaxSim$, terminating when all candidates have been assigned to the ranking. The relative weight between the original image score, s, and the maximum similarity, $MaxSim$, is specified with a single tunable parameter λ. For example, $\lambda = 1$ ranks entirely by s and $\lambda = 0$ selects maximally diverse images irrespective of the initial ranking described in Sec. 4.3.2. We found that $\lambda = 0.75$ leads to diverse rankings, as suggested in a systematic MMR evaluation on comment diversification [47].

4.3.5 Display Image Compositions

We propose *image collages* to show multiple images within a snippet. A collage is composed of equal-sized images that are displayed in a uniform grid layout. The number of images depends on the chosen snippet size (see Fig. 3): (a) for a variable snippet size, designers specify the number of images and the image size, and the overall collage size expands accordingly; (b) for a fixed snippet size, designers specify the minimum image size and the algorithm tries to fit as many images as possible into the collage.

The construction of the collage layout is based on a grid packing algorithm that determines the optimal number of images iteratively. When there are more image slots than images available, the image size gets increased to make optimal use of the given pixel space (C-Pixels). Although text within the image is not legible, the chart types and visual styles can often be identified.

In some cases, the selected images exceed the number of slots within a collage. Besides merely hiding overflow images, one option is to create an *image carousel* that consists of multiple collages and allows users to scroll through additional content. The supplemental video [SUP7] shows image collages and carousels in action, and we discuss limitations and opportunities of multi-image compositions in Sec. 5.4.

4.4 Snippet Text

Our pipeline supports three kinds of text displayed within a snippet: the bundle title, keywords, and meta-data. All the BI tools that we reviewed use only the user-specified bundle titles, but our finding that these titles may not accurately reflect the semantic content of bundles (C-Titles) leads us to a more sophisticated approach. We argue that a small set of carefully chosen keywords (or tags) can provide valuable insights without consuming too much space.

The pink boxes in Fig. 4 show the process for extracting keywords and displaying text content in a snippet: preprocess text, generated ordered list of keywords, filter keywords, add calendar years, select meta-data, and display text content.

4.4.1 Process Text and Extract Keywords

We distinguish between two different bags of words that are extracted from a bundle and used as input data, taking care to glean all information from the limited textual content (C-SparseText). First, the *visible text* includes chart and axis titles, sheet names, captions, annotations, and descriptions. Second, if available, the column names from data sources are used as additional text data and referred to as *hidden text*. This hidden text can provide a useful signal about the bundle contents [48], to address the problem of limited information content of titles (C-Titles). Both sets can contain human readable text that may, but does not always, capture meaningful semantic information. The goal of this process is to select and rank keywords that best represent the underlying bundle content.

We remove punctuation, strings with less than three characters, numbers, and English stop words. We also lemmatize the text and apply a custom stop-word filter of 349 words we identified by analyzing frequent but non-informative words or phrases in our sample collections.

We use the *term frequency-inverse document frequency* (TF-IDF) model [51] on both visible and hidden text to extract keywords. Up to 30 keywords are extracted for each bag of words; this threshold is relatively high, and in many cases fewer keywords are available.The numerical TF-IDF-weighted scores of the terms are used for the ranking.

We remove keywords that are included in the bundle title to avoid repetitions. Although we lemmatize words to capture variant forms of the same word, very similar keywords may still get selected because they are not in the lexical database. To increase the diversity of top-ranked keywords, we compute *Levenshtein distances* between pairs of words and filter similar ones [50]. The Levenshtein distance is a measure for the number of edit-operations that are needed to transform one word into another. A normalized distance of 0 means that strings are identical and 1 if strings are completely different. We identified 0.15 as a useful threshold (see Sec. 5.1).

We give precedence to *visible text* irrespective of the TF-IDF scores, and compute the distance between those keywords. If the distance between two words is lower or equal to 0.15, we check the spelling [41] and choose the alternative that is spelled correctly. We find that this mechanism significantly improves the quality of keywords. We then combine the two sets of keywords by iteratively selecting a *hidden text* keyword and comparing the distance to already selected keywords until we have reached the desired threshold.

Finally, we attempt to extract calendar years from the removed set of numbers and add them back to the keywords. We discovered that calendar years are commonly mentioned in visualizations but often not included in the bundle title. We parse the text for four-digit numbers within the range of 1900 to 2050. We conjecture that this range will result only in a small number of false positives, based on our informal manual inspection [SUP5]. We combine sequences of years into single strings, such as *2019-2021* instead of *2019, 2020, 2021*.

4.4.2 Display Text Elements

The core text element is the bundle title specified by the author. The pipeline supports title truncation on demand, but we note it requires careful consideration because longer titles are more descriptive. The titles otherwise remain unchanged because removing stop words or numbers would likely confuse authors.

The pipeline allows snippet designers to manually choose which meta-data attributes are relevant for their use case, such as the author, the number of likes, or recent date of opening.

Adding keywords to snippets follows a similar approach to displaying snippet images, where we distinguish between variable-sized and fixed-sized snippets. The priority of the keywords is predetermined by their order, as described in the previous section.

When the snippet size is variable, designers specify the number of keywords that should be displayed. In case of fixed-size snippets, we fit as many keywords as possible based on the individual title length and the meta-data attributes. More specifically, we add keywords successively and verify the size of the snippet bounding box until the given constraints are reached. This approach is computationally expensive for many results; for production environments, we suggest to predetermine the default number of keywords that should be displayed for specific snippet sizes.

In the table layout, the title and the keywords are displayed in one column and the meta-data attributes are split into separate columns. In case of all other layouts, the title, the meta-data, and the keywords are shown as three horizontal rows in the text area of a snippet. A high-level parameter controls the font size of the snippet text.

5 Results

We conducted substantial testing on the VizSnippets pipeline during and after its creation. We now present quantitative and qualitative results to demonstrate its utility, validate our claims, and discuss trade-offs related to algorithms and parameter choices. Due to page constraints, we mainly report results based on the larger Tableau Public collection; further details related to the Observable notebooks are in [SUP5].

5.1 Snippet Text

We used visual inspectors (*I1*, *I2*) and descriptive statistics to analyze the textual content of bundles and snippets.

We noted that the titles often provide little informative value, confirming our early presumptions. The average length of bundle titles in the Tableau Public collection is 21 characters (median: 18). If we would apply the same text processing steps on Tableau titles as we do for other text content, 16% of titles would be empty strings. The sample Observable titles are significantly longer and contain 31 characters on average (median: 28).

We examined the raw text content of Tableau bundles using *I1*, which indicated that the hidden data column names can serve as useful auxiliary data but are often less clear than chart-visible text, confirming our choice to select keywords from visible text first and referring to hidden text only if room remains. The percentage of correctly spelled words is 93% for visible text and 88% for hidden text.

For the Levenshtein distance to filter very similar keywords, we determined the threshold of 0.15 (on a scale from 0-1) through visually inspecting hundreds of keyword pairs using *I2*. For larger distances, words frequently have different meanings, in contrast to smaller differences that are typos or word variations not captured during lemmatization. After applying the filtering steps, the average number of keywords decreases from 20 to 16 (the median is 13).

In assessing the extraction of calendar years that are not included in the bundle title, we found that 19.7% of all bundles contain annual details (within the range of 1900-2050) and visually verified that these years provide valuable information when comparing multiple bundles, such as reports covering different time periods. In 67% of cases, none of the extracted years are mentioned in the bundle title.

5.2 Snippet Images

The average number of extracted images per bundle is 5.3 (the median is 4). After applying the filtering steps, the average number is only 2.5 (the median is 2), which demonstrates the potential for compressing visual content with minimal information loss.

5.2.1 Empty Images and Colour Diversity

To evaluate the algorithm for filtering empty and nearly empty images using a dominant colour threshold, we manually labeled 2600 images using binary labels. We then tested the accuracy for varying colour thresholds C_T and different numbers of clusters ($k = [3,5,7]$), where k corresponds to the number of dominant colours. $k = 5$ and $k = 7$ led to similar results but the performance of k-means clustering decreases with more clusters. The highest accuracy of 93%, with $k = 5$, came from setting C_T to 98.5%.

We also used visual inspector *I5* to further analyze what type of images are filtered. If C_T is too low, line charts with minimal data ink may get falsely discarded, while a very high C_T may include images that contain mostly white space and provide no informative value. By using this inspector and adjusting C_T interactively, we observed that images with a high colour diversity are more informative when scaled down, compared to mostly single-colour images. Therefore, we incorporate the ratio of the most dominant colour in the image ranking.

5.2.2 Similar Images

We compared four approaches for computing pairwise image similarities. Three methods are based on hand-crafted visual features: histogram of colours (HoC), histogram of oriented gradients (HoG), and structural similarity index (SSIM). The fourth method is a Siamese convolutional neural network (CNN) trained on image pairs to learn deep representations and the similarity function automatically.

We manually labeled 6019 image pairs to evaluate the algorithms. We constructed pairs by randomly sampling two images from the same bundle, to increase the chance of similar images and because the algorithms are generally only used to compare images within and not across bundles. Image similarity was determined solely based on the visual style and not the underlying semantics. The deciding factor was if two images provide any additional value or if one representative image is sufficient. 35% of the image pairs were ultimately labeled as similar. We divided the pairs into a training and validation set with an 80/20 split for use with the CNN; the other three methods are unsupervised.

All methods reach a similar accuracy between 93%-94%. More details about the models and quantitative results are in [SUP5].

We implemented two visual inspectors (*I3* and *I4*) to qualitatively analyze the results. *I3* shows random samples of negatively predicted image pairs (false positives, false negatives). This tool allowed us to better understand when a model diverges from the human annotator. *I4* shows a similarity matrix of all images within a bundle to help us analyze the magnitude of values apart from the binary classification.

The quantitative results already showed that all methods performed similarly, except SSIM has a slightly lower accuracy. The visual inspection revealed that all models detect nearly identical images well but grapple with larger differences that the human annotator considered irrelevant, such as slightly different layouts, or a chart legend that is illegible when the image is rescaled to a thumbnail size.

The CNN contains only two convolutional layers and was trained on 4815 pairs and learned low-level features. It might learn better high-level concepts by using a deeper architecture and a sufficiently large training set. However, in practice, ample data to train a CNN for diverse collections may not be available. For our objective of filtering only very similar and identical images within a bundle, conventional unsupervised methods, such as HoC and HoG, are adequate and can be directly applied to new visualization collections. When similar images are not detected with less complex models, the MMR algorithm will ensure that these images are far apart in the final ranking.

5.2.3 Image Ranking and Collages

We created a visual inspector (*I6*) that reveals which images are getting filtered at which step, and how the original ranking changes after diversifying it based on chart types and image similarity. This type of qualitative assessment was crucial in understanding if combining various building blocks leads to the desired outcome or if important information gets lost during the image compression.

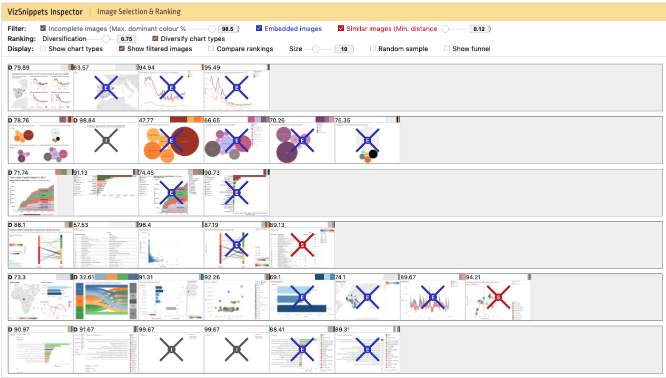

Fig. 6. Inspector *I6* to analyze the ranking and which images are filtered (I incomplete/empty, E embedded, and S similar views).

Fig. 6 is a screenshot of *I6* showing the images of 6 example bundles on a grey background. Coloured crosses indicate why images are filtered. The first 2 bundles are both reduced to a single image because of empty and embedded images. In general, we can see that most images are filtered because they are embedded in a dashboard (blue cross). The visual inspection also revealed that, with more complex dashboards, embedded views may end up very small on screen so keeping them as additional separate images might be helpful. Nevertheless, we made the trade-off to filter all embedded images because the median number of views per dashboard is only 2 and therefore sufficient details are visible in most cases.

We analyzed the number of images within a collage in relation to overflow images that are discarded or moved to additional collages in the carousel. After applying all filters, 87% of all bundles can be fully represented with collages of size 4. 94% of bundles contain 6 or fewer images. Instead of choosing a standard collage size, snippet designers only must specify the minimum image size and all images are resized or divided into multiple collages accordingly. Hence, the maximal fragmentation of a collage depends on the overall snippet size, which we will discuss below.

5.3 Page and Snippet Layout

We created the inspectors *I7* to analyze different page and snippet layouts, and *I8* to compare our layouts with existing snippet designs from the commercial tools Tableau, Observable, Power BI, and Qlik. Although these tools prioritize a uniform grid or list layout, we argue other layouts can be superior in some scenarios. Fig. 1 shows multiple automatically generated snippets of the same Observable notebook. Our pipeline supports trade-offs between compactness and information capacity, where the complexity of a bundle (C-Complexity) is directly linked to the amount of pixels used (C-Pixels). For example, snippets in a strip layout expand based on their content, so bundles with rich visual and textual content take up more space than bundles containing only a bar chart. These complexity differences are not apparent in a grid layout. The preview layout, with an example snippet shown in Fig. 1-6, goes one step further to serve as a minimum-loss representation of a bundle in return for larger pixel budget. The preview layout can, for instance, be used in a modal window to provide additional details instead of a time-consuming opening of the fully interactive version of a bundle. This use case is not supported in any of the reviewed tools, but our results show its promise.

Fig. 7 shows a comparison between a snippet of a Tableau workbook generated with our pipeline and three alternative designs used in commercial tools. This example is indicative for many other bundles we have seen in our collection. It consists of dashboards and single views, and the bundle title is vague. The three tools display only one image and a title (see Fig. 7 2-4). In contrast, the VizSnippets pipeline can create an image collage with four images and augment the snippet with keywords to expose the underlying topic (see Fig. 7-1). We see that keywords can be highly valuable in these cases where the bundle title is non-informative, and our design requires only a minimal increase of

Fig. 7. Snippet comparison: (1) VizSnippets pipeline, (2) Tableau Online, (3) Tableau Public, and (4) Qlik Sense.

the overall snippet size or a slightly smaller thumbnail. When a user hovers on the snippet, arrows are superimposed and they can scroll through additional image content.

5.4 Designer Feedback

We obtained feedback from 6 potential snippet designers within Tableau product and research groups on the final version of the VizSnippets pipeline during an informal one-hour session. They confirmed the issues related to existing snippet designs, were highly interested in the overall ability to show more relevant information to users, and saw promise in many specific aspects of the pipeline. In comparison to the baseline designs, the added keywords and the image selection and collage generation significantly enhance the information capacity of snippets. They valued the large preview capability to show more results without loading the fully-interactive version and the flexibility of generating different types of layouts for different tasks.

The interviewees stated that this work will inform new snippet designs and they could see using the underlying algorithms for future products, although they said they were unlikely to use every supported layout. In particular, they thought some multi-image layouts may be too visually busy when showing search results in a consumer product, although the information density might be much better. To address this specific issue, designers could include only a few images in one collage. This strategy can result in several collages which are then combined into an image carousel that lets users scroll through additional content without leaving the snippet. An alternative approach is to show only one image per snippet and replace it with other images when users point their cursor onto it, similar to interactive video thumbnails [67].

6 Visual Inspection Through Random Sampling

We now reflect on our approach to creating multiple lightweight visual inspectors, to cast it as a general method.

6.1 Inspectors for Iterative Development

Inspectors can support the common trial and error strategy for model or algorithm development where models and parameters are initially chosen through some combination of experience, reading related work, and ideally by running quantitative analyses with ground truth data. These algorithmic choices are then iteratively refined until results are satisfactory or improvement stagnates.

A concrete example is the detection of similar images. Using an algorithm based on the histogram of gradients, we manually assigned similarity labels to example image pairs, and then computed statistical measures of the performance of the binary classifier. Precision, recall, and other measures were useful instruments in determining model appropriateness. However, model results invariably led us to follow-up questions that were amenable to visual inspection. For example, in which cases does the model prediction diverge from the human annotator? Do those images share any special characteristics? These insights can in turn be used to tweak model parameters, such as the size of the cell for which histograms are created.

6.2 Inspector Characteristics

Inspectors can facilitate discussions and inquiry about needs with project stakeholders, and most centrally serve to internally probe and validate algorithmic choices and parameter settings. We identify four

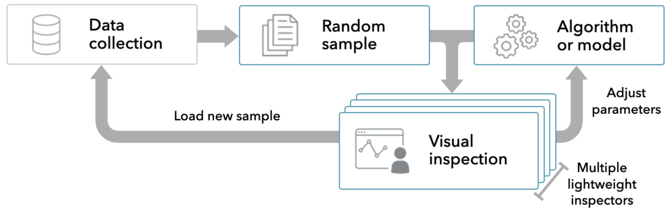

Fig. 8. A conceptual model for extensive, item-level visual inspection through random sampling.

abstract tasks that can be supported by inspectors: 1) see and compare results after applying a model or algorithm on real examples, 2) see how results change with varying thresholds, 3) see intermediate steps instead of the final result, and 4) see negative examples (e.g., false positives, false negatives).

The process, illustrated in Fig. 8, is driven by two key ideas. First, randomly load samples from a substantial data corpus for item-level inspection to avoid cherry-picking. Second, spin up many lightweight inspectors targeted at specific analysis tasks or algorithms, instead of building one feature-rich system.

All inspectors should have a button to load new samples, to help the analyst gradually progress from a partial to a more complete understanding of model or algorithm behavior, to judge its effectiveness and limitations. Random sampling guarantees that analysts scrutinize a diverse range of results. Inspectors may have control widgets, for example to adjust algorithm parameters or to constrain the samples to a specific data range. Visual inspection through random sampling actively mitigates the risk of confirmation bias, but does not provide immunity to it. Analysts may still be enticed to draw premature conclusions after seeing only a small set of examples.

The scope of each inspector should be narrowly constrained to support ultra rapid prototyping. These tools are a means to an end, and, preferably, the implementation should not take longer than a few hours or days. The design and functionality of the visual inspectors can vary substantially, but we aimed to reuse components in different inspectors and a common database. We implemented web-based inspectors in JavaScript and D3, with a Python back-end, to provide a maximum of flexibility and because we were familiar with the development stack. Ideally, inspectors provide nearly instantaneous feedback during the interaction. Thus, model predictions or algorithm output is either preprocessed on all available data or computed live on the current sample.

6.3 Visual Inspector Related Work

Numerous visual analytics approaches related to *model explanation* [17, 59], *model construction* [42,45], and *model evaluation* [2,15,32,71] have been suggested. Our concern here is not to illuminate the model internals [66] or steer the generation of new models [11]. Instead, we are using visual methods for the evaluation and refinement of computational models that range from simple algorithms to more complex machine learning models. The goal is insights on effectiveness and limitations that can inform new parameter and model choices, or data iteration [27]. Others have noted that the creation of feature-rich systems tailored to specific models [36] is not always feasible, particularly when the goal is to apply existing models to new domains.

We suggest the extensive visual inspection with lightweight tools and real-world data as an alternative approach. Similar to the visual parameter space analysis [54], our approach also depends on a sampling mechanism, but we randomly sample data for item-level inspection to see a model in action while avoiding confirmation bias, instead of systematically sampling the parameter space. Closely related is the visual diagnosis of binary classifiers by Krause [32] that includes presenting model results based on single instances. However, their focus is on instance-level explanations for binary classifications while we discuss a high-level visual-inspection strategy that is model agnostic.

Kachkaev et al. [29] provide a single example showing the value of visual inspection with real-world data, in their case survey results; we advocate injecting such inspectors at many places within a computational system to assess performance and set parameters.

7 Discussion and Future Work

We created and validated the computational pipeline through quantitative and qualitative analysis of two visualization collections, providing substantial evidence that we achieved our goal of dramatically increased information density within snippets. Follow-up benchmarks on additional collections would strengthen our claims of generality. User studies focused on specific use cases, ideally informed by the nuances of specific visualization platform characteristics and requirements, would shed further light on efficacy and would help designers select suitable page and snippet layouts. Our findings are applicable to other computational notebooks that may contain substantial visual content, such as *Jupyter Notebook* [49], and can complement code summarization techniques [65].

We chose *extractive* techniques to summarize the content of bundles verbatim. In contrast, *abstractive* techniques, such as latent Dirichlet allocation (LDA) [6], derive topics from a corpus and then use topic keywords to describe a document. These high-level concepts may be more understandable to users but the models need to be trained on large corpora of text, and even then, some bundles will get falsely allocated to a topic and represented by irrelevant keywords. Approaches related to natural language generation [46] also rely on massive amounts of training data and are not directly applicable to visualization snippets due to the limited text fragments. For example, Chen [12] generated abstractive summaries of web pages but ignored those with less than 100 words to ensure a sufficient basis for summarization. In comparison, our sample Tableau workbooks contain 37 unique words on average.

In terms of visual content, all methods are based on a set of screenshots, with the aim to accurately represent the visual style, but views and dashboards are scaled down significantly and may become illegible. One option is to redraw smaller versions of charts, which poses its own research problem, because of exotic chart types and inaccessible data and chart specifications. Another alternative is to analyze the bundle content and summarize it with abstract iconography or graphics [30].

8 Conclusion

We present a computational pipeline to compress visualization bundles, such as Tableau workbooks or Observable notebooks, into representative snippets. These snippets allow users to judge the relevance of bundles for a specific task without opening and inspecting them one by one. To gain a better understanding of requirements and challenges, we engaged with Tableau teams of potential snippet designers aware of end-user pain points, and reviewed existing snippets across a broad set of five different tools. We found that existing designs suffered from poor information density, substantially impairing the ability of users to make the expected relevance judgements. Snippets have an immense impact on the consumption of visualization collections, and designing them with higher information content can significantly improve the workflow of use cases including the presentation of search results, navigation through faceted browsing, and recommendation suggestions. The presented VizSnippets computational pipeline is responsive to a specified pixel budget, and can be adapted to many use cases and form factors through a suite of controllable layout families. The added keywords and the image selection and collage generation significantly enhance the information capacity. While the effectiveness of our pipeline was primarily validated through Tableau and Observable bundles, we argue that the suggested compression techniques can be applied to many other visualization collections due to their similar characteristics. The development and evaluation of the pipeline was informed through extensive visual inspection on random samples of real-world data. We reflect on this method and provide guidance for other researchers.

Acknowledgments

We thank our collaborators at Tableau and appreciate feedback from Madison Elliot, Steve Kasica, Zipeng Liu, Ben Shneiderman, and Mara Solen.

REFERENCES

[1] R. A. Al-Zaidy, S. R. Choudhury, and C. L. Giles. Automatic summary generation for scientific data charts. In *Proc. AAAI Conf. on Artificial Intelligence*, pp. 658–663, 2016.

[2] S. Amershi, M. Chickering, S. M. Drucker, B. Lee, P. Simard, and J. Suh. ModelTracker: Redesigning performance analysis tools for machine learning. In *Proc. ACM SIGCHI Conf. on Human Factors in Computing Systems (CHI)*, pp. 337–346, 2015.

[3] Z. Armstrong. Covid-19 viz roundup (Observable notebook). https://observablehq.com/@zanarmstrong/covid-19-viz-roundup, 2021. Accessed: 2021-02-26.

[4] D. Auber. Using strahler numbers for real time visual exploration of huge graphs. In *Int. Conf. Computer Vision and Graphics*, vol. 1, p. 3, 2002.

[5] A. Aula, R. M. Khan, Z. Guan, P. Fontes, and P. Hong. A comparison of visual and textual page previews in judging the helpfulness of web pages. In *Proc. Int. Conf. on World Wide Web*, pp. 51–60, 2010.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[7] M. Brehmer, B. Lee, P. Isenberg, and E. K. Choe. A comparative evaluation of animation and small multiples for trend visualization on mobile phones. *IEEE Trans. Visualization and Computer Graphics*, 26(1):364–374, 2019.

[8] Z. Bylinskii, S. Alsheikh, S. Madan, A. Recasens, K. Zhong, H. Pfister, F. Durand, and A. Oliva. Understanding infographics through textual and visual tag prediction. *arXiv preprint arXiv:1709.09215*, 2017.

[9] R. Capra, J. Arguello, and F. Scholer. Augmenting web search surrogates with images. In *Proc. ACM Int. Conf. Information and Knowledge Management*, pp. 399–408, 2013.

[10] J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proc. ACM SIGIR Conf. Research and Development in Information Retrieval*, pp. 335–336, 1998.

[11] D. Cashman, S. R. Humayoun, F. Heimerl, K. Park, S. Das, J. Thompson, B. Saket, A. Mosca, J. Stasko, A. Endert, et al. A user-based visual analytics workflow for exploratory model analysis. 38(3):185–199, 2019.

[12] W.-F. Chen, S. Syed, B. Stein, M. Hagen, and M. Potthast. Abstractive snippet generation. In *Proc. The Web Conference*, pp. 1309–1319, 2020.

[13] Y. Chen. Visualizing large time-series data on very small screens. In *Proc. Eurographics Conf. Visualization (EuroVis)*, pp. 37–41, 2017.

[14] Z. Chen, M. Cafarella, and E. Adar. Diagramflyer: A search engine for data-driven diagrams. In *Proc. Int. Conf. on World Wide Web*, pp. 183–186, 2015.

[15] I. K. Choi, N. K. Raveendranath, J. Westerfield, and K. Reda. Visual (dis) confirmation: Validating models and hypotheses with visualizations. In *Int. Conf. Information Visualization*, pp. 116–121, 2019.

[16] K. Church, B. Smyth, and M. T. Keane. Evaluating interfaces for intelligent mobile search. In *Proc. Workshop on Web Accessibility (W4A): Building the Mobile Web: Rediscovering Accessibility?*, pp. 69–78, 2006.

[17] D. Collaris and J. J. van Wijk. ExplainExplore: visual exploration of machine learning explanations. In *Proc. IEEE Pacific Visualization Symp.*, pp. 26–35, 2020.

[18] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, pp. 886–893, 2005.

[19] R. DeLine, M. Czerwinski, B. Meyers, G. Venolia, S. Drucker, and G. Robertson. Code thumbnails: Using spatial memory to navigate source code. In *Proc. IEEE Symp. on Visual Languages and Human-Centric Computing*, pp. 11–18, 2006.

[20] S. Dziadosz and R. Chandrasekar. Do thumbnail previews help users make better relevance decisions about web search results? In *Proc. ACM SIGIR Conf. Research and Development in Information Retrieval*, pp. 365–366, 2002.

[21] B. Erol, K. Berkner, and S. Joshi. Multimedia thumbnails for documents. In *Proc. Int. Conf. on Multimedia*, pp. 231–240, 2006.

[22] K. N. Fachry, J. Kamps, J. Zhang, et al. The impact of summaries: What makes a user click. In *Proc. Dutch-Belgian Information Retrieval Workshop*, pp. 47–54, 2010.

[23] S. Greene, G. Marchionini, C. Plaisant, and B. Shneiderman. Previews and overviews in digital libraries: Designing surrogates to support visual information seeking. *Journal of the American Society for Information Science*, 51(4):380–393, 2000.

[24] M. Gygli, Y. Song, and L. Cao. Video2Gif: Automatic generation of animated gifs from video. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1001–1009, 2016.

[25] J. Heer, J. Mackinlay, C. Stolte, and M. Agrawala. Graphical histories for visualization: Supporting analysis, communication, and evaluation. *IEEE Trans. Visualization and Computer Graphics*, 14(6):1189–1196, 2008.

[26] J. Hoffswell, W. Li, and Z. Liu. Techniques for flexible responsive visualization design. In *Proc. ACM SIGCHI Conf. on Human Factors in Computing Systems (CHI)*, pp. 1–13, 2020.

[27] F. Hohman, K. Wongsuphasawat, M. B. Kery, and K. Patel. Understanding and visualizing data iteration in machine learning. In *Proc. ACM SIGCHI Conf. on Human Factors in Computing Systems (CHI)*, pp. 1–13, 2020.

[28] S. Kaasten, S. Greenberg, and C. Edwards. How people recognise previously seen web pages from titles, urls and thumbnails. In *People and Computers XVI-Memorable Yet Invisible*, pp. 247–265. Springer, 2002.

[29] A. Kachkaev, J. Wood, and J. Dykes. Glyphs for exploring crowd-sourced subjective survey classification. 33(3):311–320, 2014.

[30] H. Kim, J. Oh, Y. Han, S. Ko, M. Brehmer, and B. C. Kwon. Thumbnails for data stories: A survey of current practices. In *IEEE Visualization Conference (VIS)*, pp. 116–120, 2019.

[31] D. E. Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, 1984.

[32] J. Krause, A. Dasgupta, J. Swartz, Y. Aphinyanaphongs, and E. Bertini. A workflow for visual diagnostics of binary classifiers using instance-level explanations. In *Proc. IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 162–172, 2017.

[33] H. Lam and P. Baudisch. Summary thumbnails: readable overviews for small screen web browsers. In *Proc. ACM SIGCHI Conf. on Human Factors in Computing Systems (CHI)*, pp. 681–690, 2005.

[34] Z. Li, S. Shi, and L. Zhang. Improving relevance judgment of web search results with image excerpts. In *Proc. Int. Conf. World Wide Web*, pp. 21–30, 2008.

[35] C. Liu, L. Xie, Y. Han, X. Yuan, et al. Autocaption: An approach to generate natural language description from visualization automatically. In *Proc. IEEE Pacific Visualization Symp.*, pp. 191–195. IEEE, 2020.

[36] S. Liu, X. Wang, M. Liu, and J. Zhu. Towards better analysis of machine learning models: A visual analytics perspective. *Visual Informatics*, 1(1):48–56, 2017.

[37] Z. Liu, S. H. Zhan, and T. Munzner. Aggregated dendrograms for visual comparison between many phylogenetic trees. *IEEE Trans. Visualization and Computer Graphics*, 26(9):2732–2747, 2019.

[38] S. Madan, Z. Bylinskii, M. Tancik, A. Recasens, K. Zhong, S. Alsheikh, H. Pfister, A. Oliva, and F. Durand. Synthetically trained icon proposals for parsing and summarizing infographics. *arXiv preprint arXiv:1807.10441*, 2018.

[39] M.-C. Marcos, F. Gavin, and I. Arapakis. Effect of snippets on user experience in web search. In *Proc. Int. Conf. on Human Computer Interaction*, pp. 1–8, 2015.

[40] D. McKay, G. Buchanan, N. Vanderschantz, C. Timpany, S. J. Cunningham, and A. Hinze. Judging a book by its cover: interface elements that affect reader selection of ebooks. In *Proc. Australian Computer-Human Interaction Conference*, pp. 381–390, 2012.

[41] D. Merejkowsky and R. Kelly. PyEnchant spellchecking library. https://pypi.org/project/pyenchant/.

[42] Y. Ming, P. Xu, F. Cheng, H. Qu, and L. Ren. ProtoSteer: Steering deep sequence model with prototypes. *IEEE Trans. Visualization and Computer Graphics*, 26(1):238–248, 2019.

[43] V. O. Mittal, J. D. Moore, G. Carenini, and S. Roth. Describing complex charts in natural language: A caption generation system. *Computational Linguistics*, 24(3):431–467, 1998.

[44] Y. Mu, Q. Wei, G. Chen, and X. Guo. An iterative multi-criteria optimization of product snippets enhanced by feature extraction from online reviews. In *Proc. Conf. Data Science and Knowledge Engineering for Sensing Decision Support*, pp. 545–552, 2018.

[45] T. Mühlbacher, H. Piringer, S. Gratzl, M. Sedlmair, and M. Streit. Opening the black box: Strategies for increased user involvement in existing algorithm implementations. *IEEE Trans. Visualization and Computer Graphics*, 20(12):1643–1652, 2014.

[46] R. Nallapati, B. Zhou, C. dos Santos, G. Çağlar, and B. Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proc. SIGNLL Conf. on Comp. Natural Language Learning*, pp. 280–290, 2016.

[47] C. G. Northcutt, K. A. Leon, and N. Chen. Comment ranking diversification in forum discussions. In *Proceedings of the Fourth (2017) ACM Conference on Learning@ Scale*, pp. 327–330, 2017.

[48] M. Oppermann, R. Kincaid, and T. Munzner. VizCommender: Computing text-based similarity in visualization repositories for content-based recommendations. *IEEE Trans. Visualization and Computer Graphics*, 27(2):495–505, 2021.

[49] Project Jupyter. Jupyter notebooks. https://jupyter.org, 2021.

[50] I. Renz, A. Ficzay, and H. Hitzler. Keyword extraction for text characterization. *Natural language processing and information systems*, 2003.

[51] S. Robertson. Understanding inverse document frequency: On theoretical arguments for IDF. *Journal of Documentation*, 2004.

[52] J. Rogers, A. H. Patton, L. Harmon, A. Lex, and M. Meyer. Insights from experiments with rigor in an EvoBio design study. *IEEE Trans. Visualization and Computer Graphics*, 2020.

[53] A. Sarikaya, M. Correll, L. Bartram, M. Tory, and D. Fisher. What Do We Talk About When We Talk About Dashboards? *IEEE Trans. Visualization and Computer Graphics*, 25(1):682–692, 2019. doi: 10.1109/TVCG.2018.2864903

[54] M. Sedlmair, C. Heinzl, S. Bruckner, H. Piringer, and T. Möller. Visual parameter space analysis: A conceptual framework. *IEEE Trans. Visualization and Computer Graphics*, 20(12):2161–2170, 2014.

[55] Y. Song, M. Redi, J. Vallmitjana, and A. Jaimes. To click or not to click: Automatic selection of beautiful thumbnails from videos. In *Proc. ACM Int. Conf. Information and Knowledge Management*, pp. 659–668, 2016.

[56] E. Spyromitros-Xioufis, S. Papadopoulos, A. L. Ginsca, A. Popescu, Y. Kompatsiaris, and I. Vlahavas. Improving diversity in image search via supervised relevance scoring. In *Proc. ACM Int. Conf. Multimedia Retrieval*, pp. 323–330, 2015.

[57] D. Srivastava, R. Wadhvani, and M. Gyanchandani. A review: color feature extraction methods for content based image retrieval. *Int. Journal Computational Engineering & Management*, 18(3):9–13, 2015.

[58] A. Stoffel, H. Strobelt, O. Deussen, and D. A. Keim. Document thumbnails with variable text scaling. *Computer Graphics Forum*, 31(3pt3):1165–1173, 2012.

[59] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush. LSTMVis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Trans. Visualization and Computer Graphics*, 24(1):667–676, 2017.

[60] H. Strobelt, D. Oelke, C. Rohrdantz, A. Stoffel, D. A. Keim, and O. Deussen. Document cards: A top trumps visualization for documents. *IEEE Trans. Visualization and Computer Graphics*, 15(6):1145–1152, 2009.

[61] J. Teevan, E. Cutrell, D. Fisher, S. M. Drucker, G. Ramos, P. André, and C. Hu. Visual snippets: summarizing web pages for search and revisitation. In *Proc. ACM SIGCHI Conf. on Human Factors in Computing Systems (CHI)*, pp. 2023–2032, 2009.

[62] N. Tintarev and J. Masthoff. A survey of explanations in recommender systems. In *IEEE Int. Conf. on Data Engineering*, pp. 801–810, 2007.

[63] R. R. Varior, M. Haloi, and G. Wang. Gated siamese convolutional neural network architecture for human re-identification. In *European Conf. on Computer Vision*, pp. 791–808, 2016.

[64] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Processing*, 13(4):600–612, 2004.

[65] J. Wenskovitch, J. Zhao, S. Carter, M. Cooper, and C. North. Albireo: An interactive tool for visually summarizing computational notebook structure. In *Proc. IEEE Visualization in Data Science (VDS)*, pp. 1–10, 2019.

[66] J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Viégas, and J. Wilson. The What-If Tool: Interactive probing of machine learning models. *IEEE Trans. Visualization and Computer Graphics*, 26(1):56–65, 2019.

[67] B. M. Wildemuth, G. Marchionini, X. Fu, J. S. Oh, and M. Yang. The usefulness of multimedia surrogates for making relevance judgments about digital video objects. *Information Processing & Management*, 56(6):102091, 2019.

[68] B. M. Wildemuth, G. Marchionini, T. Wilkens, M. Yang, G. Geisler, B. Fowler, A. Hughes, and X. Mu. Alternative surrogates for video objects in a digital library: users' perspectives on their relative usability. In *Int. Conf. on Theory and Practice of Digital Libraries*, pp. 493–507, 2002.

[69] A. Wu, W. Tong, T. Dwyer, B. Lee, P. Isenberg, and H. Qu. MobileVisFixer: Tailoring web visualizations for mobile phones leveraging an explainable reinforcement learning framework. *IEEE Trans. Visualization and Computer Graphics*, 2020.

[70] V. Yoghourdjian, T. Dwyer, K. Klein, K. Marriott, and M. Wybrow. Graph thumbnails: Identifying and comparing multiple graphs at a glance. *IEEE Trans. Visualization and Computer Graphics*, 24(12):3081–3095, 2018.

[71] J. Zhang, Y. Wang, P. Molino, L. Li, and D. S. Ebert. Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models. *IEEE Trans. Visualization and Computer Graphics*, 25(1):364–373, 2018.

[72] T. Zhang and R. Samadani. Automatic generation of music thumbnails. In *Proc. Int. Conf. on Multimedia and Expo*, pp. 228–231, 2007.