Automatic Annotation Synchronizing with Textual Description for Visualization

Chufan Lai[1], **Zhixian Lin**[1], **Ruike Jiang**[1], **Yun Han**[1], **Can Liu**[1], **Xiaoru Yuan**[1,2*]

[1]Key Laboratory of Machine Perception (Ministry of Education), Peking University
[2]National Engineering Laboratory for Big Data Analysis and Application, Peking University
{chufan.lai, zhixian.lin, jiangrk, yunhan, can.liu, xiaoru.yuan}@pku.edu.cn

ABSTRACT

In this paper, we propose a technique for automatically annotating visualizations according to the textual description. In our approach, visual elements in the target visualization, along with their visual properties, are identified and extracted with a Mask R-CNN model. Meanwhile, the description is parsed to generate visual search requests. Based on the identification results and search requests, each descriptive sentence is displayed beside the described focal areas as annotations. Different sentences are presented in various scenes of the generated animation to promote a vivid step-by-step presentation. With a user-customized style, the animation can guide the audience's attention via proper highlighting such as emphasizing specific features or isolating part of the data. We demonstrate the utility and usability of our method through a user study with various use cases.

Author Keywords

Visualization; Annotation; Natural Language Interface; Machine Learning.

CCS Concepts

•**Human-centered computing** → **Visualization toolkits;**

INTRODUCTION

Visualization plays an important role when people share their data findings. The visual elements with their properties can reflect various kinds of data features in an intuitive and appealing way. However, without proper guidance, it could be challenging for the audience to follow the (oral or textual) description of a visualization. The audience needs to comprehend the description, knowing which entities and properties have been mentioned, and then visually search for them in the image. As the visualization becomes more complicated, this process could be tedious and time-consuming. Due to limited short-term memory, the audience must frequently switch between the description and the image, focusing on only a few entities at each time.

A simple way to facilitate chart reading is to enhance the visualization with proper highlighting and annotations. An annotation is, as defined by Tamara Munzner [44], *"the addition of graphical or textual elements associated with one or more preexisting visualization elements"*. In this paper, we narrow down this definition to specifically refer to *additional textual elements*. We call the graphical ones as *graphical overlays* [35]. By providing annotations, the presenter can effectively guide the audience's attention to the image area he/she is describing [38]. It spares the audience the time for visual search and thus increases the communication efficiency.

Despite the benefits, crafting a well-annotated visualization can be challenging. Knowing which entities to emphasize, the presenter has to: (1) highlight the focal areas properly [33], (2) decide and draw the right kind of graphical overlays (for example, arrows and trend lines), and (3) find an appropriate place for the annotation so that it does not obscure critical image contents. Moreover, a complete data story often involves more than one focus, each described in multiple sentences. Annotating all of them on the same image requires a proper placement [12, 61, 3]. Using animation could be a better way if condition permits (e.g., multi-media is available), where each scene of the animation has only a few focuses. But that requires a fluid animation to smoothly transfer the audience's attention, which requires time and expertise to make.

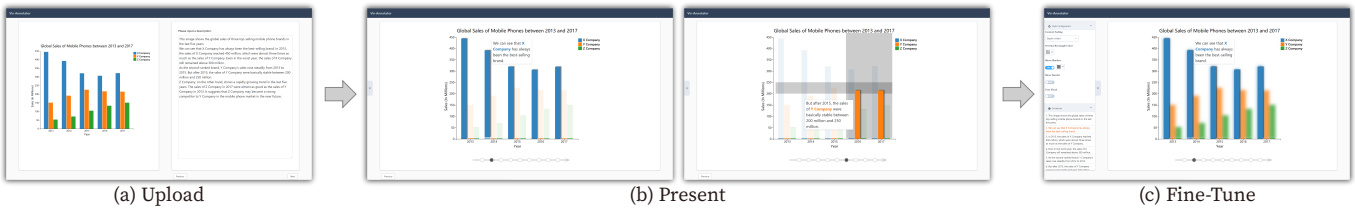
Regarding this challenge, there are little solutions in the visualization community. The widely used libraries including D3 [5] and Vega-Lite [51], do not have specialized modules for annotating. Neither does commercial software like Tableau or Excel. Academic research focuses more on generating the so-called *observational annotations* [32, 29, 6], aiming to mark out observable data features in the visualization. Research on *additive annotations* [34, 48, 62] helps users communicate with free-form annotations. However, such annotations must be crafted manually since the information is not included in the data or visualization. Besides, the existing tools do not provide assistance for making animations with annotated visualizations.

In this paper, we propose an automatic annotating technique to help users efficiently carry out their presentations with visualization charts. The user can upload a visualization image

*indicates the corresponding author.

CHI '20, April 25–30, 2020, Honolulu, HI, USA.

ACM ISBN 978-1-4503-6708-0/20/04 ...$15.00.
http://dx.doi.org/10.1145/3313831.3376443



(a) Upload (b) Present (c) Fine-Tune

Figure 1. The interface and interaction flow of Vis-Annotator. (a) The user uploads the visualization image with its description. (b) With the generated animation, the user starts the presentation by stitching between different scenes. He/She can return to the upload stage in case of updates or (c) customize the rendering effects at any time.

with its textual description and get a well-annotated animation in the blink of an eye (Fig. 1). To that end, we adopt the state-of-the-art *Object Detection (OD)* technique [18] to identify various kinds of visual entities in the visualization, as well as their visual properties. Texts in the chart are extracted using *Optical Character Recognition (OCR)*. We parse the textual description using *Natural Language Processing (NLP)* engines [23] to generate search requests for entities that have been mentioned. By fulfilling these requests, we can anchor each sentence to the corresponding focal image areas, which allows us to perform the subsequent annotating. At last, we develop a prototype tool called *Vis-Annotator* to help customize the annotations and generate a fluid animation for the purpose of presentation. User interviews show that the generated annotations can effectively promote data-relevant presentation.

RELATED WORK

In this section, we briefly review the literature concerning four relevant topics.

Annotation Assisted Story-telling

Visualization plays an essential role in telling stories behind the data. Segel and Heer [53] refer to this kind of visualization as *narrative visualization*, and provide a comprehensive review on its design space. They discover that the change of scenes [58] used in visual media (e.g., comics and films) is also applicable to visualization [20] to orient the viewer's attention. Meanwhile, highlighting and annotations can be used for pointing out key observations from the data or adding other intended message [28]. Kosara and Mackinlay [36] emphasize the importance of annotations for making the visualization a part of the story, which is evidenced by experiment results in live presentations [9]. Based on that, Lee et al. [39] state the need for supportive tools for efficiently creating, customizing and manipulating annotations.

Hullman and Diakopoulos [28] categorize annotations into three types: textual, graphical, and social. But Nicholas and Maneesh [35] exclude the graphical overlays and regard annotations as pure textual information. For each type of annotation, there do exist supportive tools in the visualization community. Many of them [8, 7, 50, 48] focus on assisting the manual or semi-automatic creation of presentation-style annotations. Some others [22, 30, 42, 62] also facilitate the use of social-style annotations in collaborative sense-making. However, all of them demand significant user effort to create annotations.

Automatically Generating Annotations

In addition to interactive tools, there do exist automation techniques for annotating. Hullman et al. [29] categorize annotations into two types: observational and additive, based on the information source. Bryan et al. [6] redefine the categories as *data-driven* and *free-form* annotations.

The automation of observational/data-driven annotations is done by extracting features from the visualization. Kong and Agrawala [35] detect perceptual features in a chart and add graphical overlays to enhance its perception. Kandogan [32] introduces an automatic technique for annotating data features (e.g., clusters and outliers) in point-based charts.

More close to our work are the techniques that automate additive annotating. Hullman et al. [29] propose to generate annotated stock charts based on news articles. Similarly, Gao et al. [15] generate annotated maps from articles. But these two are not designed for presentation scenarios. Temporal Summary Images (TSIs) [6] yields both additive and data-driven annotations for illustrative graphics. However, the additive ones must be manually created. Besides, TSIs requires the underlying data, which may not be accessible in many cases.

Unlike the above techniques, the automation we proposed is not limited to any type of annotation or visualization. In fact, our framework for matching the description with visual entities can even be applied to annotate natural images.

Extracting Information from Visualizations

Since we do not assume the underlying data to be accessible, we need to extract information from the visualization to understand the description. This procedure is also known as *reverse engineering of visualizations*. We divide the relevant literature into two parts, based on the type of the extracted entities.

Data Entity Extraction: Zhou and Tan [63] propose a boundary tracing technique based on Hough transformation to detect bars in a bar chart. Huang et al. [27, 25, 26] use edge detection techniques to extract data entities from bar, pie, and line charts. More recently, ReVision [52] adopts image processing techniques to identify the graphical marks and extract the underlying data. Choudhury et al. [10, 11] present algorithms to extract lines from line charts. Jung et al. [31] adopt a semi-automatic strategy for data extraction, while Méndez et al. [43] achieve the same goal in a more interactive way.

Auxiliary Entity Extraction: Huang and Tan [26] infer the role of texts in a chart by their spatial relationships. Figure-

Seer [55] also detects axes and legends by their alignments. Poco et al. [45] classify the roles of texts using Convolutional Neural Networks (CNNs) and then extract the textual information using different approaches.

In previous works, data entities (e.g., bars, lines, etc.) and auxiliary entities (e.g., axes, legends, etc.) are detected in different ways. Moreover, different data/auxiliary entities also require specialized handling, even though they only differ in their appearances. In our framework, all kinds of entities can be detected with the same object detection model. It remarkably reduces the complexity of the detection while maintaining the detection rate.

Natural Language Interface

Nowadays, natural language is a critical way for human-computer interaction. What we aim to achieve is to build up a *Natural Language Interface (NLI)* for annotating. Much research in the visualization community aims at a similar goal.

According to Srinivasan and Heer [56], NLIs for visualizations commonly serve two purposes: generating visualizations and triggering interactions with an existing visualization. In the former case, the system identifies high-level utterances such as data-relevant questions [2] and statements [14, 37] to understand user's demands for specific visualization or infographics [13]. Since the language is parsed into data queries, this kind of system usually works with a database.

NLIs for interactions, on the other hand, handles more specific, low-level commands for operating the visualization. They commonly serve for visual analytic systems [54, 24, 57] equipped with a rich set of interactions. Some of them provide highlighting functions similar to our work.

In a sense, the NLI we built serves both purposes, but for static charts rather than interactive visualizations. What we create is not the visualization itself, but auxiliary effects (highlighting and annotating) that often appear as operations in an interactive system. Nevertheless, the NLP techniques adopted here are close to those in the above works.

AUTOMATION OF VIS ANNOTATION

In this section, we introduce the design details of the proposed technique based on its workflow (Fig. 2) that contains three major modules. Since the data serially goes through these modules, our elaboration also follows the same order. Before that, we briefly introduce the workflow and its design rationale.

The Workflow

Our design of the workflow is inspired by the four-step process of manual annotating. First, we browse the visualization to understand what entities exist in the image and what are their properties. Then we read the textual description to comprehend what kind of entities have been described. After that, we perform the visual searching task, i.e. search for the described entities in the set of view side. At last, we highlight the entities in the image and display the sentence aside as the annotation.

The rendering is easy once the visual searching is done. Therefore, we combine the last two tasks and design three modules: Object Detection (OD), Natural Language Processing (NLP), and Annotation. The former two modules process the uploaded image and texts, respectively, to finish the former two tasks, while the Annotation module does the remaining.

Object Detection: In this module, we use Mask-RCNN [18] for to detect visual entities in the image. Since existing models cannot be applied to visualization charts, we need to train the model from scratch. The training data is a chart corpus we collected from the web and manually labeled. After training, the model can identify various kinds of visual entities and output their rough contours, which are further processed to get the fine contours. For each entity, we perform computer vision algorithms to capture visual properties and visible texts (e.g., labels) inside its contour. All detected results are sent to the annotation module, waiting to be queried. Visible texts are sent to the NLP module to help interpret the description.

Natural Language Processing: First, low-level NLP algorithms are performed to partition the texts and extract the words' part-of-speech (POS) and dependencies. Then we identify keywords for describing the chart contents based on a keyword vocabulary and the texts received from OD. A structure library is used to understand the decorating relationships between keywords in order to generate the final queries.

Annotation: This module receives both the visual entities in the chart and the described entities (as queries) in the description. Then we fulfill the queries via an automatic visual search. Entities described by visible texts can be found directly due to their presence in both the image and the description. Entities described by visual properties are found by cross-filtering based on predefined matching rules. Once an entity is found, the source sentence will be anchored to its contour. At the rendering stage, if a sentence has at least one focal entity, it will be shown aside its highlighted focuses. Sentences with no matched entities are regarded as contexts and are shown aside the whole image without any highlights. Different sentences are displayed in different scenes of a step-by-step animation, in order to avoid occlusion and promote a fluid presentation. Styles of the annotations can be customized via interactions.

Describing a Visualization

Before diving into more details, we first discuss how people usually describe a visualization. It reveals how we can reverse this process and find the described contents from vision, which is also the design rationale for the low-level sub-modules.

Based on their functionalities, entities in a visualization can be categorized into two types, namely the data and auxiliary entities. The ways people describe them also differ, which is why we handle their detection differently.

Data entities are visual elements that are used to encode the actual data, like the bars in a bar chart, or the points in a scatterplot. When referring to a data entity in natural language, the presenter tends to use its ID, which is often attached to it visually. If some text is attached to a data entity, we call it a "data label". In addition to IDs, data labels can also show numeric values [32, 6].

When the visible ID is absent, the presenter may refer to a data entity by describing its visual properties, including its shape,

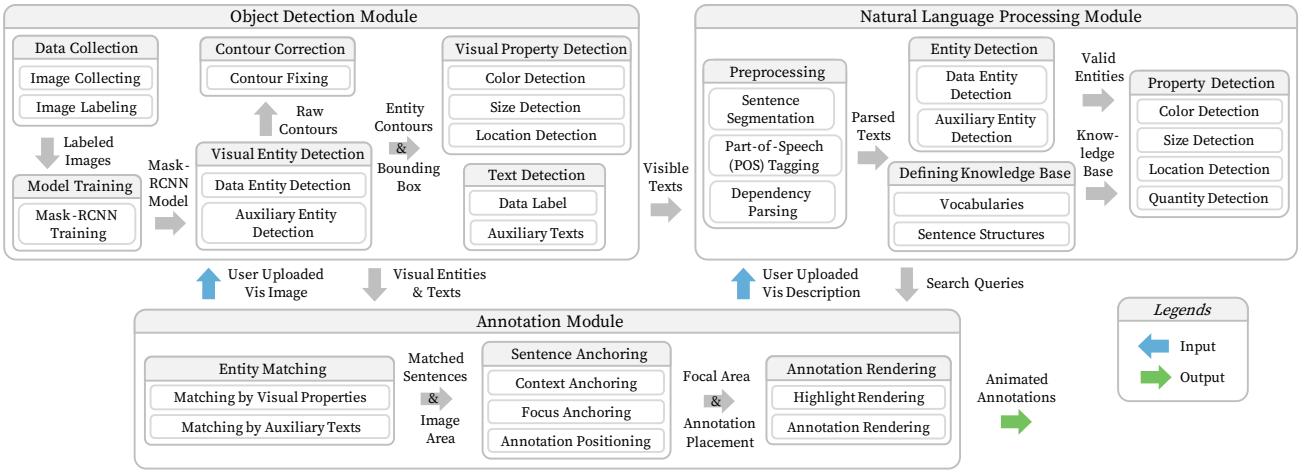


Figure 2. The workflow consists of three major modules: Object Detection (OD), Natural Language Processing (NLP), and Annotation. The OD module identifies visual entities and visible texts in the chart, along with their visual properties. The NLP module parses the description to generate queries for the described entities. The Annotation module fulfills the queries to anchor each sentence to the corresponding image areas. It also renders annotations and highlights and generates the final animation.

color, size, etc. It is the best way to help the audience quickly locate it in the image. When described by properties, the data entity often appears in the description as a noun carrying modifiers. Two types of visual properties can be used as the noun: shapes (e.g., "the red circle") and data features (e.g., "the large cluster"). Note that in the context of visualization, a data feature (like a cluster/outlier) is merely a combination of basic visual properties that are more familiar to experienced analysts than ordinary people.

Data entities alone are not enough for presenting the data. In most cases, the auxiliary information is essential for interpretation. It conveys the visual mapping or the context of the data. Visual elements carrying auxiliary information are called ***auxiliary entities***. Examples include axes, titles, legends, etc. Data labels also fall into this category. When referring to an auxiliary entity in natural language, the presenter tends to use its type name (e.g., "the Y-axis"), which is well-known to all people that have been trained for chart-reading. But if there is a visible label (like the axis title), it is more preferred.

To understand the data-relevant description, we need to know what kind of data entities are shown in the visualization, and which of them has been mentioned. We also need auxiliary information to understand some unspoken messages (e.g., visual mappings). In most cases, the presenter will not describe them if they can be read from auxiliary entities in the image (e.g., "China" refers to red entities). Therefore, the task of OD is to identify data entities, extract their visual properties, and output the auxiliary information. Given the auxiliary information, the NLP module can decode the unspoken rules and translate the description into pure visual queries.

Object Detection

Object detection [17, 49, 18] is a well-developed research topic in the computer vision (CV) domain. The task is to identify various kinds of objects in a natural image and outline their profiles, as long as objects in the same category have similar visual appearances. For the OD module in our framework, the task is similar but more specialized for visualization images.

Data Entity Detection

Shapes and data features are candidates of data entities since they are the noun-type keywords in the description.

Without loss of generality, we currently only consider three types of shapes: rectangles, circles, and sectors. They correspond to data entities in the three most commonly used charts: bar charts, bi-variate scatterplots, and pie charts. Appearances of shapes are rather stable, allowing any machine learning model to effectively converge.

But data features seldom have predictable appearances. Even the most common kind, a cluster, can diverse greatly in different visualizations (e.g., scatterplots vs. parallel coordinates). It makes them unsuitable for appearance-based models. In addition, many mature techniques already exist for data feature detection. Therefore, we only focus on shape detection in this paper and leave feature detection as our future work.

Auxiliary Entity Detection

Four types of auxiliary entities are commonly seen in a visualization: axes, titles, legends, and data labels. Title as a brief introduction of the chart doesn't convey visual mappings and seldom needs to be emphasized. Therefore, we only consider the detection of the other three. As we shall see, not all auxiliary entities are suitable for appearance-based models.

Axes convey visual mappings concerning locations. A complete axis should contain at least two elements: the direction and the tick values. Locations of data entities are compared with the ticks along the axis direction to illustrate certain relationships. Usually, an axis also has a title and a unit to show the category and measurement of the tick values.

Legends convey visual mappings other than locations, including color, size, texture, etc. A complete legend should contain: (1) an exemplary element with some specific visual property

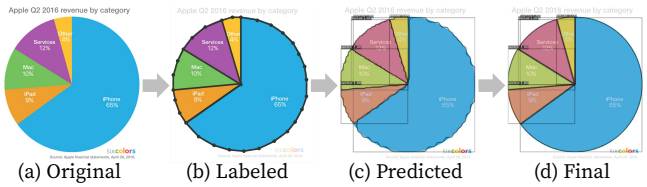


Figure 3. The workflow of visual entity detection. We collect (a) a chart corpus from the web and (b) label them for model training. The trained model identifies visual entities and returns (c) their bounding boxes and rough contours, which are (d) further processed to get the fine contours.

and (2) a legend name. Different types of legends have different appearances. Even for the most frequently used type, i.e., color legends, the quantitative and categorical ones appear differently and may be identified as different categories by the model. Poco et al. [46] have already addressed the extraction of color mappings for both quantitative and categorical legends. For simplicity, we only detect categorical color legends at the current stage. The detection can be easily extended to other legends by increasing new appearance categories.

As mentioned, data labels are texts attached to the data entities. As pure texts, labels do not have predictable forms. It is meaningless to build an appearance category for them. Additionally, close and distant labels appear differently. The latter usually have leader lines while the former don't.

Mask-RCNN for Visual Entity Detection

As stated in section "Extracting Information from Visualizations", there do exist simple detection techniques for both data and auxiliary entities. However, most of them are highly specialized and need to be redesigned for new types of visualizations or entities. Besides, none of them can handle data and auxiliary entities in the same framework.

In this approach, we seek to build a more concise and extensible framework that can not only integrate the detection of both entities but also extend to more visualization/entity types. Mask-RCNN [18], the state-of-the-art deep learning network for object detection, is a perfect candidate. Supported by Faster-RCNN [49], a highly efficient region proposal network, a Mask-RCNN model can identify the category of each object and outline its bounding box and contour in real-time. In addition to being efficient and accurate, Mask-RCNN can detect any type of entities, as long as they have predictable appearances. Moreover, the model is capable of recognizing occluded objects, which makes it suitable for crowded scenarios in visualizations.

Corpus Collecting: At the data collection stage, we consider three types of visualizations: bar charts, pie charts, and bivariate scatterplots. We examined some chart corpora used in previous works [60, 52, 31], but many of the images have poor qualities. Therefore, we decide to build our own chart corpus, following the standard data collecting procedures [45].

Our corpus is randomly collected from the web. Specifically, all images are collected from search results returned by the Google Image engine. The search keywords are "bar chart", "pie chart" and "scatterplot", respectively. Assumptions are made to filter out skewed scenarios and ensure image quality. The filtering criteria are listed in the supplementary material. After filtering, we obtain 208 bar charts, 161 pie charts, and 100 scatterplots in total. For each type of chart, two-thirds of the images are assigned to the training set, with one-third being the validation set.

Image Labeling: At the labeling stage, we outline the contour of each entity (Fig. 3b). For small dots in scatterplots, we use bounding boxes instead. Bounding boxes are also used for auxiliary entities since they do not need the exact contours for highlighting. After labeling, we have obtained $1,753/749$ rectangles, $1,878/829$ circles, $722/351$ sectors, $437/195$ axes, and $507/224$ color legends in the training/validation set.

Model Training: The training is based on Detectron [16], an open-source object detection library provided by Facebook AI Research (FAIR). The model we use is RetinaNet [41], a variant of Mask-RCNN that combines Feature Pyramid Network (FPN) [40] with Residual Neural Network (ResNet) [19].

The training runs at a workstation with 2 Intel(R) Xeon(R) CPU processors (48 processors in total, E5-2650 v4, 2.20GHz) and 8 GeForce GTX 1080Ti GPUs (11GB, 1582MHz). For the aforementioned dataset, the whole training process lasts for about 19 hours for around 70,000 iterations. The training only runs once for each deployment.

After training, the model can return both the bounding box and the image area of each detected entity. Boundary points are extracted to output the rough contour (Fig. 3c). We tested the performance of the model, details of which can be found in the supplementary material.

Contour Correction

For an auxiliary entity, we only need the bounding box for information extraction. But for a data entity, an accurate contour is essential for the purpose of highlighting. We add an extra step to get the precise contours.

Since we do not deal with textures, we assume that each data entity contains only one major color. Given the major color, we search inside the bounding box for the image area containing this color and outline its profile as the fine contour. However, we found that the dominant color inside a bounding box is not necessarily the entity's major color (consider concave polygons). That's why we need rough contours for more accurate color extraction. Fig. 3d shows an example where the contour has been fixed. Note that the correction procedure is only performed for shapes since we do not need precise contours of the auxiliary entities.

Visual Property Detection

With the fine contours, the next step is to extract visual properties of each data entity. It provides information to handle property-based description.

Color Detection: First, we turn the perceptual colors into 11 most frequently used color terms (see section "Property Detection" for more details). Color naming is a tough problem that introduces an independent research topic [4, 59, 21]. Based on the literature, we divide the HSV color space into 11 exclusive ranges. We examine the image area of each entity

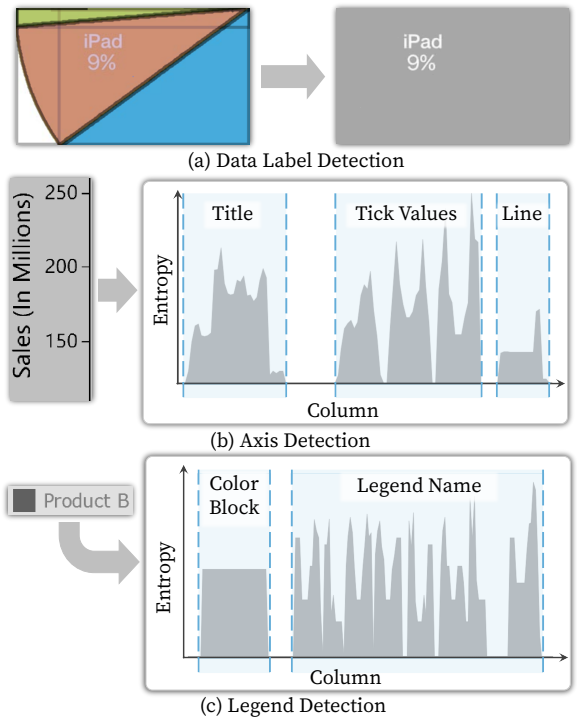


Figure 4. (a) To detect labels inside a data entity, we fill the area outside the contour with the entity's major color. (b) We divide the axis's bounding box into different parts based on row-/column-wise entropy of grayscales. (c) We divide the legend's bounding box into different parts based on column-wise entropy of grayscales.

and extract percentages of these 11 colors. The one with the largest portion is the major color.

Size Detection: Three size attributes are most frequently described: area, X-range, and Y-range. This applies to most cases in the Cartesian coordinates. With the contour, it is easy to derive all three attributes using basic measurements.

Location Detection: Given the contour, we simply output the position of its centroid as the location.

Text Detection

The challenge for text extraction comes from the limitation of our third-party engine, namely Google's Tesseract [1], a state-of-the-art OCR library. It may easily fail if the image contains anything other than texts. We locate and isolate the texts using image processing techniques to improve detection accuracy.

Data Labels: Data labels often lie inside a data entity, which is commonly seen in pie charts. We fill the areas outside the contour with the entity's major color to get an image that purely contains the label (Fig. 4a). However, the situation becomes complicated when the label appear outside the entity. See section "Discussion and Future Work" for more details.

Axes: An axis contains at least two components: the axis body and the tick values. Sometimes there is also a title. We first determine the axis's orientation (horizontal or vertical), judging by the size of the bounding box. For a horizontal/vertical axis, we derive the column-/row-wise entropy, which indicates the consistency of pixels in each image column/row (Fig. 4b). Intervals usually have the lowest entropy. Components other than the thinnest one (the axis body) often contain pure texts.

Legends: The handling of a color legend is similar. Two components are assumed: the color block and the legend name. The color block is the leftmost part with non-zero entropy (Fig. 4c). We extract its major color as the legend's color. The legend name is read from the remaining part.

After text detection, all visible texts are sent to the NLP module to help interpret the description.

Natural Language Processing

The NLP module aims to detect entities that are mentioned in the description and generate the corresponding queries.

Preprocessing

We adopt spaCy [23], an open-source NLP library written in Python, to pre-process the textual description. We use one of its core language models, which has been pre-trained using CNN on the OntoNotes 5 corpus. Specifically, we partition the description into multiple sentences, identify the part-of-speech (POS), and extract dependencies between the words (Fig. 5a).

Defining Knowledge Base

As mentioned in section "Describing a Visualization", there are two ways to describe a data entity in the visualization: use its visible label, or describe its visual properties. Visible texts are received from the OD module. Their synonyms are unlikely to be used. However, visual properties can be described with all sorts of words, which are invisible from the image, but familiar to the audience. Many of them are synonyms for each other. Therefore, we need a knowledge base to identify the keywords for describing visual properties.

First, we define two vocabularies. Despite the diversity of wording in natural language, we need a *standard vocabulary* to support internal communication between modules. Each standard word can have multiple synonyms, which are allowed to be mentioned and can be recognized by our system. Once identified, synonymous keywords will be transformed into standard words for subsequent processing. The *synonyms vocabulary* reflects the ability of our NLI to process different wordings. Together with visible texts, the vocabularies constitute the knowledge base for keyword matching (Fig. 5a).

In addition to the vocabularies, we also define a *structure library* to handle the diversity of expressions. For example, "the red point" and "the point in red" express the same meaning but have different dependencies between the keywords ("red" and "point). This library helps recognize the decorating relationship between keywords (Fig. 5b, 5c). It reflects the degree of freedom of sentence-making in our NLI.

The knowledge base (two vocabularies and one structure library) is defined for each type of visual property.

Entity Detection

Given the keywords and their relationships, we still need to know which keywords are the properties/entities. In the context of visualization, shapes always appear as nouns. Legends

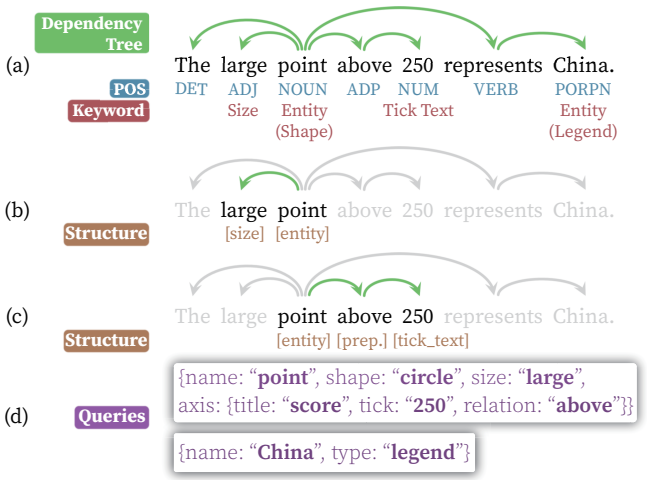


Figure 5. For each sentence, we derive (a) the dependency tree and the POS tags. We also identify keywords for describing visual properties and visible texts. Then we interpret the (b, c) decorating relationships based on the sentence structure. (d) shows an exemplary output.

and data labels will only be mentioned using their visible texts. These three types form the valid entities in the description.

We assume there is no need to highlight an individual axis. Therefore, axes are regarded as position-relevant properties (Fig. 5d). It can only be described by its title, unit, and tick values. The tick values are, in fact, reference points for spatial search. People express spatial filtering by describing the relationships between the data entity and the tick value. For example: "the bars above 250". Currently, eight standard words are defined to convey this spatial relationship: "at", "below", "above", and so on (see section "Entity Matching").

Property Detection

Among the visual properties, colors are special due to the ambiguity in its definition. Even a simple "red" could mean different colors to different people. Brent and Paul [4] conduct a worldwide survey to study the color terms in different languages. They find that 11 terms are most frequently used in English: "red", "orange", "brown", "yellow", "green", "blue", "purple", "pink", "black", "gray" and "white". We take these terms as the standard vocabulary for colors. Based on the survey, the authors also suggest a division of the color space, which we have taken into consideration in the OD module.

For sizes, we define six standard words: "large", "small", "long", and so on. They correspond to the two extrema in 3 attributes: area, X-range (width), and Y-range (length). For locations, the standard vocabulary contains five standard words: "middle", "top", "bottom", etc., corresponding to the two extrema in X and Y directions.

Descriptions of all visual properties have highly similar sentence structures. For example, the structure "[entity] [prep.] [color]" (e.g. "the bars in red") is also applicable to describe locations: "[entity] [prep.] [location]" (e.g. "the bars on the left"). Due to space constraints, we skip further details of the knowledge base. Both the vocabularies and the structure library can be extended to increase the freedom of expression.

Annotation

The annotation module has two major tasks: (1). to match the description with the entities, and (2). to highlight the mentioned entities with the description shown aside as the annotation.

Entity Matching

For each sentence, we perform an entity matching for each entity it mentions. Note that the "entity" here refers to an entity returned by NLP, which could be a shape, a data label, or a legend name. The matching procedure differs for different types of description.

If an entity is described by its visual properties, we can find it in two steps. First, we search the OD results for its category, i.e. its shape. Second, we search for the exact data entity/entities by combining the matching results for all visual properties.

Color Matching: Color matching is the easiest, since the OD module has returned the major color term for each data entity. With the percentage of colors, we also support color search concerning multiple colors (e.g. "the black and white pie").

Size Matching: Size matching is enabled for three attributes: area, X-range and Y-range. We rank all data entities in each size related attribute, and choose the highest ranked ones as the outcome. It is easy to achieve since both the quantity and the extrema are returned by the NLP module. If no quantities are provided (e.g. "the lowest bars"), we simply choose the top quarter as the focus.

Location Matching: Location matching is enabled for the X and Y directions. Similar to size matching, we choose the highest ranked entities. If no quantities are given, we choose the top quarter.

If an entity is described by auxiliary texts, there are three situations that should be handled separately.

Data Label Matching: If it is a data label, we can easily find the corresponding data entity. That is because the labels are already attached to their entities in the OD results. It also applies to situations where multiple entities have the same label.

Legend Matching: If it is a legend name, the search includes two steps. First, we extract from the OD results the visual feature described by this name. In our case, it is the digital color value of the legend. Any data entity could be chosen as long as its major color is close enough to the legend color.

Axis Matching: If an axis is mentioned, the situation is somehow complicated for two reasons.

First, we do not know if it is an ordinal, categorical, or numerical axis. For numerical axes, words like "below" and "above" convey threshold ranges starting at the exact position of the tick value. But for non-numerical axes where each tick value represents a range, the threshold starts at the boundary point of the tick range. To cope with this problem, we define two sets of standard prepositions for numerical and ordinal axes

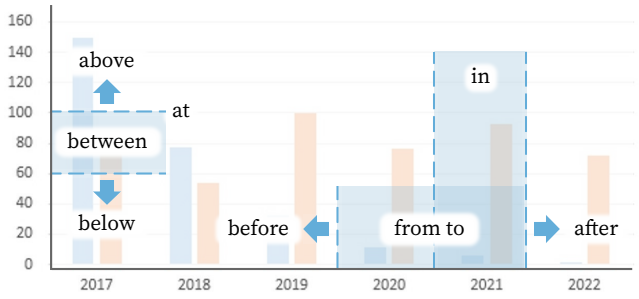


Figure 6. We identify the type of an axis (numerical, ordinal, or categorical) by the standard preposition used to describe the relationship between the data entity and the tick value. "at", "below", "above", and "between" are used for an numerical axis. "in", "before", "after", and "from ... to" are used for an ordinal axis. For a categorical axis, only the preposition "in" can be used.

respectively (see Fig. 6). Thus, we can recognize the type of an axis from the way it is described. For example: "before 2014" (ordinal) vs. "below 2014" (numerical). Categorical axes can only be described by relationship "in".

Second, we do not know how the axis ticks are used. In a bar chart, the Y axis is used to measure the heights while the X axis is used for locations. Our solution is to detect whether a data entity has intersections with the described threshold range in the axis's direction. If an intersection exists, the data entity will be chosen. Specially, for "below" thresholds, the entity must be entirely included in the threshold range. All thresholds concerning non-numerical axes are transformed into a union of "in" thresholds. For example, "before 2017" may be turned into "in 2014, 2015, and 2016".

In real applications, different types of descriptions are often used together. We simply get the intersection of all matching results as the final focus.

Sentence Anchoring

The previous step matches each sentence with the described data entity/entities. A sentence without any matched entity is called a context sentence. We assign each sentence to a single frame in a series of animations. It can not only avoid the occlusion between annotations, but also more fluently guide the audience's attention [53].

A context sentence will be displayed aside the whole image without any highlighting effect. For a sentence with focal entity/entities, we highlight the focal area(s), and find an appropriate position to display the description as the annotation. More specifically, we set forces between the focal areas and the annotation box to shorten their distances. Meanwhile, collision detection is performed to avoid undesired occlusion. If no available space is found, the search runs again with a new aspect ratio chosen from a predefined set of values. If all searches have failed, we display it beside the image.

Annotation Rendering

With experience drawn from the vast literature [35, 48, 33], we combine both internal and external visual cues to provide various types of vivid focus+context effects.

Internal visual cues refer to modifications to the existing entities in a visualization. Visual characteristics like transparency, saturation, depth of field (i.e. blurring), and brightness of image areas can be changed to de-emphasize the context.

External visual cues refer to all sorts of graphical overlays over the original image used for emphasizing the focus. They include entity contours, shaded areas, auxiliary lines on axes, and most importantly, the textual annotations. All data labels and color legend names are highlighted in the annotation using their major colors. It helps the audience better locate the corresponding entities.

Fading effects are used for the entrance/exit of each visual cue so that the audience can easily follow the changes between different scenes. In order to help the audience better understand the visualization, auxiliary entities are always remained as the focus in the entire animated presentation. For more details, please refer to the complementary video.

EVALUATION

In this section, we evaluate the effectiveness of our work via a user study and multiple use cases collected from users' inputs.

User Study

As mentioned in section "The Workflow", the proposed technique accomplishes four tasks: i.e., reading the chart, comprehending the description, visual searching, and annotation rendering. The former three tasks are carried out for the same goal: to match the description with the described image areas correctly. Given the annotations, the rendering task aims to better guide the audience's attention for a more fluid presentation. Considering the two goals are independent of each other, we verify them via two different studies.

Participants: We recruit the same group of participants for both studies. Since these studies serve different purposes, there will not be cross-influence. The participants consist of 10 graduate students with computer science and visualization backgrounds. There are 8 males and 2 females, aging from 21 to 26 (with a median of 24).

Procedures: First, we briefly introduce the goal of this work. Then we introduce the interface and the interactions of our prototype system (Fig. 1). In order to avoid skewed use cases that go beyond our scope, we explain the allowed chart types and other predefined assumptions (see the supplementary material). We also present the synonyms vocabulary so that participants are aware of the degree of freedom of their expressions. We elaborate on these metrics, simply because failures caused by undesired inputs cannot prove an approach ineffective. After the tutorial, each participant goes through the two studies respectively.

Assessing the Annotating

The goal of this study is to evaluate accuracy of the proposed annotating technique. Specifically, we ask each participant to prepare two charts that meet the metrics concerning chart type and image quality. We combine all 20 images into one corpus. Then each participant randomly pick out 12 images from the corpus and write one descriptive sentence for each image. Both types of descriptions (i.e., by visible texts and by

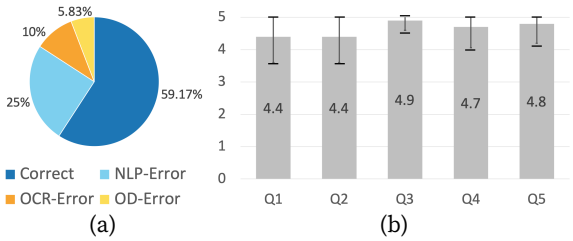


Figure 7. The user study evaluates (a) accuracy of the automatic annotating, ratios of different error types, and (b) the rendering effects regarding multiple aspects.

visual properties) are demanded. The image and its sentence are then uploaded to test the annotating accuracy. The result is regarded as "correct" if and only if all described focal areas have been correctly located and highlighted. At last, we have obtained 120 use cases for 20 images. The resulting annotating accuracy is 59.16% (Fig. 7a), with 71 correct cases and 49 incorrect ones.

In the next step, we further look into the reasons for the failures. First, we briefly examine some of the incorrect cases. It turns out all failures result from the back-end errors, i.e., wrongly identified results returned by the OD and NLP modules. Among the OD errors, some are caused by the object detection model (e.g., unrecognized data entities), while others are caused by OCR mistakes (e.g., wrongly identified texts). Therefore, we divide the errors into three types: NLP-Error, OD-Error, and OCR-Error and go through all 49 incorrect cases to examine their error types. Results show that up to 61.2% errors are cased by NLP mistakes, while only 24.5% and 14.3% are related to object detection and OCR (Fig. 7a).

There are two reasons why the NLP module is so error-prone. First, none of the participants are native English speakers. We find that many of the descriptions have grammar errors. Invalid sentence structures are particularly common. The wording has no such problem since the vocabulary is included in the tutorial. The second reason lies in the unstable nature of the language model. We find that dependency parsing results are highly sensitive to changes. For example, "the car on the left" and "the point on the left" could have different word dependencies. Using a more advanced NLP engine may improve this situation.

Assessing the Rendering

In this study, we evaluate the rendering of highlighting, annotations and animations. Since annotating accuracy is not the focus, we use a correct case to test its rendering results. Specifically, each participant is required to first view the given chart and read its textual description until he/she understands the data story. Then the participant needs to upload both the image and the texts to Vis-Annotator and view the generated animation. After that, we ask the participant to grade on the following questions (1 for the worst and 5 for the best):

1. Do the annotations occlude important image areas?
2. Are the annotating texts easy to read?
3. Are the highlights helpful for locating described contents?
4. Do the graphical overlays facilitate chart reading?
5. Are the animations easy to follow?

Judging from the results (Fig. 7b), most participants consider the highlighting (Q3), graphical overlays (Q4) and animations (Q5) useful for facilitating presentation and chart reading. In contrast, there is still room for improvement concerning the placement (Q1) and styling (Q2) of the annotations.

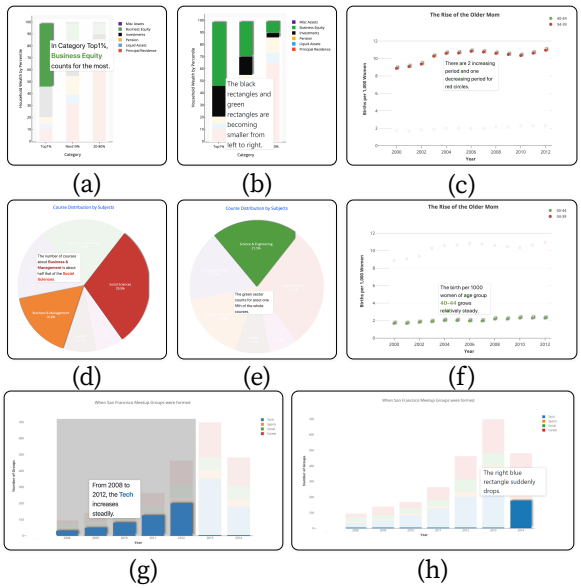


Figure 8. Correctly annotated cases in the user study

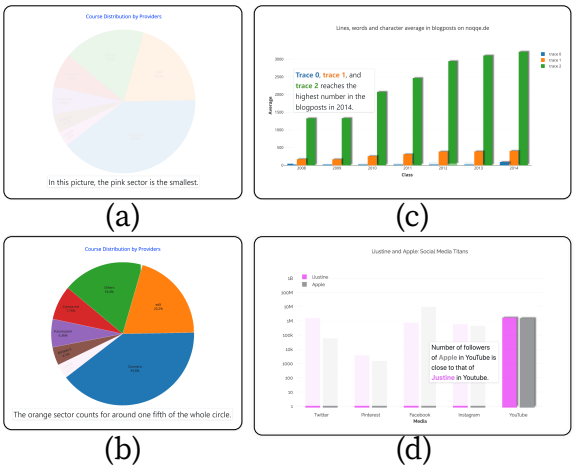


Figure 9. Wrongly annotated cases in the user study

Annotation Gallery

We collect several use cases from the user study to illustrate the effectiveness of our approach and also examine some of the failures to discuss the reasons behind them.

Fig. 8 shows 8 correctly annotated cases on four images including two bar charts, one pie chart, and one scatterplot. For each image, we include two cases with both types of descriptions. Fig. 8a, d, f, and g present cases with visible-text-based descriptions, while Fig. 8b, c, e, and h show cases with visual-property-based descriptions. See the supplementary material

for their textual descriptions. These cases demonstrate the effectiveness of our approach for handling different types of visualizations and descriptions. Sentences with multiple focuses (Fig. 8b, d) can also be properly handled.

More informative than the correct cases are the incorrect ones. Here we present 4 bad cases with typical errors. Fig. 9a shows a case where the user fails to highlight the smallest sector in a pie chart. We examine the detection results and find that the Mask-RCNN model fails to detect this sector. It is a typical case where the machine learning model fails in extreme scenarios. The only way to improve is to increase the diversity of the training data by including more extreme examples.

In the second case (Fig. 9b), user describes the orange sector but gets no response. This sector is not highlighted because its color is identified as "brown" by the system. It reflects the difficulty for color naming. We see that when nothing is highlighted, the smallest sector remains dimmed. It confirms our findings in the previous case.

Fig. 9c shows an NLP error where phrases like "A and B in 2014" fails while "A in 2014 and B in 2014" can be identified. Fig. 9d presents an OCR error where the legend name "iJustine" is wrongly identified as "Justine".

DISCUSSION AND FUTURE WORK

In this section, we discuss the current limitations of the proposed automatic annotating technique.

In section "Describing a Visualization", we introduce two ways to refer to a data entity: by its (visible) ID, or by its visual features. In fact, there exists a third type of description, namely by the relationship between entities. Consider the situation when entity "A" has just been described, and the presenter mentions an anonymous entity next to it. In fact, our technique already supports the detection of basic spatial relationships, such as adjacency and inclusion. However, more complicated relationships are often presented in relational data visualizations (e.g., node-link diagrams). In order to handle such relationships, we still need further studies to understand their universal patterns.

In the OD module, we do not detect data features considering their unpredictable appearances (e.g., clusters). Such a decision is subject to the model we used. In natural images, each detectable type has a rather stable appearance. When it comes to data features, the performance of object detection could be worse than specialized feature detection algorithms. One plausible solution is to classify the visualization type [52, 31] and detect data features accordingly using specialized algorithms.

There are also limitations regarding auxiliary entity detection. Automatic label reading itself could be an independent research topic. Currently, we only consider labels inside data entities. Text detection modules like Darknet [47] can be used to identify labels outside the data entities. However, further studies are still needed on how to match a label with its corresponding data entity, especially in a crowded space. The styles of axes and color legends are also constrained, e.g., only categorical legends are considered. These constraints exist merely to narrow down the focus of our work. Many challenges untouched here have already been addressed by other works [55, 45, 46].

In the NLP module, we base our detection on a universal language model, and predefined vocabularies and sentence structures. The current settings can already achieve accurate information extraction in most cases. However, when there is a new situation, configuration works are still required since different structures are handled case by case. A more intelligent way is to collect descriptions for visualizations and train an end-to-end language model for directly translating vis-descriptions into structured data. More NLP functions like coreference resolution can be added to enhance the degree of freedom in users' expression, though there is no technical difficulty in such extensions.

Despite being recognized in the user study, Vis-Annotator is not ready for production environments. In the current version, the user cannot make corrections if there are incorrect annotations. We also need to provide more controls over the presentation process. For example, speech recognition can be added to control the display of animations. Based on that, real-time annotating is also achievable in the near future. It will be great if the animations can change according to the contents of the presenter's talk.

CONCLUSION

In this paper, we propose an automatic annotating technique for promoting efficient data presentation with visualizations. The presenter can upload a visualization image with the corresponding textual description and get a series of vivid well-annotated animations. The process is highly efficient and can be finished in seconds. It saves the presenters lots of time and effort for annotation crafting, allowing them to focus more on designing the contents of data stories.

To achieve that, we adopt the state-of-the-art object detection model for effectively extracting information from the visualization. Natural language processing models and techniques are used for generating search queries from the description. By fulfilling the visual searches, we are able to locate the focal image areas of the entities described in each sentence. We provide various types of highlighting and annotating choices to help the presenters customize their presentations. Case studies with multiple use cases demonstrate the effectiveness of our approach. In the future, we plan to extend our technique for more accurate, real-time, speech-controlled visualization annotating.

ACKNOWLEDGMENT

This work is supported by NSFC No. 61872013 and NSFC No. 61672055. This work is also kindly supported by the Okawa Research Grant.

REFERENCES

[1] https://github.com/tesseract-ocr/tesseract.

[2] https://www.tableau.com/products/new-features/ask-data.

[3] Ronald Azuma and Chris Furmanski. 2003. Evaluating label placement for augmented reality view management.

In *Proceedings of the 2nd IEEE/ACM international Symposium on Mixed and Augmented Reality*. IEEE Computer Society, 66.

[4] Brent Berlin and Paul Kay. 1991. *Basic color terms: Their universality and evolution*. Univ of California Press.

[5] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D^3 Data-Driven Documents. *IEEE Trans. Vis. Comput. Graph.* 17, 12 (2011), 2301–2309.

[6] Chris Bryan, Kwan-Liu Ma, and Jonathan Woodring. 2017. Temporal Summary Images: An Approach to Narrative Visualization via Interactive Annotation Generation and Placement. *IEEE Trans. Vis. Comput. Graph.* 23, 1 (2017), 511–520.

[7] Yang Chen, Scott Barlowe, and Jing Yang. 2010a. Click2Annotate: Automated Insight Externalization with rich semantics. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology, IEEE VAST 2010, Salt Lake City, Utah, USA, 24-29 October 2010*. 155–162.

[8] Yang Chen, Jing Yang, Scott Barlowe, and Dong Hyun Jeong. 2010b. Touch2Annotate: generating better annotations with less human effort on multi-touch interfaces. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems, CHI 2010, Extended Abstracts Volume, Atlanta, Georgia, USA, April 10-15, 2010*. 3703–3708.

[9] Eun Kyoung Choe, Bongshin Lee, and m. c. schraefel. 2015. Characterizing Visualization Insights from Quantified Selfers' Personal Data Presentations. *IEEE Computer Graphics and Applications* 35, 4 (2015), 28–37.

[10] Sagnik Ray Choudhury, Shuting Wang, and C. Lee Giles. 2016a. Curve Separation for Line Graphs in Scholarly Documents. In *Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries, JCDL 2016, Newark, NJ, USA, June 19 - 23, 2016*. 277–278.

[11] Sagnik Ray Choudhury, Shuting Wang, and C. Lee Giles. 2016b. Scalable algorithms for scholarly figure mining and semantics. In *Proceedings of the International Workshop on Semantic Big Data, San Francisco, CA, USA, July 1, 2016*. 1.

[12] Jon Christensen, Joe Marks, and Stuart Shieber. 1995. An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics (TOG)* 14, 3 (1995), 203–232.

[13] Weiwei Cui, Xiaoyu Zhang, Yun Wang, He Huang, Bei Chen, Lei Fang, Haidong Zhang, Jian-Guan Lou, and Dongmei Zhang. 2020. Text-to-Viz: Automatic Generation of Infographics from Proportion-Related Natural Language Statements. *IEEE Trans. Vis. Comput. Graph.* 26, 1 (2020), 906–916.

[14] Tong Gao, Mira Dontcheva, Eytan Adar, Zhicheng Liu, and Karrie G. Karahalios. 2015. DataTone: Managing Ambiguity in Natural Language Interfaces for Data Visualization. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology, UIST 2015, Charlotte, NC, USA, November 8-11, 2015*. 489–500.

[15] Tong Gao, Jessica Hullman, Eytan Adar, Brent J. Hecht, and Nicholas Diakopoulos. 2014. NewsViews: an automated pipeline for creating custom geovisualizations for news. In *CHI Conference on Human Factors in Computing Systems, CHI'14, Toronto, ON, Canada - April 26 - May 01, 2014*. 3005–3014.

[16] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. 2018. Detectron. `https://github.com/facebookresearch/detectron`. (2018).

[17] Ross B. Girshick. 2015. Fast R-CNN. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. 1440–1448.

[18] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. 2017. Mask R-CNN. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. 2980–2988.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 770–778.

[20] Jeffrey Heer and George G. Robertson. 2007. Animated Transitions in Statistical Data Graphics. *IEEE Trans. Vis. Comput. Graph.* 13, 6 (2007), 1240–1247.

[21] Jeffrey Heer and Maureen C. Stone. 2012. Color naming models for color selection, image editing and palette design. In *CHI Conference on Human Factors in Computing Systems, CHI '12, Austin, TX, USA - May 05 - 10, 2012*. 1007–1016.

[22] Jeffrey Heer, Fernanda B. Viégas, and Martin Wattenberg. 2007. Voyagers and voyeurs: supporting asynchronous collaborative information visualization. In *Proceedings of the 2007 Conference on Human Factors in Computing Systems, CHI 2007, San Jose, California, USA, April 28 - May 3, 2007*. 1029–1038.

[23] Matthew Honnibal and Mark Johnson. 2015. An Improved Non-monotonic Transition System for Dependency Parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, 1373–1378.

[24] Enamul Hoque, Vidya Setlur, Melanie Tory, and Isaac Dykeman. 2018. Applying Pragmatics Principles for Interaction with Visual Analytics. *IEEE Trans. Vis. Comput. Graph.* 24, 1 (2018), 309–318.

[25] Weihua Huang, Ruizhe Liu, and Chew Lim Tan. 2007. Extraction of Vectorized Graphical Information from Scientific Chart Images. In *9th International Conference on Document Analysis and Recognition (ICDAR 2007), 23-26 September, Curitiba, Paraná, Brazil*. 521–525.

[26] Weihua Huang and Chew Lim Tan. 2007. A system for understanding imaged infographics and its applications. In *Proceedings of the 2007 ACM Symposium on Document Engineering, Winnipeg, Manitoba, Canada, August 28-31, 2007*. 9–18.

[27] Weihua Huang, Chew Lim Tan, and Wee Kheng Leow. 2003. Model-Based Chart Image Recognition. In *Graphics Recognition, Recent Advances and Perspectives, 5th InternationalWorkshop, GREC 2003, Barcelona, Spain, July 30-31, 2003, Revised Selected Papers*. 87–99.

[28] Jessica Hullman and Nicholas Diakopoulos. 2011. Visualization Rhetoric: Framing Effects in Narrative Visualization. *IEEE Trans. Vis. Comput. Graph.* 17, 12 (2011), 2231–2240.

[29] Jessica Hullman, Nicholas Diakopoulos, and Eytan Adar. 2013. Contextifier: automatic generation of annotated stock visualizations. In *2013 ACM SIGCHI Conference on Human Factors in Computing Systems, CHI '13, Paris, France, April 27 - May 2, 2013*. 2707–2716.

[30] Petra Isenberg and Danyel Fisher. 2009. Collaborative Brushing and Linking for Co-located Visual Analytics of Document Collections. *Comput. Graph. Forum* 28, 3 (2009), 1031–1038.

[31] Daekyoung Jung, Wonjae Kim, Hyunjoo Song, Jeongin Hwang, Bongshin Lee, Bo Hyoung Kim, and Jinwook Seo. 2017. ChartSense: Interactive Data Extraction from Chart Images. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, Denver, CO, USA, May 06-11, 2017*. 6706–6717.

[32] Eser Kandogan. 2012. Just-in-time annotation of clusters, outliers, and trends in point-based data visualizations. In *2012 IEEE Conference on Visual Analytics Science and Technology, VAST 2012, Seattle, WA, USA, October 14-19, 2012*. 73–82.

[33] Ha Kyung Kong, Zhicheng Liu, and Karrie Karahalios. 2017. Internal and External Visual Cue Preferences for Visualizations in Presentations. *Comput. Graph. Forum* 36, 3 (2017), 515–525.

[34] Nicholas Kong and Maneesh Agrawala. 2009. Perceptual interpretation of ink annotations on line charts. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*. ACM, 233–236.

[35] Nicholas Kong and Maneesh Agrawala. 2012. Graphical Overlays: Using Layered Elements to Aid Chart Reading. *IEEE Trans. Vis. Comput. Graph.* 18, 12 (2012), 2631–2638.

[36] Robert Kosara and Jock D. Mackinlay. 2013. Storytelling: The Next Step for Visualization. *IEEE Computer* 46, 5 (2013), 44–50.

[37] Abhinav Kumar, Jillian Aurisano, Barbara Di Eugenio, Andrew E. Johnson, Alberto Gonzalez, and Jason Leigh. 2016. Towards a dialogue system that supports rich visualizations of data. In *Proceedings of the SIGDIAL 2016 Conference, The 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 13-15 September 2016, Los Angeles, CA, USA*. 304–309.

[38] Bongshin Lee, Rubaiat Habib Kazi, and Greg Smith. 2013. SketchStory: Telling More Engaging Stories with Data through Freeform Sketching. *IEEE Trans. Vis. Comput. Graph.* 19, 12 (2013), 2416–2425.

[39] Bongshin Lee, Nathalie Henry Riche, Petra Isenberg, and Sheelagh Carpendale. 2015. More Than Telling a Story: Transforming Data into Visually Shared Stories. *IEEE Computer Graphics and Applications* 35, 5 (2015), 84–90.

[40] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. 2017a. Feature Pyramid Networks for Object Detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 936–944.

[41] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. 2017b. Focal Loss for Dense Object Detection. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. 2999–3007.

[42] Narges Mahyar and Melanie Tory. 2014. Supporting Communication and Coordination in Collaborative Sensemaking. *IEEE Trans. Vis. Comput. Graph.* 20, 12 (2014), 1633–1642.

[43] Gonzalo Gabriel Méndez, Miguel A. Nacenta, and Sebastien Vandenheste. 2016. iVoLVER: Interactive Visual Language for Visualization Extraction and Reconstruction. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, San Jose, CA, USA, May 7-12, 2016*. 4073–4085.

[44] Tamara Munzner. 2014. *Visualization Analysis and Design*. A K Peters.

[45] Jorge Poco and Jeffrey Heer. 2017. Reverse-Engineering Visualizations: Recovering Visual Encodings from Chart Images. *Comput. Graph. Forum* 36, 3 (2017), 353–363.

[46] Jorge Poco, Angela Mayhua, and Jeffrey Heer. 2018. Extracting and Retargeting Color Mappings from Bitmap Images of Visualizations. *IEEE Trans. Vis. Comput. Graph.* 24, 1 (2018), 637–646.

[47] Joseph Redmon. 2013–2016. Darknet: Open Source Neural Networks in C. `http://pjreddie.com/darknet/`. (2013–2016).

[48] Donghao Ren, Matthew Brehmer, Bongshin Lee, Tobias Höllerer, and Eun Kyoung Choe. 2017. ChartAccent: Annotation for data-driven storytelling. In *2017 IEEE Pacific Visualization Symposium, PacificVis 2017, Seoul, South Korea, April 18-21, 2017*. 230–239.

[49] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. 91–99.

[50] Arvind Satyanarayan and Jeffrey Heer. 2014. Authoring Narrative Visualizations with Ellipsis. *Comput. Graph. Forum* 33, 3 (2014), 361–370.

[51] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2017. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Trans. Vis. Comput. Graph.* 23, 1 (2017), 341–350.

[52] Manolis Savva, Nicholas Kong, Arti Chhajta, Fei-Fei Li, Maneesh Agrawala, and Jeffrey Heer. 2011. ReVision: automated classification, analysis and redesign of chart images. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, Santa Barbara, CA, USA, October 16-19, 2011*. 393–402.

[53] Edward Segel and Jeffrey Heer. 2010. Narrative Visualization: Telling Stories with Data. *IEEE Trans. Vis. Comput. Graph.* 16, 6 (2010), 1139–1148.

[54] Vidya Setlur, Sarah E. Battersby, Melanie Tory, Rich Gossweiler, and Angel X. Chang. 2016. Eviza: A Natural Language Interface for Visual Analysis. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology, UIST 2016, Tokyo, Japan, October 16-19, 2016*. 365–377.

[55] Noah Siegel, Zachary Horvitz, Roie Levin, Santosh Kumar Divvala, and Ali Farhadi. 2016. FigureSeer: Parsing Result-Figures in Research Papers. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VII*. 664–680.

[56] Arjun Srinivasan and John T. Stasko. 2017. Natural Language Interfaces for Data Analysis with Visualization: Considering What Has and Could Be Asked. In *Eurographics Conference on Visualization, EuroVis 2017, Short Papers, Barcelona, Spain, 12-16 June 2017*. 55–59.

[57] Arjun Srinivasan and John T. Stasko. 2018. Orko: Facilitating Multimodal Interaction for Visual Exploration and Analysis of Networks. *IEEE Trans. Vis. Comput. Graph.* 24, 1 (2018), 511–521.

[58] Barbara Tversky, Julie Heiser, Rachel Mackenzie, Sandra Lozano, and Julie Morrison. 2008. Enriching animations. *Learning with animation: Research implications for design* (2008), 263–285.

[59] Joost van de Weijer, Cordelia Schmid, Jakob J. Verbeek, and Diane Larlus. 2009. Learning Color Names for Real-World Applications. *IEEE Trans. Image Processing* 18, 7 (2009), 1512–1523.

[60] Shiv Naga Prasad Vitaladevuni, Behjat Siddiquie, Jennifer Golbeck, and Larry S. Davis. 2007. Classifying Computer Generated Charts. In *International Workshop on Content-Based Multimedia Indexing, CBMI '07, Bordeaux, France, June 25-27, 2007*. 85–92.

[61] Frank Wagner, Alexander Wolff, Vikas Kapoor, and Tycho Strijk. 2001. Three rules suffice for good label placement. *Algorithmica* 30, 2 (2001), 334–349.

[62] Jian Zhao, Michael Glueck, Petra Isenberg, Fanny Chevalier, and Azam Khan. 2018. Supporting Handoff in Asynchronous Collaborative Sensemaking Using Knowledge-Transfer Graphs. *IEEE Trans. Vis. Comput. Graph.* 24, 1 (2018), 340–350.

[63] Yan Ping Zhou and Chew Lim Tan. 2000. Hough Technique for Bar Charts Detection and Recognition in Document Images. In *Proceedings of the 2000 International Conference on Image Processing, ICIP 2000, Vancouver, BC, Canada, September 10-13, 2000*. 605–608.