Linking Text and Visualizations via Contextual Knowledge Graph

Xiwen Cai, Di Weng, Taotao Fu, Siwei Fu, Yongheng Wang, and Yingcai Wu, *Senior Member*, *IEEE*

Abstract—The integration of visualizations and text is commonly found in data news, analytical reports, and interactive documents. For example, financial articles are presented along with interactive charts to show the changes in stock prices on Yahoo Finance. Visualizations enhance the perception of facts in the text while the text reveals insights of visual representation. However, effectively combining text and visualizations is challenging and tedious, which usually involves advanced programming skills. This paper proposes a semi-automatic pipeline that builds links between text and visualization. To resolve the relationship between text and visualizations, we present a method which structures a visualization and the underlying data as a contextual knowledge graph, based on which key phrases in the text are extracted, grouped, and mapped with visual elements. To support flexible customization of text-visualization links, our pipeline incorporates user knowledge to revise the links in a mixed-initiative manner. To demonstrate the usefulness and the versatility of our method, we replicate prior studies or cases in crafting interactive word-sized visualizations, annotating visualizations, and creating text-chart interactions based on a prototype system. We carry out two preliminary model tests and a user study and the results and user feedbacks suggest our method is effective.

Index Terms—Natural Language Understanding, Text Visualization Linking, Knowledge Graphs

1 Introduction

In the past decades, data visualization has been extensively applied in data news, online reports, and interactive documents to illustrate data facts. In these applications, insight communication benefits from the integration of text and visualizations. On the one hand, graphical charts convey data more intuitively, thereby enhancing the arguments in the articles visually [1]. On the other hand, text descriptions (e.g., captions and annotations) provide explanations and reveal insights for visualizations, which facilitates data interpretation [2]. However, in many cases, text and visualizations are presented together but not effectively integrated. There may be numerous pieces of text referring to a visualization and many visual elements in the visualization, while the relationships between them are not obvious. To gain insights, readers need to pay effort for finding the association between text references and the corresponding visual elements.

To facilitate text-visualization integration, the VIS community have paid considerable attention to linking text and visualizations. Here we define *linking text and visualizations* as associating textual references and the corresponding visual representations via intuitive visual cues or interactions. Effectively linking text and visualizations helps to improve the reading experience for general readers in different scenarios. Typical examples include:

Word-sized Visualization. Embedding visualizations at the word scale into text could avoid splitting attention between text and visualizations, thus reducing cognitive load [3].

- *Xiwen Cai, Taotao Fu, and Yingcai Wu are with the State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China. E-mail: {xwcai, taotaofu, ycwu}@zju.edu.cn.*
- *Di Weng is with School of Software Technology, Zhejiang University, Hangzhou, China. E-mail: dweng@zju.edu.cn. Di Weng is the corresponding author.*
- *Siwei Fu is with School of Management, Zhejiang University, Hangzhou, China. E-mail: siwei.fu@zju.edu.cn*
- *Yongheng Wang is with Zhejiang Lab, Hangzhou, China. E-mail: wangyh@zhejianglab.com.*

Visualization Annotation. Well-annotated visualizations guide readers' attention and facilitate the interpretation of key points in charts [4] (see Figure 2 I 2 and II 1 versus IV 3 for a comparison between separated text and chart versus an annotated chart).

Text-Visualization Interaction. When readers interact with text (e.g., select a sentence), highlighting the visual elements mentioned in charts helps readers understand the relationship between text and charts [1], thereby improving the reading experience.

Readers would benefit from linking text and visualizations. However, for the authors such as data analysts and data journalists, manually crafting text-visualization links (TVLs) may involve customizing visualizations, changing the visual style of the text, specifying text-visualization interaction, etc. It usually requires advanced skills and could be tedious. To facilitate the process of linking text and visualizations, automatic models have been used for coupling text with visualizations or tables [1], [5], [6], [7] and annotating visualizations [8], [9], [10], [11]. Though existing methods can be adopted for crafting TVLs, most of them are specific to certain systems or usage scenarios and can hardly be generalized to different applications. Furthermore, when the automatic models produce errors, users have to revert to manual methods, which increases the workload and reduces efficiency. There is a lack of a mechanism that allows users to fully leverage the models' capabilities through human intervention.

In this paper, we present a semi-automatic pipeline consisting of four stages to allow users to iteratively specify TVLs. The input includes the dataset to be visualized, visualization specification, and text (descriptions, explanations, or questions that contain references to the visualization). Initially, users specify a chart and our model convert the chart to a KG (Stage I). Then, users provide the text, and our model extracts key phrases (data attributes and visual property values) from the text (Stage II) and then group the key phrases and map them to entities in the KG (Stage III), thereby connecting them to the visual elements. Default styles would be applied to the detected phrases (Stage IV), and users can

check the results of the model through interaction (e.g., brushing the interest text to see whether the referred visual elements are correctly highlighted). When errors are detected, users can revise the detected phrases (Stage II) or regroup the phrases (Stage III). After ensuring the correctness, users can revise the styles of the text and visual elements or specify their integration (Stage IV). The output of our pipeline can be an article with embedded word-sized visualization, an annotated visualization, or an interactive article with text visualization interactions. Presently, we support 6 basic chart types (bar chart, line chart, scatterplot, bubble chart, donut chart, and pie chart).

In our approach, KGs serve as both the medium and the context for crafting TVLs. We reformulate the task of associating text and visualizations as the task of associating text and KGs, and there are several benefits. First, by representing visualizations as KGs, our method becomes independent of specific visualization systems or grammars. This allows for wider applicability in associating text with visualizations. Second, KGs can provide contextual information to help identify relevant entities and relationships mentioned in the text. This aids in detecting references from text automatically. Third, by binding phrases to KG entities and transforming phrase relationships into KG patterns, we can easily retrieve the associated visual elements. This practice streamlines the process of mapping text references to the corresponding visual elements. Fourth, the phrase-entity bindings and phrase relationships can be manipulated by users, enabling human-model collaboration. Users are allowed to revise the intermediate results of different stages and the model would update its behaviors.

We demonstrate the effectiveness of our method by replicating the cases of annotating visualizations [11], crafting interactive word-sized visualizations [12], and creating text-visualization interactions [13]. To evaluate the performance of our automatic model, we conducted two preliminary tests. First, we compare the performance of our model with Kori's model [7]. The results indicate that our model outperforms Kori's model in terms of reference detection. Second, we test the performance of our model based on the dataset provided by Kim et al. [14], which includes human-generated questions and explanations that contain references to charts. Our model achieved an overall accuracy of 69.0%, with 62.8% for the questions and 76.3% for the explanations. To evaluate the usability of our pipeline, we conducted a task-based user study based on our prototype system. Users used the system to craft TVLs and provided positive feedback on our method.

The major contributions of this work are as follows:

- A semi-automatic pipeline, which enables users to craft TVLs through collaborating with an automatic model.
- A system prototype and three usage scenarios that demonstrate the usefulness and the versatility of the developed method.

2 Related Work

2.1 Intelligently Linking Text and Visualizations

Researchers have explored methods for linking text and visualizations for improving document reading experience (e.g., [1], [6], [15]), reducing attention splitting (e.g., [3]), facilitating storytelling (e.g., [11]), etc. In our work, we focus on methods that use automatic models for resolving the relationship between text and visualizations. Additionally, in some visualization systems with text generation features (e.g., [16]), the linking between system-generated text and visualization can be pre-applied. These methods are less relevant to us as they generate text rather than process it.

Text and visualizations can be linked in different manners. Goffin et al. [17] characterize two modes of combining text with word-sized visualizations: document-centric (text is the reading focus) and visualization-centric (visualizations are the reading focus). Inspired by them, we categorize existing work into three categories: text-centric, visualization-centric, and non-centric.

Text-Centric TVL: text is in the central role, and visualizations are incorporated into the text space (e.g., as word-sized visualizations) or adapted to the text (e.g., as overlaying tooltips when hovering the text) to provide additive context. Typical text-centric methods for linking text with visualizations are word-sized visualizations. Researchers have explored methods [12], [18] for authoring documents containing interactive word-scale visualizations. However, these methods entail labeling text with HTML tags, while it is unclear whether these tags can be created intelligently. Additionally, there is research on linking text to visualizations automatically generated for adaptive document reading. Elastic Documents [6] automatically couples text to tables, generates visualizations, and links text to visualizations. Metoyer et al. [19] present a method for crafting TVLs by automatically extracting information from the text and combining it with data to generate visualizations.

Visualization-Centric TVL: visualizations are in the central role and text is embedded into visualization space (e.g., as annotations overlaying on visualizations) or adaptive to visualizations (e.g., as captions alongside visualizations) for explaining visualizations or guiding attention. Annotating visualizations is a typical visualization-centric method for linking visualizations with text. Contextifier [8] automatically products annotated stock visualizations according to news articles. Its successor, NewsViews [9], generates annotated maps in a similar manner based on text mining. While these two systems focus on additive annotations (annotations from external data sources which provide context beyond the visualizations), Temporal Summary Images [10] supports both observational annotations (annotations that describe the visualizations) and additive ones for multi-variate temporal visualizations. More recently, Lai et al. [11] present a method for annotating visualization images, which is based on object detection and NLP models.

Non-Centric TVL: it is unclear whether text or visualizations is in the centric role, and they are combined by applying visual cues (e.g., highlighting the mentions and the corresponding elements with the same background color) but without the embedding or adaption above-mentioned. Kong et al. [1] propose a crowdsourcing pipeline for extracting references between text and visualizations. These references were highlighted to help readers better understand the relation between text and visualizations. Kim et al. [5] present a pipeline for linking text and tables via automatically extracting and highlighting the references between them. Kori [7] is a mixed-initiative interface which uses an automatic model to suggest references and allows users to manually create TVLs.

The existing methods are specific to certain visualization systems or scenarios. It is unclear how they can be generalized to different applications. Moreover, most of these methods barely support the scenarios where users interact with the text consecutively and flexibly, such as document editing. To our knowledge, only Kori [7] supports flexible TVL authoring, which is most related to us. Kori proposes an interactive interface for creating interactions between text and charts while we propose a semi-automatic pipeline that can be applied to different scenarios and applications. Though Kori provides automatic suggestions of TVLs and allows users to

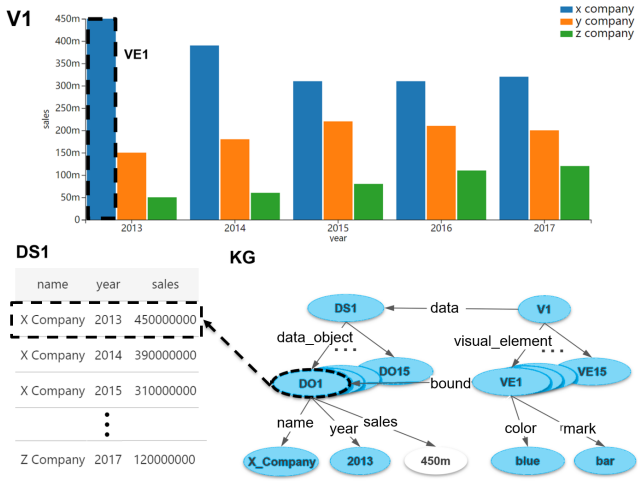

Fig. 1. A simplified knowledge graph (bottom right) for representing the grouped bar chart (top) and its underlying data (bottom left). The left part of the knowledge graph describes the data while the right part represents the bars in the visualization. Blue nodes present entities and white nodes represent literals. For simplicity, more specific details are left out.

manually specify TVLs, it lacks a mechanism to allow users to collaborate with its model. When Kori's model produces errors, users can only resort to a manual method, which increases the workload and reduces efficiency. With our method, users can revise the intermediate results produced by the model at different stages, thereby better utilizing the model and enhancing efficiency.

2.2 Data Visualization with Knowledge Graphs

KGs are widely adopted in applications such as search engines, recommendation systems, and dialogue systems. By leveraging ontologies as schema layers, KGs support logical inference for reasoning over data. In the visualization domain, considerable attention is paid to visualizing ontologies and KGs. While the majority of them focus on browsing and exploring ontologies and KGs in a hierarchical layout or graph format, several works [20], [21], [22], [23], [24], [25] have explored methods for extracting data from KGs, applying data transformations, and visualizing the processed data. Except for visual analytics on KG data, the VIS community rarely explore adopting KGs and related techniques to support visualization applications. To the best of our knowledge, only CAVA [26] uses KGs for data augmentation of tabular data to facilitate exploratory analysis and KG4VIS [27] leverages a KG embedding technique for visualization recommendation. Our work is another attempt at employing KGs for solving visualization-related tasks. In our work, KGs serve as the context and medium for facilitating of the creation of TVLs.

3 Knowledge Graph Usage

Recent visualization research with KGs (e.g., [23], [26]) has provided a rich introduction to the basic knowledge of KGs. To avoid redundancy, we omit the basic introduction of KGs and start by explaining a simplified KG for representing a grouped bar chart borrowed from Lai et al. [11]. Then, we provide an exemplary use case to help readers comprehend how it works.

The KG shown in Figure 1 is used to depict the bars in the visualization (**V1**) and their underlying dataset (**DS1**). In the KG, two nodes (subject and object) linked by a direct edge with semantical meaning (predicate) in the KG indicate a **(subject, predicate, object)** relationship. In our work, we use KGs mainly for depicting entity-attribute-value relationships. We exemplify several triplets related to the visual element **VE1** and its data object **DO1** as follows:

(DO1, year, 2013): 2013 is the value of the **year** attribute of **DO1**.
(VE1, bound, DO1): DO1 is **bound** to **VE1**.
(VE1, color, blue): blue is the value of the **color** property of **VE1**.

These triplets can be stored in a Resource Description Framework (RDF) database. In KGs, literals are distinguished from entities. Literals are values of general data types (e.g., number and string) while entities are objects with properties and relations. A formal instance of triplets stored in RDF form is like (**ltv-ctx:VE1, ltv-owl:bound, ltv-ctx:DO1**). For simplicity, prefixes are omitted and we use **bold font** to represent entities.

A simple use case is to find those VEs which represent "the sale of X Company in 2013". After detecting **X Company** and **2013** and identifying the corresponding attributes (**name** and **year**), we can make a SPARQL (an RDF query language) query like:

```
SELECT ?ve WHERE{
?do name X_Company.
?do year 2013.
?ve bound ?do.}
```

In this query, our purpose is to find those visual elements, so it begins with "SELECT ?ve" (In SPARQL, variables are prefixed by "?"). Then, each sentence in the braces stands for a triplet pattern to be found in the RDF database. First, we find those data objects (?do) that have values of **X Company** and **2013** via the first two sentences. Then, we obtain the bound visual elements (?ve). Same with Cashman et al. [26], we also believe KGs simplify the way to "think about the relationship of data objects".

4 Overview

In this section, we first introduce the common tasks in linking text and visualizations. Then, we present our four-stage pipeline for supporting these tasks. We implement a prototype system and incorporate a usage scenario to illustrate our pipeline and demonstrate the effectiveness of our method.

4.1 Task Analysis

In order to design an intelligent method for linking text and visualizations, we summarized the tasks in this process according to the existing research works (as mentioned in subsection 2.1). We identified four common tasks, which can be automatically completed by an automatic model, manually by users, or in a mixed-initiative manner through human-model collaboration.

T1 Visualization Specification and Abstraction. In many scenarios, visualizations are not assumed to be preexisting, and specifying visualization is a necessary step. Especially in some text-centric applications (e.g., [6]), visualizations need to be created according to the content of the text. In addition, it involves abstracting them into operational data objects via extracting the data and visual encoding.

T2 Information Extraction from Text. To resolve the relationship between text and visualizations, it generally requires extracting key information from the text. The key information includes: entities (e.g., Google), numerals (e.g., 30), and dates (e.g.,

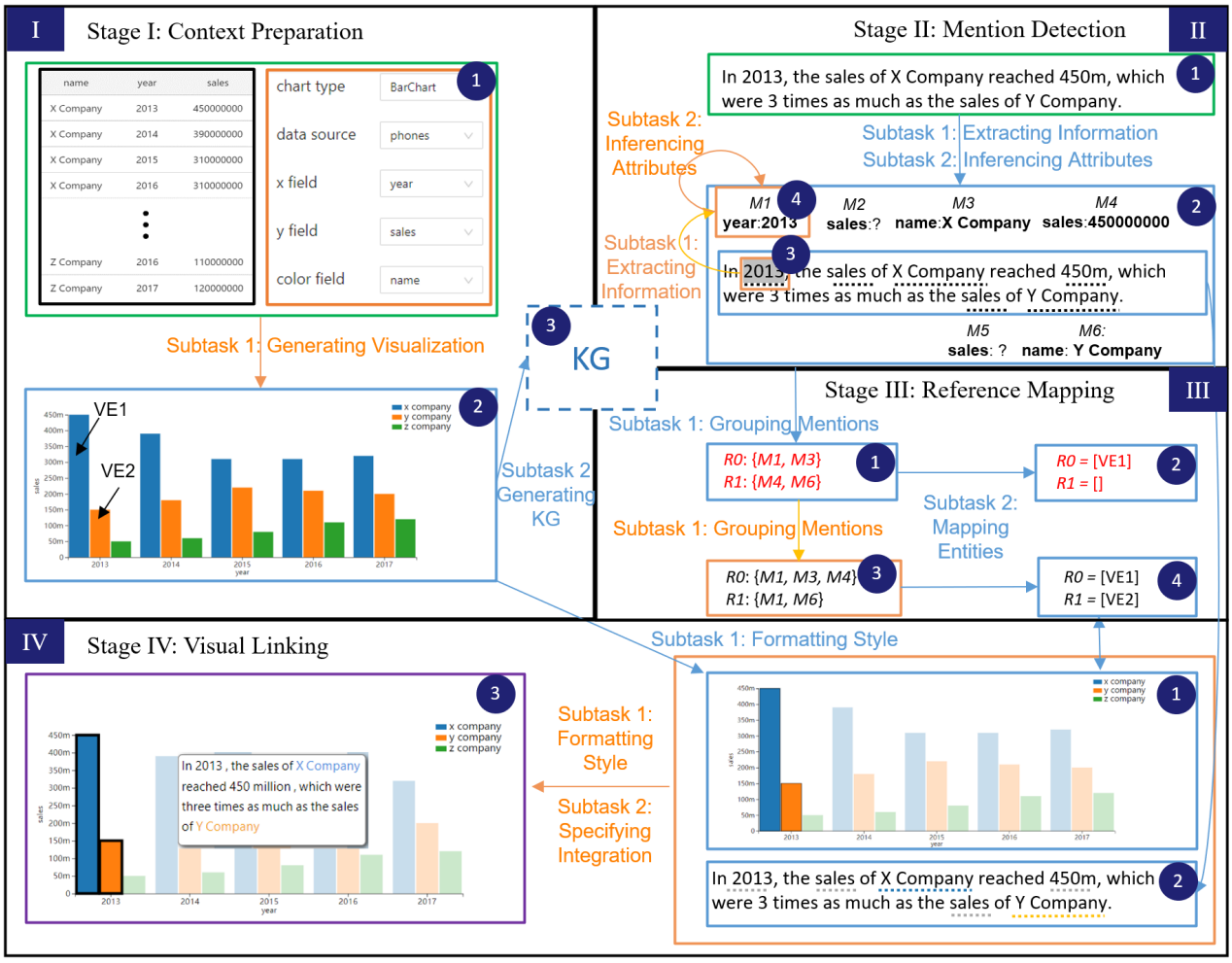

Fig. 2. An illustration of our four-stage pipeline for linking text and visualizations. Green boxes and purple boxes indicate the input and output respectively. Blue boxes and arrows represent the results and the tasks done automatically while orange ones represent those done by users. After a user uploads the data and specifies the visualization (I 1), our model renders the visualization (I 2) and generates the knowledge graph (I 3). Then, the user inputs the text (II 1), our model detects the mentions of visual elements through extracting the key information and making attribute inferences (II 2). The user can revise the results of information extraction and attribute inference via interactions (II 3 and II 4). In this case, the user does not change the result since the mention (*M1*) is correctly detected. Then, the mentions are grouped into references (III 1) and then mapped to the entities in the KG (III 2). In this example, since the mentions are wrongly grouped, the user regroup the mentions (III 3) to get the desired result (III 4). Finally, default styles are applied to the chart (IV 1) and the text (IV 2), and the user could reformat the styles and specify the text-visualization integration (IV 3).

2021) that can be mapped to the attributes and values in a dataset; visual descriptions (e.g., red bar) that can be mapped to the visual features in visualizations; particular expressions that require further parsing (e.g., the largest country).

T3 Mapping between Text and Visual Elements. The key task for text-visualization linking is mapping text references with visual representations. In this process, it may require combining the key information to form integrated references according to the context of the visualization. For example, in "the sales of X Company in 2013", the key phrases (**sales**, **X Company**, **2013**) should be combined into one reference.

T4 Visually Linking Text and Visualizations. Text and visualizations are visually integrated to improve the reading experience and facilitate storytelling. The visual linking can be established by creating spatial proximity (e.g., embedding word-sized visualization alongside the text and adding text annotations to the visual elements), visual similarity (e.g., text mentions and the corresponding visual elements are highlighted with the same background color), interaction, etc.

4.2 Pipeline and Prototype System

We propose a four-stage pipeline (Figure 2) to support the above-mentioned tasks for intelligently linking text and visualizations. Each stage corresponds to one task and is divided into two subtasks. Each of the subtasks can be solved by users, completed by the automatic model, or dominated by users while the model provides default solutions. Our method follows "agency plus automation", the design rationale proposed by Heer [28] which encourages integrating human agency with machine automation in an interactive system.

To better illustrate our pipeline as well as to demonstrate the effectiveness of the proposed method, we developed a prototype system and the interface is shown in Figure 3. For the sake of simplicity in illustration, we use *mention* to represent a key phrase (e.g., **X Company**) which describes an attribute value of the visual elements and *reference* to represent a group of conjunct mentions (e.g., the **sales** of **X Company** in **2013**) that refer to the same visual element(s). After a user specifies a visualization (Figure 3 I),

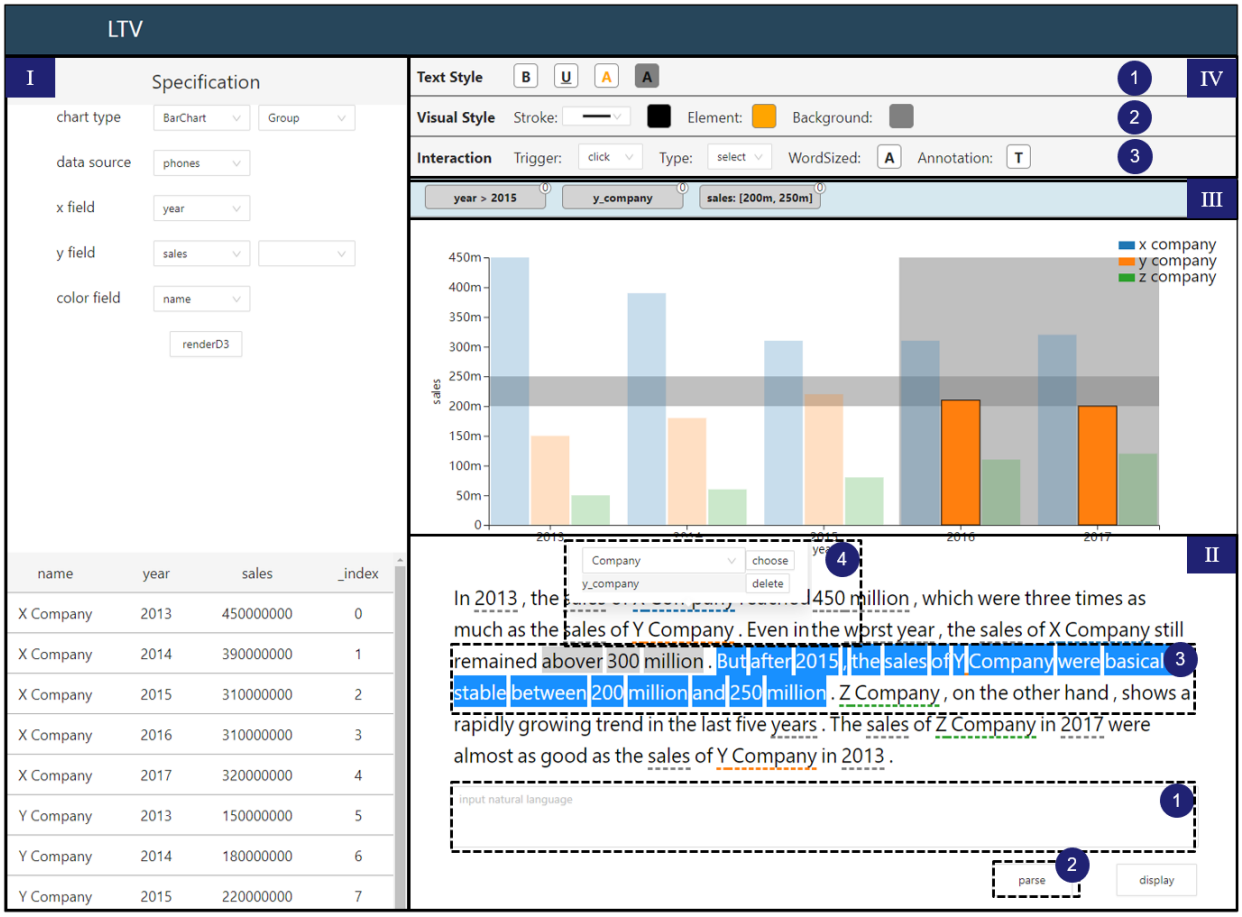

Fig. 3. The interface of our prototype system. The left panel (I) is used for specifying visualization showing the underlying data. In the right part, users type in the text and parse it (II 1 and 2). Then, they can check and edit the result of automatic linking. Users can brush a sentence (II 3) to see the result or right click a detected key phrase to check and set its alias (II 4). Moreover, users could regroup the key phrases via interaction on the tag box (III). They can further edit the visual linking through manipulations of the menu bars (IV).

the model represents the visual elements with the underlying data as a KG (T1) in Stage I. Then, the user types in text and triggers parsing (Figure 3 II 1 and 2), and the model extracts mentions from the text (T2), and infers the attribute type (e.g., name and year) of each mention in Stage II. After that, the model first combines the mentions into references and then maps the references to the KG entities which represents the visual elements (T3) in Stage III. The user can interact with the text to check the referred visual elements (Figure 3 II 3), and reset a mention (Figure 3 II 4) or regroup several mentions (Figure 3 III). After the previous stages, the relations of the text and the referred visual elements are semantically resolved but are not visually established. Default visual styles are applied to both text and visual elements, and users can further specify their styles and integrations via manipulation (Figure 3 IV) in Stage IV.

In the following subsections, we incorporate a concrete usage scenario borrowed from Lai et al. [11] to introduce the each stage of our pipeline. Bob, a market analyst, uses our system for crafting an annotated bar chart to illustrate his findings. To avoid verbosity, we have relegated some technical details such as predefined keywords to the supplementary materials.

4.2.1 *Stage I: Context Preparation*

In this stage, a visualization is specified (Subtask 1) and the system would transform the visual elements together with their underlying data into a KG stored in an RDF database (Subtask 2). We limit the data fields that users can match on different visual channels (e.g., only numeric data can be mapped to size channel) to avoid mismatches between data and visualizations.

Subtask 1: Generating Visualization. The first step of our pipeline is to generate a visualization. In our pipeline, it is assumed that data and visualization specifications are available, and visual encoding can be extracted from the specifications. Currently, user are required to manually specify visualizations. We discuss the techniques for facilitating visualization specification in the second limitation (**Support for Visualization Specification**) in subsection 7.1.

Subtask2: Generating KG. After a visualization is specified, our model generates a contextual KG to describe the data as well as the visual elements. We extend the method of Antonion et. al [29] in generating KG from tables, as it does not cover how to represent visual elements. In some scenarios, users may specify data transformations in a specification. For example, users may declare mapping the sum of sales to y channel when using Vega-lite or Tableau. Our backend module would bind visual elements to data objects in the transformed dataset. We assign a unique ID to each of the visual elements, by which the model could access the visual elements. An exemplar KG for representing visual elements is shown in Figure 4. Blue nodes represent entities and white nodes represent literals Since data objects and visual elements are in a one-to-one relationship, data objects (marked with a red dashed oval) are integrated into visual elements and omitted in the

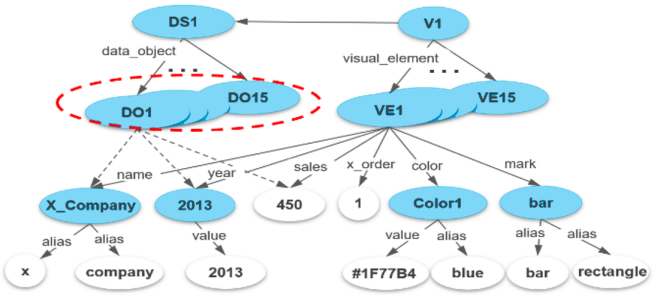

Fig. 4. A subgraph of our KG used to depict a data object and the corresponding visual element. Green nodes represent entities and white nodes represent literals. For convenience, we assign the attributes of the data object to the visual element. The original data objects are omitted in the consequent processes.

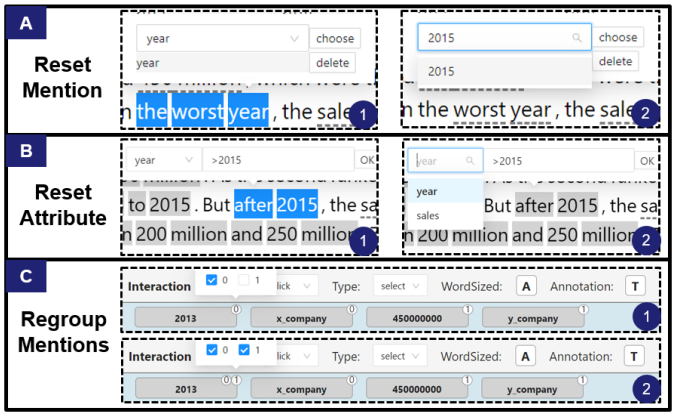

Fig. 5. The interactions for revising the errors, which include resetting mentions (A), resetting attributes (B), and regrouping mentions (C).

consequent process.

Use Case. In order to craft an annotated bar chart, Bob needs to generate a bar chart first. He chooses a CSV file which contains three fields (**name**, **year**, and **sales**) and 15 rows. It describes the **sales** of three Companies (**X Company**, **Y Company**, and **Z Company** in the **name** field) from **2013** to **2017** (in **year** field). After uploading the file, he specifies a bar chart by mapping **year** to the x channel, **sales** to the y channel, and **name** to the color (Figure 3 I). Then, he presses the "render" button to generate the visualization and the system would soon get prepared.

4.2.2 *Stage II: Mention Detection*

In this stage, our model first extracts mentions from the text (Subtask 1). Then, to generate RDF queries in an entity-attribute-value form in the next stage, it infers the corresponding attributes (data attributes and visual properties) of the mentions of values (Subtask 2). For example, our model first extracts "2013" from the text and then infers that it is a value of **year**. However, there may exist ambiguity issues discussed in the previous works [30], [31]. In our pipeline, similar to the previous researches on NLIs (e.g., [30], [31]), users can interactively revise the result.

Subtask 1: Extracting Information. The initial process of our method for mention detection is extracting key phrases from text (T2). We use Stanford's CoreNLP [32] for obtaining the lemmas, POS tagging, NERs of the words, and the dependency parsing result of the sentence. We extract three types of mentions: direct mention, filter expression, and data operation. Direct mentions are the keywords that can be mapped to the aliases or the values of the entity in the KG. Based on the lemmas of the words and the aliases of the entities, we map the words to the entities. To find those n-gram phrases as well as to reduce ambiguity, we combine the sequential words into an integrated mention that can be mapped to the same entity. Besides, we extract cardinal numbers via POS tag ("CD") and date via NER ("DATE"), and try to map them to the literals and entities in the KG (e.g., 450m and **2013**). Filter expressions are the numeric ranges (e.g., "between 200m to 250m" and "after 2015"), which are identified according to the combination of predefined keywords with cardinal numbers and dates. Data operations are extracted according to predefined keywords (e.g., "largest") and mapped to data operations (e.g., *MAX*). In terms of data operation, we only support maximum and minimum, which are transformed to argmax and argmin operations. We do not include sum and average since they make no difference to the referenced entities. We do not support composite data operations (e.g., "the largest sum"), which is left to future work.

Subtask 2: Inferencing Attributes. After detecting the mentions, we transform them into an "attribute: value" form (e.g., "**name**: **X Company**" and "**year**: **2013**") for querying the visual elements. Among the mentions, we distinguish attribute mentions (direct mentions of attributes) from value mentions (direct mentions of values, filter expressions, and data operations). For value mentions, we would infer their attributes to accomplish the transformation. The criteria for attribute inference is that the data type of the attribute should be consistent with the value mention. In some cases, we can find a proper attribute mention for a value mention in the context. For example, in "The sales is the largest", **sales** can be the attribute of *MAX* (largest). In other cases, we need to make inferences since the valid attribute of a value mention is not in the context. Finding the attribute of a direct mention of value is straightforward since this can be done by retrieving the triplets in KGs. If a direct mention is linked by different attribute links in the KG, we would assign it with the most frequent attribute. To exactly determine the attribute of a filter expression or a data operation is somehow challenging when there exist different attributes that match the criteria. Therefore, we used heuristic methods to generate default results. For an filter expression, we would consider the ranges of the data attributes to find one that has the largest overlap with the FE. For an data operation, we try to assign a data attribute according to the priority of visual channels (size $>$ y $>$ x). The priority rule is also applied for an filter expression when two or more data attributes have the same overlap with it.

Use Case. After the visualization is generated, Bob types in the text and presses the "parse" button. Through exploring the text, Bob finds that while many key phrases are correctly detected, there exist some mistakes in the process of mention detection (Stage II). In "Even in the worst year, the sales of X Company still remained above 300 million", the model failed to recognize "the worst year" as **2015** given the context of "the sales of X Company ". Bob brushes this phrase and right clicks it (Figure 5 A1), and then sets it aligned to **2015** via typing in "2015" and choosing the item in the drop-down list (Figure 5 A2). Besides, he also checks whether the model properly processed "after 2015". He right clicks "after 2015" and finds it is recognized as "**year**: $>$ 2015" (Figure 5 B1). If the model wrongly inferred the attribute, he can change the attribute via choosing the right one (Figure 5 B2).

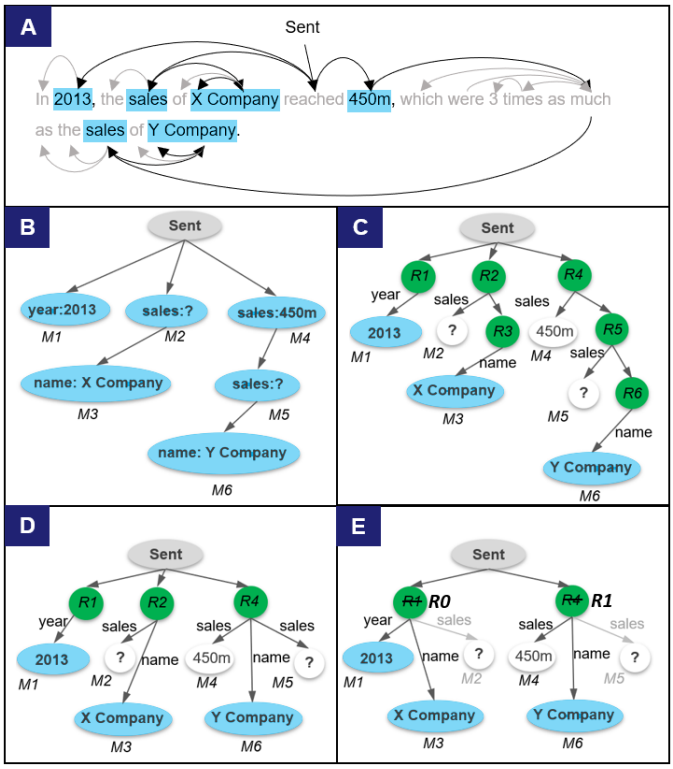

Fig. 6. An example of mention grouping. We begin with the result of dependency parsing (A). The non-detected words (in grey) are omitted and the original tree is converted into a tree representing the dependency relations of the mentions (B). Then, the mentions are transformed into reference nodes (green node) which contains "attribute: value" relationships (C). The reference nodes are grouped in a bottom-up manner via post-order traversal (D-E). Finally, we omit the non-specified mentions (M2 and M5 in E) and reindexed the references.

4.2.3 Stage III: Reference Mapping

In this stage, we construct the relationship between the text and the referred visual elements (T3) via first grouping the mentions into references (Subtask 1) and then mapping the references to the visual elements (Subtask 2). Each of the references is an RDF query like the example provided in section 3, which can be used to find the visual elements. In many cases, it can be a challenging task (called linearization) to detect coherent phrases among the conjuncts [33]. To address this challenge, we adopt a heuristic method inspired by Evizon [33] for automatically grouping the mentions. Users can regroup the mentions via interaction to revise the result of the automatic model. After mention grouping, the referred visual elements of each reference are found through KG queries. Note that users can select any segment of the text and mention grouping only defines how the selected mentions are combined.

Subtask 1: Grouping Mentions. In order to group the conjunct mentions, our model converts each of the sentences into a tree manner based on the result of dependency parsing (Figure 6 A) after extracting the mentions. In this process, unlike most of the existing works (e.g., [11], [30], [34]), the relation types of the dependencies are omitted. This is because we believe a keyword in a dataset can be any POS in a sentence and have any dependency relations with the other words. We combine the words in detected chunks and link each of them to the lowest common ancestor of the words in it (Figure 6 B). We begin with regarding each of the mentions as a reference (Figure 6 C). Then, our model tries to group mentions in a bottom-up manner via post-order traversal inspired by Evizon [33]. The post-order traversal ensures that the mentions in the same dependency brunch are combined first (Figure 6 D). Two references can be combined if they do not contain the same mention and there is no mention conflict after they are combined. The types of mention conflict are listed in Table 1. After combining the references, we omit *M2* and *M5*, whose values are not specified, and the result is Figure 6 E. Note that in this process, *R1* and *R4* are reindexed as *R0* and *R1* respectively to avoid confusion to the users.

TABLE 1
Types of mention conflicts.

Mention Type	Conflict Type	Example
Direct Mention & Direct Mention	same attribute	"**name**:**X Company**" & "**name**: **Y Company**"
Direct Mention & Filter Expression	same attribute no overlap	"**year**: **2013**" & "**year**: > 2015"
Filter Expression & Filter Expression	same attribute no overlap	"**sales**: $< 200m$" & "**sales**: $> 250m$"
Data Operation & Data Operation	same attribute	"**sales**: **MIN**" & "**sales**: **MAX**"

Subtask 2: Mapping Entities. After elementary mentions are grouped into references, our model maps them to the KG entities which represent visual elements. In Figure 6 E, each reference node consists of two triplet patterns, which can be seen as a reference to the visual elements. For example, *R1* consists of **(?, year, 2013)** and **(?, name, X Company)**, which refers to **VE1**. The references can be transformed into KG queries as illustrated in section 3. Our method supports the queries consisting of there types of patterns, each of which is corresponding to a relational query pattern: direction mention is corresponding to lookup (e.g., find the visual elements whose **year** is **2013**), filter expression is corresponding to filter (e.g., find the visual elements whose **sales** are larger than 200m), and data operation is corresponding to aggregate (e.g., find the visual elements whose **sales** is the largest).

Use Case. Bob brushes a sentence ("In 2013, the sales of X Company reached 450m, which were 3 times as much as the sales of Y Company.") and finds there is something wrong in reference mapping (Stage III). Only the bar which represents the **sales** of **X Company** in **2013** (VE1 in Figure 2 A2) is selected, while he expects to see both this bar and the one which represents the **sales** of **Y Company** in **2013** (VE2 in Figure 2 A2) to be selected. After checking the mention groups, he notices that the mentions are wrongly grouped: **2013** is only in the first group but not in the second group; 450m, which should be in the first group, is assigned to the second group (Figure 5 C1). Thus, he makes **2013** in both groups and changes the group index of 450m from 1 to 0 (Figure 5 C1-C2). After that, he sees the correct visual elements are selected when he brushes the sentence.

4.2.4 Stage IV: Visual Linking

The last stage of our pipeline is visually linking text references and visual elements (T4). After the previous stages, the relationship between the text and the visualization is resolved, but they are not linked visually. Effective TVLs usually involve the change of the styles of text and the visual representations (Subtask 1). Then, how they are integrated to achieve TVLs should be defined (Subtask 2). Previous researchers have proposed different methods for the integration and interaction between text with visualization, as mentioned in subsection 2.1. In our pipeline, different choices

TABLE 2
Techniques for highlighting text and visual elements.

highlight text	highlight visual elements	adopted in our method
Font Color	Fill Color	Yes
Background Color	Background Color	Yes
Underline	Stroke	Yes
Font Size	Size	-
Font Style	-	-
Font Weight	Stroke Width	Yes
Rectangular Border	Stroke	-
Spaced Out Font	-	-
Text Shadow	Shadow	-
Font Family	-	-
Capitalization	-	-
Strike Through	-	-
Blinking	Blinking	-

are provided for linking text and visualizations, with a focus on highlighting the text mentions and the referenced visual elements.

Subtask 1: Formatting Style. To establish visual linking between text and visual elements, we first format their visual styles. By default, extracted mentions are highlighted via dashed underline for direct mentions and data operations and grey background color for filter expressions. We used dashed underline to hint that the key phrases are automatically detected and have not been verified. When users interact with the text (e.g., brush a sentence), our system highlights the referenced visual elements via applying borders to them and letting other visual elements fade out. Besides, gray brush boxes are added as the background for filter operations. However, such results may not be desirable. Users may want to change the highlighting method to control readers' attention [35], [36]. Thus, more choices should be provided for users to edit the highlighting and the interaction mode of text and visualizations.

For facilitating the visual linking, we explored different options for highlighting the referenced visual element. The first column in Table 2 shows the common text highlighting methods summarized by Strobelt et al. [37] and we try to align each of them to a method for highlighting visual elements. Among them, font style, space out font, font family, capitalization, and strike through are specific to text and cannot be directly applied to visual elements. Besides, we exclude size, shadow, and blinking for different reasons: font and visual sizes are usually predetermined [38]; shadow may cause blurred vision in text and is not obvious in visualizations; blinking involves motion, which may be disturbing and cannot be used in static scenarios. Users can specify the style of textual references and their corresponding visual elements for facilitating the intuitiveness of TVLs.

Subtask 2: Specifying Integration. In addition to formatting the visual style of text and visualizations, a user could specify the integration of text and visualizations to craft different types of text-visualization linking. The integration can be achieved in different ways, and we provide basic solutions which integrate text and visualizations via interaction and spatial proximity. Users are allowed to specify the text segment to be linked and set the trigger mode (e.g., hover and click). Moreover, users can embed a visualization in a word-sized style alongside the text mention to it or add the text into the visualization as an annotation, while highlighting the referenced visual elements. In our work, the text to be linked and the linkage location (the segment of text can be interacted, the placement of word-sized visualization in text, and the position of text in visualizations) are manually specified. The techniques for identifying the insights and the linkage location in text to extend our pipeline are left to future work.

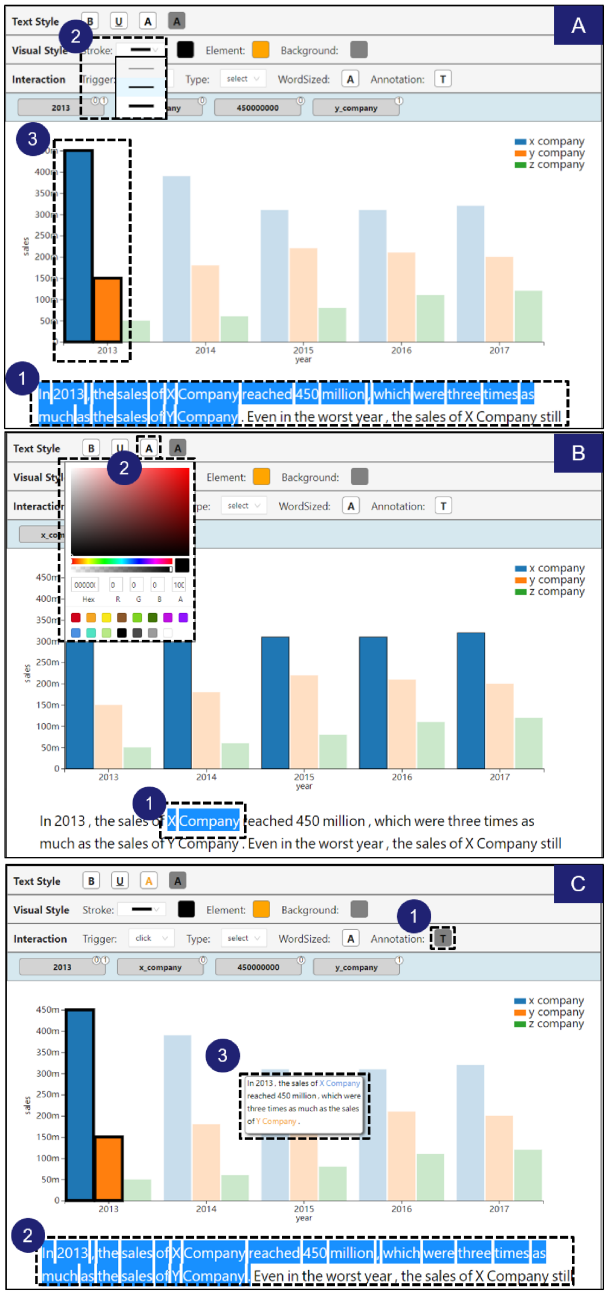

Fig. 7. An example for crafting annotated visualizations. After the previous steps in text parsing, a user resets the stroke thickness of the visual elements (A1-A3) and sets the color of the phrase (B1-B2). Finally, he embeds the text into the chart as the annotation (C1-C3).

Use Case. After regrouping the mentions, Bob wants to embed the sentence into the bar chart as an annotation (Stage IV). To make the key information more explicit, he revises the styles of the text and the visual elements. He brushes the sentence (Figure 7 A1) and sets the stroke width (Figure 7 A2) of the visual elements (Figure 7 A3). In addition, he brushes "X Company" (Figure 7 B1) and changes the font color to blue via the font color component (Figure 7 B2) to highlight these two phrases. In the same way, he changes the font color of "Y Company" to orange. After that, he presses the "Annotation" button (Figure 7 C1) and brushes the sentence (Figure 7 C2) again to embed the sentence into the bar chart (Figure 7 C3).

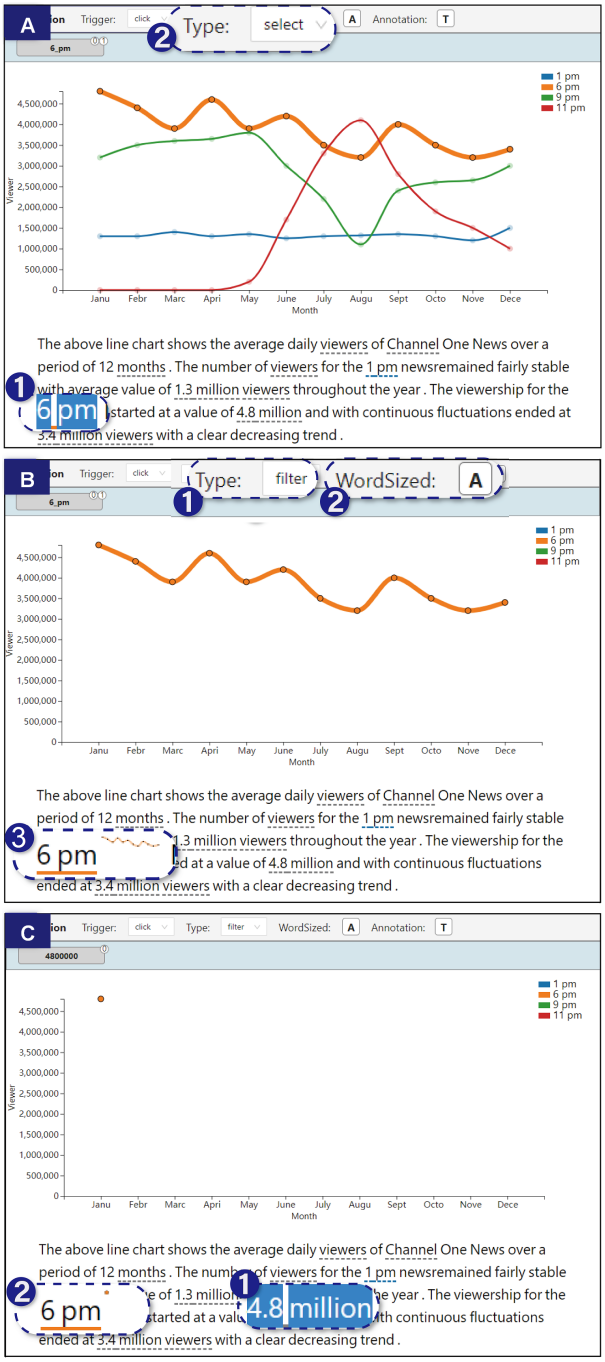

Fig. 8. A demo for crafting an online document with interactive word-sized visualization. After the initial steps of visualization specification and text parsing, we can select the keyword (A1), change the interaction type from "select" (A2) to "filter" (B1), and embed a word-sized visualization (B2) into the text (B3). Then, we brush the keyword "4.8 million" (C1) and the word-sized visualization also changed (C2).

4.3 Implementation and Performance Analysis

The system is implemented as a REACT application with a Python server. The external libraries used in our systems include: pandas [39] for data transformation; Stanford's CoreNLP [32] for NLP tasks (POS tagging, stemming, NER, and dependency parsing); RDFLib [40] for simulating RDF databases and supporting SparQL; D3.js [41] for drawing visualizations and supporting interaction; Ant Design [42] for developing the user interface.

We specify the number of visual elements as n and the number of words in the text as m. The time complexity of generating KG is $O(n)$. The time complexity of mention detection is $O(mn)$ because scanning the words is $O(m)$ and find the triplets contain a word is $O(n)$. The complexity of entity grouping is no more than $O(m)$ and that of resolving a reference is $O(n)$. Thus, the complexity of entity mapping is no more than $O(mn)$ given that the number of references is no more than m. We cannot tell the complexity of NLP tasks solved by Stanford's CoreNLP. In our experiments, the time cost for NLP tasks is approximately 2 seconds per thousand words (with 3.20 GHz 8-Core Intel Core i7 and 32 GB memory). More analysis can be found in the supplemental materials.

5 Usage Scenarios

In the previous section, we present a complete usage scenario, in which a user (Bob) followed our four-stage pipeline to create an annotated bar chart [11]. In this section, we present three other usage scenarios of linking text and visualizations, which demonstrate the effectiveness of our method.

5.1 Crafting Online Documents with Interactive Word-Sized Visualizations

The first example is creating interactive word-sized visualizations in online documents [12]. After the visualization is generated and the text is parsed, Bob brushes "6 pm" (Figure 8 A1) and the corresponding line is highlighted in the chart. To make it clearer, he chooses "filter" (Figure 8 B1) as the interaction type, where "select" (Figure 8 A2) is set as the default. The changes of the chart are shown in Figure 8 B. Via pressing "WordSized" button (Figure 8 B2), he embeds the line chart into the text as a word-sized visualization ("6 pm ~~~ ", shown in Figure 8 B3). Then, he brushes "4.8 million" (Figure 8 C1)) and only a point is shown in the word-sized visualization (Figure 8 C2) since the interaction mode is "filter". After specifying the interaction, he could share the crafted document with the public.

Note that in our work, the line chart is customized to support linking. In line charts, different from bar charts and pie charts, data objects are bound to the circles rather than the paths. For the visualizations where data are not directly mapped to the visual elements (e.g., line charts and area charts), our pipeline should be further extended. We discuss this as a limitation in section 7.

5.2 Specifying Chart-Text Interactions in Interactive Documents

The second example is specifying chart-text interactions in an interactive document [13] which describes the relation between Covid deaths and vaccines. After generating a line chart and parsing the text (Figure 9 A), Bob notices that the model detected many keywords, some of which he does not care. He chooses to focus on those phrases (Figure 9 A1-A4) he wants to emphasize. After brushing these phrases, he ensures that they are correctly connected to the visual elements. Then, he applies underlines (Figure 9 A5) to each of them and sets the trigger mode to "click" (Figure 9 A6). After that, he publishes this document. In the published document, the other detected keywords are omitted since Bob does not specify their linking. When readers click the phrases (Figure 9 B1-B4), the corresponding visual elements are highlighted in the chart (Figure 9 C1-C4). Note that in this example, we assume that Bob added meaningful grouping labels (e.g., "devastating

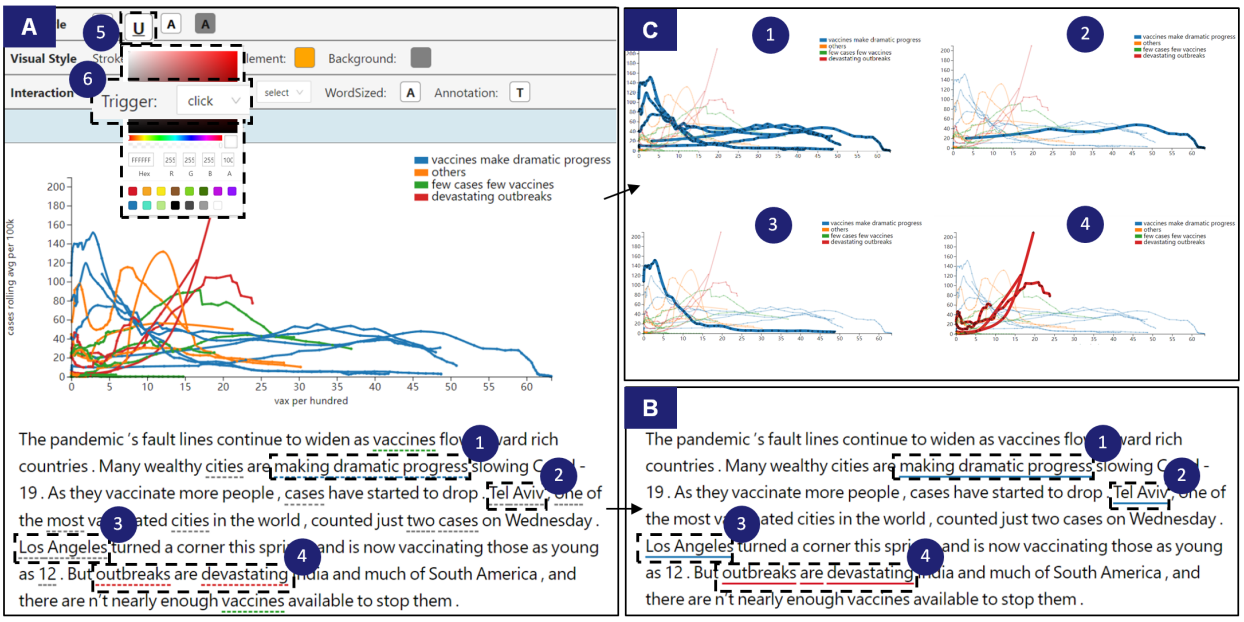

Fig. 9. An example for specifying chart-text interactions in an interactive document. First, a data analyst specifies the links between the chart and four phrases (A1-A4) via editing text style (A5) and trigger mode (A6) and publishes the document. Then, readers can interact with these phrases (B1-B4) to see the emphasized information in the chart (C1-C4).

outbreaks") during data preprocessing so that our model could recognize those keywords. The behavior or adding labels helps creating semantically meaningful legend in visualizations and can be found in a variety of data analysis projects. However, the grouping labels can be meaningless in some cases. In such cases, our backend model is unable to recognize the keywords and map them to the labels since it does not have the external knowledge (as discussed in **Capability of Natural Language Understanding** in section 7), and users need to interact with the system as described in Section 4.2.2 (Fig. 5A) to complete the task of mention detection.

6 Evaluation

In this section, we first present a user study to test the usability of our method. Then, we report the accuracy of the model and discuss the distribution of the errors and the failure conditions.

6.1 User Study

To evaluate the effectiveness of our pipeline in helping users craft TVLs, we carried out a user study with two types of tasks: (1) revising the errors of the model in text processing (Stage II-III), and (2) visually linking text and visualizations (Stage IV) when there is no error in the previous stages. Since the most related works [6], [7], [11] are not available, we cannot employ an appropriate baseline to conduct a comparative study. We focus more on the difficulties our users encountered when using our prototype system and the feedbacks provided in the post-experiment interview. The experiments are conducted on a desktop Windows 10 machine with a 27-inch monitor (3840×2160 resolution).

6.1.1 Participants and Experiment Design

Participants. We recruited 8 volunteers (4 males and 4 females, aged 23-25, and all reported normal or correct-to-normal vision) from a data visualization course. The participants were post-graduate students in data science (4) and artificial intelligence (4).

TABLE 3
Tasks

t1	Revise the wrongly detected mention.
t2	Revise the wrongly inferred attribute.
t3	Regroup the detected mentions.
t4	Create an annotated chart.
t5	Embed a word-sized visualization into the text.

To ensure the participants are able to assess our system, we only recruited those who had recent (in the past half year) experience in crafting analytical reports or presentations which incorporate text and visualizations. They all reported being skilled in using GUI based tools (e.g., Excel) to generate charts. Each of the participants received a 5$ gift card as a reward.

Tasks and Materials. To help the participants better assess our system, we designed 5 tasks (Table 3) based on the key interactions in our pipeline. Among t1-t3, participants needed to check the output of the model and revise the errors made by our model while the tasks involve no visual formatting and integration. The entailed interactions are illustrated in Figure 5. t4 and t5 require the participants to replicate the previous demos (Figure 2 IV 3 and Figure 8 C) and there is no error in mention detection and entity mapping. The materials for training and formal testing, including the tables and the text, were the same with Figure 2 and Figure 8. For creating the tasks, we manually revised the result of the text parsing. We did not employ more charts and the failure cases from the model experiment since through a preliminary test we found the diversity and the complexity of charts would cause unnecessary difficulty in comprehending the chart.

Procedure. At the beginning, we introduced the concept of linking text and visualizations and the workflow of our system to the participants. Then, they would get accustomed to the manipulation of the system under our instructions. After they got prepared, they were required to complete the tasks in Table 3. As they finished all the tasks, we encouraged them to test our system with different

TABLE 4
Questions

Q1	The system is easy to learn.
Q2	The system is easy to use.
Q3	The interactions are helpful in handling the errors of the automatic model.
Q4	The interactions are helpful in crafting text-visualization linking.

datasets and their own text. Then, they were asked to finish a questionnaire (Table 4), which is a seven-point Likert scale (1 - strongly disagree, 7 - strongly agree). Finally, we would make an interview with each of them. The time of the entire study was about 30 minutes for each participant.

6.1.2 Result

Generally, the participants were able to complete the tasks. Given that the difficulties of completing the tasks largely originated from comprehending the relationship between text and charts, we did not make a formal analysis of the completion time and the accuracy of each task. Instead, we focused on our observations, the questionnaire ratings, and the user feedbacks.

Overall, the users agreed (gave an average rating that approximates 6) that our system is easy to use (Q2, Mean = 6.25, SD = 0.46) and that the interactions are helpful in handling the errors of the automatic model (Q3, Mean = 6.25, SD = 1.04) and crafting text-visualization linking (Q4, Mean = 6, SD = 0.76). Among the questions, the average rating on ease of learning (Q1, Mean = 5.5, SD = 0.76) was the lowest.

6.1.3 User Feedback

Learning Cost. Overall, these interactions are not difficult to learn, except for grouping mentions (Q1). According to our observation, the majority (6 out of 8) of the participants paid more effort to grasp grouping mentions compared with other interactions, which may partly account for the comparatively low rating on Q1. The learning cost is mainly in understanding the concept rather than in learning the interaction. Thus, we need to explore a way to help users understand this task more easily through an user-friendly expression (such as "combining key phrases").

System Intelligence. One user gave 4 points on Q3. He mentioned that manually setting aliases of the keywords was not intelligent and there should be an inferencing model to help him achieve this goal. Besides, the users expressed their demands on the intelligent functions of the system, such as style recommendation and creating multiple links at the same time. We noticed that to meet these requirements, our methods still have room for improvement in terms of system intelligence.

Linking Format. The two users who gave 5 points on Q4 both mentioned the concerns about the use of word-sized visualizations. They mentioned that they had never seen word-sized visualizations embedded in the text before and had questions about the effectiveness of their usage. Word-sized visualization is not a common form for general users, so they may have hesitation. To address this issue, we have two considerations. On the one hand, we need to provide more guidance for the ways of linking text and visualization that users may not be familiar with, in order to eliminate their hesitation. On the other hand, we can hide features that general users do not understand, such as storing them in a secondary menu, in order to avoid confusion caused by these features.

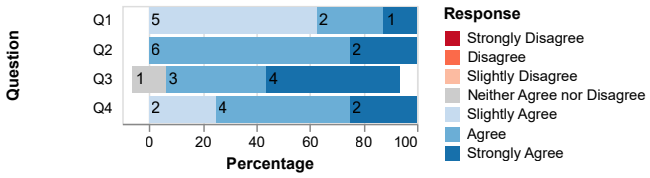

Fig. 10. The ratings on the questions.

6.2 Model Evaluation

We have conducted two preliminary tests to evaluate our automatic model. To make a direct comparison between our work and Kori [7], we used the Kori dataset [1] to test our method in the first test. However, this dataset is specifically designed to evaluate the performance of reference detection. It does not encompass aspects of reference mapping, which differs from our target of evaluation. In addition, the available part of this dataset mainly consists of direct mentions (point references), with few filter expressions (interval references) or grouped mentions included. To complement this, we conducted the second test, where we constructed another dataset based on the dataset presented by Kim et al. [14] to test our model. We chose this dataset since the chart specifications and data are available and the questions and explanations contain references to the visual elements in the chart. Please note that, as visualization practitioners, our primary focus is on the collaboration between human users and automatic models rather than solely presenting an automated method. The model evaluation aims to provide preliminary insights into the capability of the model, rather than demonstrating we outperform the existing models.

6.2.1 Test 1: Reference Detection

Dataset. The Kori dataset consists of two parts: text-chart pairs selected from Kong et al.'s collection [1] and annotated by Kori's authors (Part 1) and those created by Kori's authors (Part 2). While Part 1 is completely available, Part 2 is unavailable to us for the following reasons: 1) it does not include the corresponding data (e.g., CSV files) for generating the charts; 2) many of the chart specifications are not supported by us due to the chart types or unique Vega-Lite syntax. Thus, we only used Part 1.

Results. On Part 1 of the Kori dataset, our model detects 100 correct references (out of 169 references) and generates 17 false references, achieving an accuracy of 0.59 and a recall of 0.85. Our F1 score (0.69) is higher than the average of Kori's (0.43) on this part. After reviewing Kori's paper, we speculate that there might be two reasons for our better outcomes: 1) Our approach provides better support for numbers (as mentioned in Kori's paper, their method is based on FastText [43], which has problems in handling numbers); and 2) Kori utilizes a vector similarity technique, which excels at identifying synonyms but may detect semantically similar yet irrelevant terms, thereby lowering the precision. Initially, we also considered adopting word vector methods. However, during practical testing, we found such approaches to yield unsatisfactory results. Thus, we ultimately decided against using word vectors.

6.2.2 Test 2: Associating Text with Visual Elements

Dataset. The dataset presented by Kim et al. [14] contains 52 charts along with 629 human-generated questions (and answers) and 748 human-generated explanations. Among the 52 chart specifications,

1. https://dwr.bc.edu/kori/

16 were left out since they contain data transformation (e.g., string operations and conditional operators) we do not support. We excluded the questions and explanations on these charts and randomly sampled 100 questions and 100 explanations from the rest. Based on the prototype system, we generated the charts according to the specifications, took each question and each explanation as the input, and selected the whole sentence to highlight the groups of key phrases. We invited 2 experts (PhD candidates who have at least one first author TVCG paper) to judge whether the referenced entities in the text are correctly highlighted. They were asked to be strict about the result, which means irrelevant visual elements should not be highlighted and important information in the text should not be missed. To avoid induction, we hid the hints of the key phrases in the text.

Result. Among the samples, 26 cases (6 questions and 20 explanations) were excluded as the experts thought that these cases contain no valid reference to the charts. Our model correctly highlighted 69.0% (120 out of 174) sentences, with 62.8% (59 out of 94) accuracy for questions and 76.3% (61 out of 80) for the explanations. Through a preliminary analysis of the cases, we find the imbalance in the accuracy of the questions and that of the explanations is mainly due to two points. First, the key information in the explanations tends to be simpler than that in the questions. For example, the questions sometimes involve calculations we do not support (e.g., "Which variety has the largest sum of yield?") while the explanations involve fewer such expressions. Second, there seems to be a higher ratio of valid explanations provided with simple charts (e.g., simple bar chart) compared to those with more complex ones (e.g., grouped bar chart). The reason might be that people make valid explanations more easily with simpler charts.

Among the failure cases, 85.2% (46 out of 54) fail in mention detection (Stage II) and only 14.8% (8 out of 54) fail in reference mapping (Stage III). The errors mainly lay in mention detection, which is consistent with our knowledge and experiences. On the one hand, effectively extracting keywords from the text and understanding their meaning in the context is a challenging task. On the other hand, in most cases it involves neither complex relations of the key phrases nor complex relationships between the text references and the visual elements.

6.2.3 Failure Conditions

According to the result of this experiment and our experience in developing the model, we distinguish three conditions in which our model may fail:

Human knowledge is required to resolve ambiguities or to recognize key phrases. In some cases, the text may contain keywords which are not included in the data. For example, in the sentence "How many countries in Asia will have their economy improved based on majority votes", our model cannot understand "countries in Asia" since it does not have the knowledge about which countries are in Asia.

Challenging NLP tasks are involved in processing the text. When the dependency structure of the sentence is complex, which involves multiple clauses, the dependency parsing module may fail. Besides, we do not implement coreference resolution since we found there too many errors in the coreference resolution functions of the NLP tools we have tested. Thus, it cannot recognize those pronouns.

Further calculations are required to resolve the relationship between text and visualizations. Our model does not support finding VEs through combining different calculations, such as finding which company has the **largest sum** of sales.

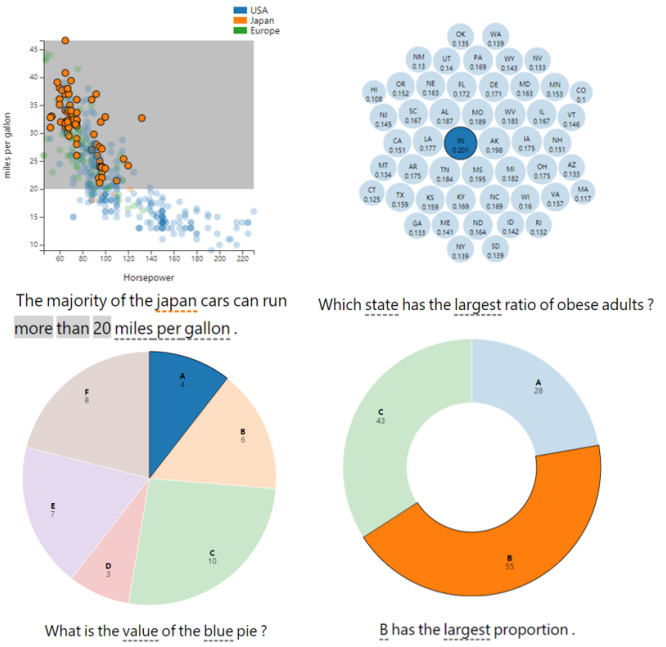

Fig. 11. The examples of our method in supporting scatter plot, bubble chart, pie chart, and donut chart.

Among the above-mentioned failure conditions, the third one is beyond the scope of the present work. It requires specific mechanisms to map natural language to computational processes and we are currently unaware of the possible solutions. The first and second conditions are expected to be solved by human users via the interactions we provided. We regard these failure conditions as a limitation of the natural language understanding capability. The pathways to improving it is further discussed in the third limitation (**Capability of Natural Language Understanding**) in Section 7.1.

7 Discussion

Through the replication of the key functions in the previous works and the evaluation based on the prototype system, we demonstrate the usability and effectiveness of our method. Our method supports different TVL scenarios. Meanwhile, we allow users to modify our model's errors, thereby enhancing its usability when it makes mistakes. Existing works only allow users manually link visualizations and text or let it done automatically by their models, but do not enable users to modify the models' outputs. Compared to these works, our work allows users to modify the results generated by our model. When the results of our model are partially usable, users do not need to manually build TVLs from scratch. Instead, they can archive their goals by modifying the current results, which enhances the efficiency of crafting TVLs. Through the user study, we found that users were satisfied with the interactions provided for handling the errors and crafting TVLs.

7.1 Limitations

Although the method can be applied in various applications and the automatic model achieves reasonable performance, it needs to be improved in different aspects. We discuss the limitations and potential pathways for improvement as follows:

Support for Different Types of Visualizations and Data. Presently, our pipeline works with tabular data and several basic

charts (see Figure 11 for examples of supported chart types other than bar charts and line charts). Our model binds visual elements with data objects and stores them in KGs so that the visual representations can be referenced via data values. It may not work well with those charts where the mapping between the visual elements and the data objects is more complex (e.g., area charts). More appropriate representations should be designed for supporting different types of visualizations.

Support for Visualization Specification. In our work, we focus on linking text and visualizations and the tasks for specifying visualizations are suggested to be better supported by external models. Manually specifying visualizations could be inefficient, especially there a set of charts to be created [44]. There have been different techniques for facilitating specifying visualizations (e.g., visualization recommendation [45] and natural language interface (NLI) [31]) to support this task. To our knowledge, a promising idea is combining NLIs for specifying visualizations according to the context of the text. To verify this, we hope to explore the possibilities in the future.

Capability of Natural Language Understanding. Although our model achieves acceptable performance on the tested text samples, we find it works poorly when: it lacks the necessary knowledge; it needs to deal with challenging NLP tasks; further calculations are required. To address the issue that our model lacks the necessary knowledge for entity detection, we have considered connecting our KG to knowledge bases such as Wikipedia. However, incorporating external knowledge is not easy since there are so many similar entities and resolving ambiguities via entity resolution is also challenging. Besides, we employed Stanford's CoreNLP [32] and applied heuristics for parsing text, which is ineffective in dealing with complex sentences that involves clauses, coreferences, and further calculations. In recent years, large language models (LLMs) have been used to solve a series of visualization-related tasks [46], [47], [48], [49]. It would be interesting to explore using LLMs to enhance the NLP capability of our method.

Design Space of Text-Visualization Linking. In our pipeline, we allow users to edit the visual style and the integration of text and visualization. We only explore a small area of the design space of text-visualization linking and provide baseline choices for highlighting the referenced visual elements. The effectiveness of the highlighting techniques for text and visualizations has not been well tested yet. Further study should be conducted to explore the design space for linking text and visualizations and compare the effectiveness of different methods.

8 Conclusion

In this work, we propose a semi-automatic pipeline for linking text and visualizations. To resolve the relationship between text and visualizations, we employ a model to transform a visualization and the underlining data into a contextual knowledge graph. Based on the knowledge graph, the model automatically maps key phrases to the visual elements, while allowing users to cooperate with the model to revise the TVLs. We demonstrate the usability of our method via several usage scenarios and a user experiment.

Acknowledgment

The work was supported by Key "Pioneer" R&D Projects of Zhejiang Province (2023C01120), NSFC (U22A2032), and the Collaborative Innovation Center of Artificial Intelligence by MOE and Zhejiang Provincial Government (ZJU).

References

[1] N. Kong, M. A. Hearst, and M. Agrawala, "Extracting references between text and charts via crowdsourcing," in *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, 2014.

[2] J. Hullman and N. Diakopoulos, "Visualization Rhetoric: Framing effects in narrative visualization," *IEEE TVCG*, 2011.

[3] F. Beck and D. Weiskopf, "Word-sized graphics for scientific texts," *IEEE TVCG*, 2017.

[4] E. Segel and J. Heer, "Narrative Visualization: Telling stories with data," *IEEE TVCG*, 2010.

[5] D. H. Kim, E. Hoque, J. Kim, and M. Agrawala, "Facilitating document reading by linking text and tables," in *Proc. of the Annual ACM Symposium on User Interface Software and Technology*, 2018.

[6] S. K. Badam, Z. Liu, and N. Elmqvist, "Elastic Documents: Coupling text and tables through contextual visualizations for enhanced document reading," *IEEE TVCG*, 2018.

[7] S. Latif, Z. Zhou, Y. Kim, F. Beck, and N. W. Kim, "Kori: Interactive synthesis of text and charts in data documents," *IEEE TVCG*, 2021.

[8] J. Hullman, N. Diakopoulos, and E. Adar, "Contextifier: Automatic generation of annotated stock visualizations," in *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, 2013.

[9] T. Gao, J. R. Hullman, E. Adar, B. Hecht, and N. Diakopoulos, "NewsViews: An automated pipeline for creating custom geovisualizations for news," in *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, 2014.

[10] C. Bryan, K.-L. Ma, and J. Woodring, "Temporal Summary Images: An approach to narrative visualization via interactive annotation generation and placement," *IEEE TVCG*, 2016.

[11] C. Lai, Z. Lin, R. Jiang, Y. Han, C. Liu, and X. Yuan, "Automatic annotation synchronizing with textual description for visualization," in *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, 2020.

[12] S. Latif, D. Liu, and F. Beck, "Exploring interactive linking between text and visualization." in *EuroVis (Short Papers)*, 2018.

[13] Cai, Weiyi and Gamio, Lazaro and Leatherby, Lauren and McCann, Allison, "The pandemic has split in two," 2021, [Accessed 12-December-2021]. [Online]. Available: https://www.nytimes.com/interactive/2021/05/15/world/covid-inequality-vaccines.html

[14] D. H. Kim, E. Hoque, and M. Agrawala, "Answering questions about charts and generating visual explanations," in *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, 2020.

[15] N. Sultanum, F. Chevalier, Z. Bylinskii, and Z. Liu, "Leveraging text-chart links to support authoring of data-driven articles with vizflow," in *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, 2021.

[16] A. Srinivasan, S. M. Drucker, A. Endert, and J. Stasko, "Augmenting visualizations with interactive data facts to facilitate interpretation and communication," *IEEE TVCG*, 2018.

[17] P. Goffin, W. Willett, J.-D. Fekete, and P. Isenberg, "Design considerations for enhancing word-scale visualizations with interaction," in *Posters of the IEEE Conf. on Information Visualization*, 2015.

[18] S. Latif, K. Su, and F. Beck, "Authoring combined textual and visual descriptions of graph data." in *EuroVis (Short Papers)*, 2019.

[19] R. Metoyer, Q. Zhi, B. Janczuk, and W. Scheirer, "Coupling story to visualization: Using textual analysis as a bridge between data and interpretation," in *International Conf. on Intelligent User Interfaces*, 2018.

[20] K. Thellmann, M. Galkin, F. Orlandi, and S. Auer, "LinkDaViz–automatic binding of linked data to visualizations," in *International Semantic Web Conference*. Springer, 2015.

[21] B. Dumas, T. Broché, L. Hoste, and B. Signer, "Vidax: An interactive semantic data visualisation and exploration tool," in *Proc. of the International Working Conference on Advanced Visual Interfaces*, 2012.

[22] F. Haag and T. Ertl, "Filter Dials: Combine filter criteria, see how much data is available," in *Proc. of the International Working Conf. on Advanced Visual Interfaces*, 2014.

[23] S. Xia, N. Anzum, S. Salihoglu, and J. Zhao, "KTabulator: Interactive ad hoc table creation using knowledge graphs," in *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, 2021.

[24] B. Chan, L. Wu, J. Talbot, M. Cammarano, and P. Hanrahan, "Vispedia: Interactive visual exploration of wikipedia data via search-based integration," *IEEE TVCG*, 2008.

[25] Y. Liu, Y. Ma, Y. Zhang, R. Yu, Z. Zhang, Y. Meng, and Z. Zhou, "Interactive optimization of relation extraction via knowledge graph representation learning," *Journal of Visualization*, 2024.

[26] D. Cashman, S. Xu, S. Das, F. Heimerl, C. Liu, S. R. Humayoun, M. Gleicher, A. Endert, and R. Chang, "CAVA: A visual analytics system for exploratory columnar data augmentation using knowledge graphs," *IEEE TVCG*, 2020.

[27] H. Li, Y. Wang, S. Zhang, Y. Song, and H. Qu, "KG4Vis: A knowledge graph-based approach for visualization recommendation," *IEEE TVCG*, 2022.
[28] J. Heer, "Agency Plus Automation: Designing artificial intelligence into interactive systems," *Proc. of the National Academy of Sciences*, 2019.
[29] G. Antoniou and F. Van Harmelen, *A semantic web primer*. MIT press, 2004.
[30] T. Gao, M. Dontcheva, E. Adar, Z. Liu, and K. G. Karahalios, "Datatone: Managing ambiguity in natural language interfaces for data visualization," in *Proc. of the Annual ACM Symposium on User Interface Software and Technology*, 2015.
[31] A. Narechania, A. Srinivasan, and J. Stasko, "NL4DV: A toolkit for generating analytic specifications for data visualization from natural language queries," *IEEE TVCG*, 2020.
[32] "CoreNLP." [Online]. Available: https://stanfordnlp.github.io/CoreNLP/index.html
[33] E. Hoque, V. Setlur, M. Tory, and I. Dykeman, "Applying pragmatics principles for interaction with visual analytics," *IEEE TVCG*, 2017.
[34] A. Srinivasan and J. Stasko, "Orko: Facilitating multimodal interaction for visual exploration and analysis of networks," *IEEE TVCG*, 2017.
[35] S. He, Y. Chen, Y. Xia, Y. Li, H.-N. Liang, and L. Yu, "Visual Harmony: text-visual interplay in circular infographics," *Journal of Visualization*, 2024.
[36] N. Chotisarn, S. Gulyanon, T. Zhang, and W. Chen, "VISHIEN-MAAT: Scrollytelling visualization design for explaining siamese neural network concept to non-technical users," *Visual Informatics*, vol. 7, no. 1, 2023.
[37] H. Strobelt, D. Oelke, B. C. Kwon, T. Schreck, and H. Pfister, "Guidelines for effective usage of text highlighting techniques," *IEEE TVCG*, 2015.
[38] S. M. Rebelo, T. Martins, D. Ferreira, and A. Rebelo, "Towards the automation of book typesetting," *Visual Informatics*, 2023.
[39] "pandas." [Online]. Available: https://pandas.pydata.org/
[40] "RDFLib." [Online]. Available: https://rdflib.readthedocs.io/en/stable/
[41] "D3.js." [Online]. Available: https://d3js.org/
[42] "Ant Design." [Online]. Available: https://ant.design/
[43] T. Mikolov, É. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in pre-training distributed word representations," in *Proc. of the International Conf. on Language Resources and Evaluation*, 2018.
[44] P. Soni, C. de Runz, F. Bouali, and G. Venturini, "A survey on automatic dashboard recommendation systems," *Visual Informatics*, 2024.
[45] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer, "Voyager: Exploratory analysis via faceted browsing of visualization recommendations," *IEEE TVCG*, 2015.
[46] Y. Ye, J. Hao, Y. Hou, Z. Wang, S. Xiao, Y. Luo, and W. Zeng, "Generative AI for Visualization: State of the art and future directions," *Visual Informatics*, 2024.
[47] Y. Tian, W. Cui, D. Deng, X. Yi, Y. Yang, H. Zhang, and Y. Wu, "ChartGPT: Leveraging LLMs to generate charts from abstract natural language," *IEEE TVCG*, 2024.
[48] L. Shen, Y. Zhang, H. Zhang, and Y. Wang, "Data Player: Automatic generation of data videos with narration-animation interplay," *IEEE TVCG*, 2023.
[49] Y. Zhao, Y. Zhang, Y. Zhang, X. Zhao, J. Wang, Z. Shao, C. Turkay, and S. Chen, "LEVA: Using large language models to enhance visual analytics," *IEEE TVCG*, 2024.

Xiwen Cai is currently a Ph.D. candidate in the State Key Lab of CAD&CG, Zhejiang University in China. He received his B.Sc. degree in Psychology and M.Sc degree in Computer Science from Zhejiang University. His research interests include human computer interaction and visual analytics.

Di Weng Dr. Di Weng is a ZJU100 Young Professor at School of Software Technology, Zhejiang University. His main research interest lies in information visualization and visual analytics, focusing on interactive data transformation and spatiotemporal data analysis. He received his Ph.D. degree in Computer Science from State Key Lab of CAD&CG, Zhejiang University. Prior to his current position, Dr. Weng was a researcher at Microsoft Research Asia from 2022 to 2023. For more information, please visit https://dwe.ng.

Taotao Fu was a master student in School of Software Technology, Zhejiang University in China. He received his B.Sc. degree in Computer Science from Zhejiang University City College. His research interests mainly include big data and data mining.

Dr. Siwei Fu is a tenure-track Associate Professor in the School of Management, Zhejiang Lab. His main research interests include visual analytics, human-AI collaborative decision-making, and natural language interfaces. He received his Ph.D. degree in Computer Science and Engineering from the Hong Kong University of Science and Technology. For more information, please visit https://fusiwei339.bitbucket.io.

Yongheng Wang Yongheng Wang is a research specialist in Big Data Intelligence Research center of Zhejiang Lab. His research interests include Big data analytics, machine learning and intelligent decision making. His current research is on intelligent interactive data analysis, with emphasis on the intelligent algorithms to support real-time response and knowledge-based analysis.

Dr. Yingcai Wu is a Professor at the State Key Lab of CAD&CG, Zhejiang University. His main research interests are information visualization and visual analytics, with focuses on urban computing, sports science, immersive visualization, and social media analysis. He received his Ph.D. degree in Computer Science from the Hong Kong University of Science and Technology. Prior to his current position, Dr. Wu was a postdoctoral researcher in the University of California, Davis from 2010 to 2012, and a researcher in Microsoft Research Asia from 2012 to 2015. For more information, please visit http://www.ycwu.org.