CrossData: Leveraging Text-Data Connections for Authoring Data Documents

Zhutian Chen
University of California San Diego
La Jolla, CA, USA
zhutian@ucsd.edu

Haijun Xia
University of California San Diego
La Jolla, CA, USA
haijunxia@ucsd.edu

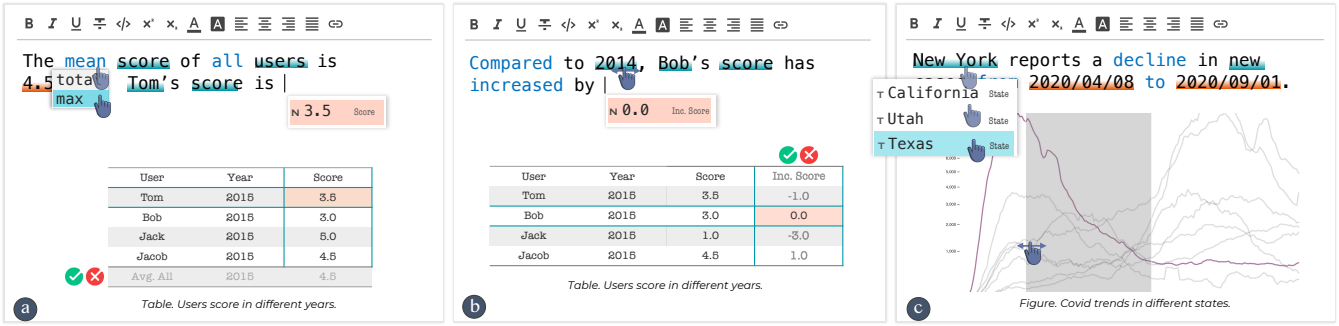

Figure 1: CrossData leverages text-data connections to enable users to efficiently retrieve (a), compute (b), interactively explore data (a, b, c), and adjust tables (a, b) and charts (c) during their writing processes, while also automatically maintaining data consistency between their text, data, tables, and charts.

ABSTRACT

Data documents play a central role in recording, presenting, and disseminating data. Despite the proliferation of applications and systems designed to support the analysis, visualization, and communication of data, writing data documents remains a laborious process, requiring a constant back-and-forth between data processing and writing tools. Interviews with eight professionals revealed that their workflows contained numerous tedious, repetitive, and error-prone operations. The key issue that we identified is the lack of persistent connection between text and data. Thus, we developed CrossData, a prototype that treats text-data connections as persistent, interactive, first-class objects. By automatically identifying, establishing, and leveraging text-data connections, CrossData enables rich interactions to assist in the authoring of data documents. An expert evaluation with eight users demonstrated the usefulness of CrossData, showing that it not only reduced the manual effort in writing data documents but also opened new possibilities to bridge the gap between data exploration and writing.

KEYWORDS

Language-oriented Authoring, Text-based Editing, Natural Language Processing, Data Document, Interactive Article

CHI '22, April 29-May 5, 2022, New Orleans, LA, USA

ACM ISBN 978-1-4503-9157-3/22/04.
https://doi.org/10.1145/3491102.3517485

ACM Reference Format:
Zhutian Chen and Haijun Xia. 2022. CrossData: Leveraging Text-Data Connections for Authoring Data Documents. In *CHI Conference on Human Factors in Computing Systems (CHI '22), April 29-May 5, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3491102.3517485

1 INTRODUCTION

Data documents employ text, tables, and visualizations to report findings from data analyses and present data-rich narratives, and are an indispensable component of every domain that uses data, such as scientific research, finance, public health, education, and journalism. As our world becomes increasingly data-driven, there has been a surge in the variety of data documents (e.g., data-rich documents [4], data-driven articles [56], and interactive articles [14]), as well as in the research that has sought to support the authoring and consumption experiences of data documents.

However, despite the proliferation of applications and systems that have been designed to support data analyses, visualization, and communication, authoring data documents remains a laborious task. During a typical workflow, a user will explore their data by performing data analysis operations (e.g., filtering, sorting, creating tables and charts, etc.) to generate insights using data processing tools and then they will synthesize the insights into a document using a word processing application. During this process, the user will need to switch back and forth between applications to take notes about the insights they discover, retrieve data from data processing tools and enter it into their document, as well as ensure that there is consistency between the data reported in their document and their underlying dataset. As the user's underlying data is updated or they iteratively refine, explore, and change their insights, the user will need to re-analyze their data, refine the corresponding

tables and charts, and carefully identify and revise any out of date data in their document. This workflow is not only error-prone, but also requires significant manual and cognitive effort.

The key reason that such tedious and ineffective workflows exist is due to the lack of persistent bindings or connections that exist between the text in data documents and the data in datasets. Most commercial applications do not support the creation or maintenance of text-data connections, instead requiring that users maintain these connections in their mind and perform tedious, manual updates to their documents and data. The state-of-the-art research systems that have been created to support the authoring of dynamic and interactive data documents all require the use of programming to specify data bindings [14, 33], thus posing a higher barrier to entry for novice users. In addition, for each data connection, a user will need to write and update source code to specify and maintain any connections, resulting in tedious workflows, especially for data documents that contain a large amount of data.

One observation, however, is that the data reported in data documents is naturally embedded with highly descriptive text. These natural embeddings present an interesting opportunity to solve this text-data connection problem in that they may enable systems to infer text-data connections directly from text during one's writing process. This work thus explores how language-oriented data bindings could be derived from the latent connections that exist between text and data. To systematically explore how language-oriented text-data connections can assist in the authoring of data documents, this research sought to understand the general workflow, pain points, and challenges that exist when authoring data documents by conducting a formative study with eight professionals from different domains who write data documents extensively as part of their daily work. Informed by the findings from this study, we then developed CrossData (Figure 1), a research prototype that explores the potential of extracting latent language-oriented data bindings that exist within highly descriptive text and reifying them as persistent, interactive, first-class objects [5, 19, 22, 62] to assist in the authoring of data documents.

CrossData utilizes a connection engine that automatically detects, establishes, and maintains text-data connections during the writing process by using state-of-the-art natural language processing (NLP) techniques. While writing text for their documents, CrossData enables users to efficiently retrieve, compute, explore data, and refine tables and charts using interactive interaction techniques that are enabled by the language-oriented data bindings that are identified and created. CrossData leverages these bindings to automatically ensure consistency and congruency between the text, data, tables, and charts. In addition, data documents written with CrossData automatically become interactive documents for readers, enabling them to have a dynamic, explorable reading experience. To assess the performance of the connection engine in extracting latent text-data connections, a technical evaluation was conducted. The results showed that the engine correctly constructed 88.8% of 529 text-data connections identified from 206 sentences, demonstrating its effectiveness. To assess the utility of language-oriented data bindings, an expert evaluation was conducted and demonstrated that CrossData's interaction techniques can significantly reduce the manual effort required while writing data documents and also enable fluid and enjoyable workflows. Feedback from experts also indicated that language-oriented authoring exposes new possibilities for data exploration and authoring.

This systematic exploration of language-oriented authoring for data documents thus contributes:

1. An understanding of the challenges that exist when authoring data documents today.
2. A language-oriented data binding approach that extracts latent text-data connections from written text.
3. A set of novel interaction techniques that enable users to efficiently author and iterate on data documents.
4. The CrossData prototype system, i.e., an implementation of language-oriented authoring for data documents, which was evaluated by experts along the dimensions of the usefulness and usability of the interaction techniques that the system supported.

2 RELATED WORK

As this research aims to leverage the connections that exist between highly descriptive text and data to ease the authoring of data documents, prior work on authoring data-driven content, linking text to other visual media, and natural language interfaces for data queries and visualization, are reviewed.

2.1 Authoring Data-driven Content

Significant research in HCI and data visualization has explored how to support the authoring of data-driven content, such as charts [12, 46], infographics [13, 63], data-driven comics [28], videos [3], and articles [56]. Within this research, bindings were created between the visual components and the underlying data so that the data-driven content could be updated whenever the data changed, and vice versa. This thus reduced the repetitive effort necessary to manually update content and enabled rich, dynamic interactive experiences.

There has been a proliferation of research systems that have assisted in the creation of data visualizations that have followed the principles of direct manipulation [5, 52] as alternatives to the template-based chart editing methods that lack customizability and the programming libraries that require significant expertise and are often cognitively demanding to use. For example, Data Illustrator [37], DataInk [63], and Lyra [49] enabled users to directly create a set of visual encodings, which could be applied to all the data points in a dataset to quickly generate data visualizations. Victor proposed a system that captured parameterized drawing steps, which could later be reused to generate an entire visualization [60]. Charticulator also allowed authors to interactively specify chart layouts and employed a constraint-based method to realize layouts [46].

Recent research has extended the concept of data-driven content to other media such as data-driven articles, which consist of text, charts, interactive equations, simulations, and so on. For example, Victor presented Explorable Explanations, a type of data-driven article where the numbers and equations reported in the text were bounded to the underlying data and computation models enabled readers to manipulate the author's assumptions and see the consequences [59]. Dragicevic et al. applied a similar idea to scientific

reports, enabling readers to explore the different analytical results of a study [17]. Computational notebooks (e.g., Jupyter [24], R Markdown [45]), an modern embodiments of Knuth's literate programming notion [30], also allowed users to integrate data with text, executable code, and visualizations to reproduce and share explorations. Creating such data-driven content, however, is tedious and time-consuming because, unlike data visualizations where users can easily configure a small set of visual encodings to create and adjust the entire visualization, each binding in a data-driven article often requires specific configurations with the underlying data. As a result, state-of-the-art systems designed to support authoring data-driven articles use programming languages and require users to manually configure each desired data-driven element. For example, Idyll, a markup language for web-based interactive documents, enabled users to bind data or reader events (e.g., page scrolling) to text, visualizations, and other elements in documents, thereby creating an interactive reading experience [14]. Computational notebooks require users to write code to manipulate and bind data to other content, while text is mainly used for explanatory descriptions alongside code to facilitate documentation.

Instead of requiring users to manually specify data-driven bindings using programming languages, CrossData infers and recommends connections that implicitly exist between text and data to the user during the writing process. Coupled with a set of novel interaction techniques that enable users to easily select and update text-data connections, CrossData not only significantly reduces the manual effort needed to create data documents, but also simultaneously enables an interactive reading experience for readers without any additional effort.

2.2 Linking Text to Other Visual Media

There has been significant research exploring how text can be leveraged and enhanced to facilitate both content consumption and creation processes. To facilitate data communication and help users efficiently synthesize information distributed across a data document, prior work has explored connecting text with other data representations (i.e., tables [4, 27] and charts [31, 33, 56]) to enhance reading experiences, using a variety of techniques including direct manipulation, mixed-initiative, crowdsourced, and fully automatic methods. For example, Sultanum et al. enabled users to specify desired links between text and charts and leveraged these text-chart links to adapt content to a range of layouts [56]. Latif et al. developed a mixed-initiative interface by leveraging NLP techniques to construct interactive references between text and charts [33]. Kong et al. developed an interactive document reading application that utilized crowdsourced links between text and charts to enable users to easily navigate from text to referred marks in a chart [31]. Kim et al. leveraged NLP techniques to connect text with corresponding cells in data tables within PDF documents to enhance reading experiences [27]. Recent advances in deep neural network have also led to a sequence of automatic methods to facilitate the reading of visualizations with text, such as visualization annotation [32], chart captioning [36], and chart question answering [25, 26].

Beyond linking text with different data representations, extensive research in NLP, computer vision, and machine learning has explored the automatic conversion of domain-specific descriptive text into visual content, such as 3D shapes [11] and scenes [10, 15], infographics [16], as well as short video clips [38], to help content creators. For example, the WordsEye system matched word semantics to the functional and spatial properties of 3D models to automatically convert text descriptions into 3D scenes [15]. Research in HCI has also leveraged the links between text and visual content to assist in the creation process. For example, Rubin et al. [47] and Troung et al. [57] leveraged the linear temporal properties that are common across text, audio, and video to assist in the editing of media clips. Perhaps the most closely related work to the present research is Crosspower [61], which leveraged desired correspondences between linguistic structures and graphical structures to enable users to flexibly and quickly create and manipulate graphical elements, as well as their layouts and animations. The present research also seeks to support content creation. However, it focuses on the domain of data documents, which resulted in a different set of interaction techniques to coherently address several challenges in users' workflows while authoring data documents.

2.3 Natural Language Interfaces for Data Queries and Visualization

Recent advances in NLP have renewed interest in natural language interfaces (NLIs) for data analysis. Compared to traditional data analysis systems, systems with NLIs enable users to interact with data by using questions and commands expressed via natural language rather than via interface actions or domain-specific languages (e.g., SQL), thereby lowering barriers for non-experts to access data [1]. These systems can be roughly divided into two categories based on if they support data queries or if they support the creation of, and interaction with, data visualizations.

Querying data through natural language has been extensively studied in the field of database systems. Many systems from this field adopted a parsing-based strategy [1, 44], with the goal of constructing SQL queries by identifying entities and their relationships in an input query. For example, ATHENA [48] parsed and mapped natural language queries to entities in an ontology generated automatically from a database and then translated the input query into SQL. Recently, machine learning-based methods have been gaining traction due to the success of deep learning [51, 58]. These methods use supervised neural networks to translate a natural language query to SQL. Seq2SQL [64], for example, used a deep reinforcement learning model to generate SQL based on an input query. To leverage the best of both methods, some systems (e.g., QUEST [6]) have utilized parsing- and learning-based methods as part of a multi-step pipeline.

NLIs for data visualizations can be seen as an extension of NLIs for databases, which enable users to visualize query results and interact with the generated visualizations. For example, a user can type "show me the medals for hockey and skating by country" to generate a visualization of this specific data. A key challenge when generating visualizations based on natural language is to resolve the ambiguities that exist in the query. DataTone [21], for example, proposed a mixed-initiative approach that enabled users to resolve ambiguities by interacting with ambiguity widgets. NL4DV [39] was a toolkit that took a tabular dataset and a query as input and returned a JSON specification of generated visualizations. Ambiguous results were then highlighted in the specification. In addition

to generating visualizations, researchers have also used natural language to interact with visualizations. For example, Eviza [50] enabled users to continually revise and interact with a visualization by asking questions. InChorus [54] supported multimodal input with both speech and touch to interact with visualizations. Recently, Srinivasan et al. [55] presented a dataset of visualization-oriented utterances collected from an online study, providing a benchmark of NLIs for visualization.

Overall, these NLI systems treated natural language and text as commands, so there were no persistent connections between the text and the data. While CrossData was built using similar NLP techniques, highly descriptive text was viewed as another representation of the underlying data so it was important to preserve the connections that existed between the text and data. These persistent connections were then leveraged to provide rich interactions that could be used during the writing process.

3 FORMATIVE STUDY WITH PROFESSIONALS

To better understand the general workflow, pain points, and best practices while writing data documents, a formative interview study was conducted.

3.1 Participants and Procedure

Eight professionals from various domains, including business services, e-commerce, accounting, banking, biomedical science, retail, and internet services were interviewed (4 female, age 27 – 30). Each had 3 – 7 years working in their current role and their responsibilities included exploring, analyzing, and reporting data. The interviews were conducted remotely using videotelephony and lasted between 45 to 60 minutes.

During the interviews, the professionals were asked to describe a recent, memorable experience while writing data documents, common pain points, and their solutions. They were also asked to share their documents and tools through screen sharing, if possible. The interview ended with a questionnaire to collect demographic information. Four pilot interviews with another 4 professionals were conducted beforehand to develop the study protocol.

Interviews were audio-recorded, transcribed, and analyzed using a reflexive thematic analysis [7]. The codes and themes were generated both inductively (i.e., bottom-up) and deductively (i.e., top-down), focusing on the workflow breakdowns, repetitive operations, and workarounds that occurred while writing data documents.

3.2 Findings and Discussion

The general process of producing data documents mainly included data exploration and writing. During the exploration stage, participants cleaned, processed, and explored their data with a concrete goal or question assigned to them by their manager. Excel was the most common tool used for this process (7/8). All participants said that when insights and findings were discovered within the data, they would "*create or screenshot the table or chart (of the insights), insert it to a Word document, and write a short description for it*" (P3). After accumulating enough insights, participants moved to the writing stage. All participants indicated that they frequently revisited the data while writing, as their original insights could be unclear, complicated, incorrect, obsolete, or unappealing to present. Their document would often be iterated on by collaborators, leading to additional data exploration. Thus, their writing processes were highly intertwined with data exploration. Finally, the document would be carefully reviewed together with the data to ensure that there were no inconsistencies between the document and data before delivery.

3.2.1 Tedious and Frequent Data Retrieval (T1). When writing data documents, participants needed to retrieve data from their data analysis applications (e.g., Excel) to document in the authoring applications they used (e.g., Word). All participants reported that the "*frequent application switching and navigation to the data*" caused significant friction to the retrieval process. For example, with Excel, participants needed to first identify the correct datasheet, and then scroll within the sheet to locate the data they wanted (P1-6, P8). Participants (P1-4, P6) often would use the Search function to accelerate their navigation, which required them to memorize specific data properties and navigation pathways when multiple matches were found. Once data was located, participants needed to transfer it to a text editor. While participants often relied on copy-and-paste to avoid errors, they often needed to change the data format (e.g., converting large absolute values to abbreviated forms, P5) or perform simple calculations (e.g., ratio of change, P2), so they had to manually type the data into the document. Each of these steps was tedious but also repeated numerous times during authoring, resulting in time-consuming and error-prone workflows.

3.2.2 Inefficient and Error-prone Maintenance of Data Consistency (T2). Ensuring consistency between a document and its underlying data was regarded as important, as erroneous data reporting could lead to extra iterations of a document (P2), bad records in one's career history (P1), or even financial losses for a company (P3). Professionals reported that the inconsistencies were usually caused by data updates. For example, P5, a marketing manager, often started to draft a document before all the data became available so that they could meet deadlines, which led them to update their analysis and document as soon as new data became available. P3, who worked in a financial services company, frequently updated her documents when there were adjustments in model parameters. Whenever the underlying data was updated, all participants reported that they needed to "*read through [their] documents carefully and fix the inconsistent content manually*" (P5), which was "*inefficient and prone to error*" (P1). P1 noted that the IT team in his company developed a plugin that synchronized the data between Excel and Word automatically, but it required the user to manually connect cells to words. P3 mentioned that a professional review team in her company would proofread her documents to highlight any inconsistencies. Nevertheless, these methods were noted as being cumbersome, expensive, and time-consuming.

3.2.3 Significant Overhead for Iteration (T3). Participants reported that exploring different ways to present data was a common but time-consuming task (7/8). They needed to perform additional data exploration during the writing stage, because "*only when I write down the data in the document, I know what's the best way to present it*" (P2). As an operating officer in an IT company, P2 reported that she needed to frequently switch the presentation of user growth data on a yearly, quarterly, and monthly basis.

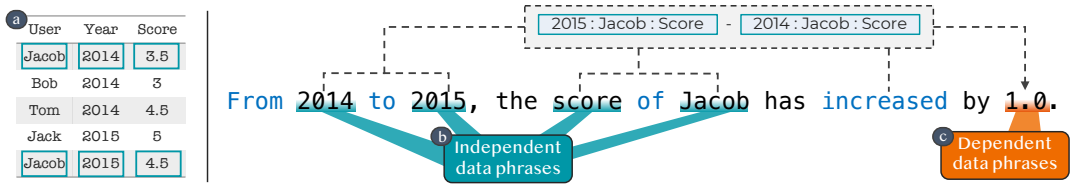

Figure 2: The connections between text and data. a) The dataset to report. b) Data phrases directly reporting the underlying data. c) A data phrase connecting with the data under the constraints of other phrases. The Blue text represents the keywords used to compute dependent phrases.

Exploring alternative data presentations, however, was reported as being time-consuming, because participants often needed to repeat their analysis steps, create new tables and charts, and update the relevant text with new data. P6 mentioned she always used tables or charts to show evidence for the insights reported in the text, i.e., "*if I want to report a new metric, I will add one more column to the table*" (P6). P8 noted that to "*add one more sentence*" to introduce "*the ratio of a group of users to all users*", he needed to go back to Excel, perform numerous operations to re-create tables and charts, and then insert them into the document.

Participants reported that during the writing stage, they frequently iterated on the presentation of data. However, even the smallest changes caused significant ripple effects to the data reported in the text, as well as the corresponding tables and charts. Due to such significant overhead, participants and their collaborators had to iterate on the document offline when iterations were suggested in real-time, requiring additional meetings and discussions, thus hindering their collaborative process.

3.3 Summary

The formative study found that professionals encountered several issues while writing data documents with mainstream tools and they addressed these issues manually. They struggled while inputting the data into their documents, maintaining the consistency between their documents and data, and handling the numerous interconnected components during iterations. The findings indicate that the key reason for their tedious and ineffective workflows was the lack of connections that existed between the text in data documents and the data in datasets, which needed to be created and maintained with minimal effort from users.

4 CROSSDATA

When using text to describe data from a dataset in a document, a user establishes an abstract connection between the text and the data elements in their mind. A key insight from the formative study was that current tools require the user to mentally maintain these connections, leading to tedious, repetitive, and error-prone operations. We propose reifying these connections as persistent, first-class objects [5, 19, 22] and leveraging them to address the issues that occur during the writing process. To this end, two steps were undertaken: 1) we developed a connection engine to automatically establish and maintain these connections during writing processes and 2) we designed a set of interactions based on these connections to tackle the issues identified in the formative study. The present work focuses on tabular data, which is one of the most common data formats.

5 THE CONNECTION ENGINE FOR TEXT-DATA CONNECTIONS

Given the text in a data document and an underlying dataset, our goal was to infer, establish, and maintain text-data connections.

5.1 Connections Between Text and Data

When describing data using text, the phrases in text can connect with the underlying data in two ways:

1. **Independent data phrases**, directly report items (rows), attributes (columns), and values (cells) in the dataset. For example, in Figure 2, *2014*, *2015*, *score*, and *Jacob* (Figure 2b) are connected to the cells in the table (Figure 2a). Independent data phrases can be used as arguments to compute dependent data phrases.
2. **Dependent data phrases**, present the output of data operations that take other data phrases as arguments. A dependent data phrase can report data in the dataset or derived values that do not exist in the dataset. For instance, the last term *1.0* (Figure 2c) is calculated based on the other phrases and connects to the data dependently. The data operations to compute a dependent data phrase are described by keywords such as *from*, *to*, and *increased*.

5.2 Establishing Text-Data Connections

The Connection Engine helps users establish and maintain connections during the writing process (Figure 3). Suppose that after writing the first half of a sentence (S_{former}), a user is typing a new phrase (P_{cur}). The Connection Engine generates all potential connections for P_{cur}, which are presented as a list of data phrases to the user. Once a data phrase is chosen by the user, the Connection Engine inserts the phrase into the document with the text-data connection and all relevant meta information is maintained.

5.2.1 Establishing Connections for Independent Data Phrases. The Connection Engine generates the potential independent phrases for P_{cur} by performing string matching of P_{cur} with all strings in the dataset and synonym matching with all attribute names in the dataset. The synonym matching is achieved by calculating the similarity of the word embeddings provided by Spacy [18], an industrial-strength NLP toolkit. All matches will then be returned

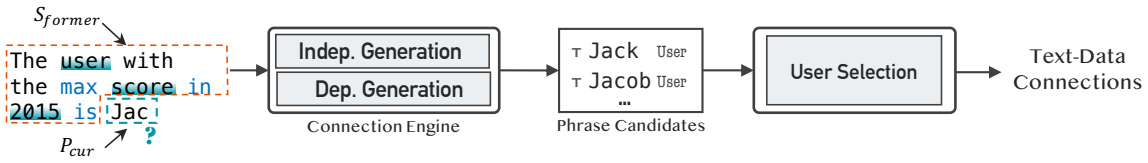

Figure 3: The pipeline to establish text-data connections. The Connection Engine takes a sentence as input and outputs a list of data phrase candidates. The user can select from the candidates to establish text-data connections.

as suggestions, ordered by their matching scores. Selecting a suggestion will insert an independent phrase and create a connection between the independent phrase and the underlying dataset.

5.2.2 Establishing Connections for Dependent Data Phrases. Since dependent data phrases are the result of data operations that take other phrases as arguments, the Connection Engine takes three steps to identify, assemble, and execute the data operations, and then returns the results of the data operations as suggestions to the user. Selecting a suggestion will insert a dependent data phrase and establish a connection with the underlying data operation:

1. **Identifying data operations**: To detect data operations, the Connection Engine matches words and phrases with keywords in a predefined operation dictionary. The dictionary is derived from Amar et al.'s work, which summarized 10 low-level analytical operations for data analysis, such as retrieve value, filter, and compute derived value [2]. This summarization has been widely used in NLI systems to extract desired data operations from users' input queries [21, 39]. An operation takes a few arguments as input and outputs either an item (row), an attribute (column), a value (cell), or a derived value of the underlying dataset. The detailed definition of operations implemented in the current system is provided in the supplemental materials.
2. **Assembling data operations with arguments**: As an operation needs arguments to compute output, the arguments of an operation can either be independent data phrases or the output of other operations. To infer the arguments for each operation, we parse the input text as a constituency tree using the Berkeley Neural Parser [29] through its integration with Spacy. Within a constituency tree, each node represents a text phrase in the sentence (e.g., noun/verb/proposition phrases), with smaller phrases being deeper in the tree, i.e., the leaf nodes are words. Therefore, the Connection Engine uses a bottom-up order to recursively examine whether the independent data phrases and operations in a node can be assembled as a complete data operation, as well as whether data operations should be assembled as compounded data operations. The Connection Engine employs a rule-based method to achieve the examination, as explored in previous NLI research [21, 39, 50]. Specifically, the Connection Engine matches the set of phrases and their grammatic relationships (also provided by Spacy) of a node with pre-constructed rules, each of which describes the necessary arguments for a data operation and the required data types (i.e., item, attribute, or value) for the arguments. The pseudocode for the assembling process is provided in the supplemental materials.
3. **Executing data operations**: Finally, the Connection Engine executes the data operation in the root node of the sentence to obtain the result. Since a keyword may match different operations, the Connection Engine employs a greedy strategy to enumerate all possible matched operations for a keyword, assemble them into complete operations, and return all the results as dependent phrase candidates for the user.

Take the sentence "*The user with the max score in 2015 is*" as an example. The Connection Engine starts the inferring process from the leaf node "*2015*", which reports a value in the data. Since "*2015*" is an independent phrase and the only one at the lowest level, no data operations can be inferred. The Connection Engine then recursively processes the parent nodes of 2015 to a proposition phrase (PP) node and infers a filter operation for the keyword "*in*" with "*2015*" as the argument (Figure 4a1). Similarly, the Connection

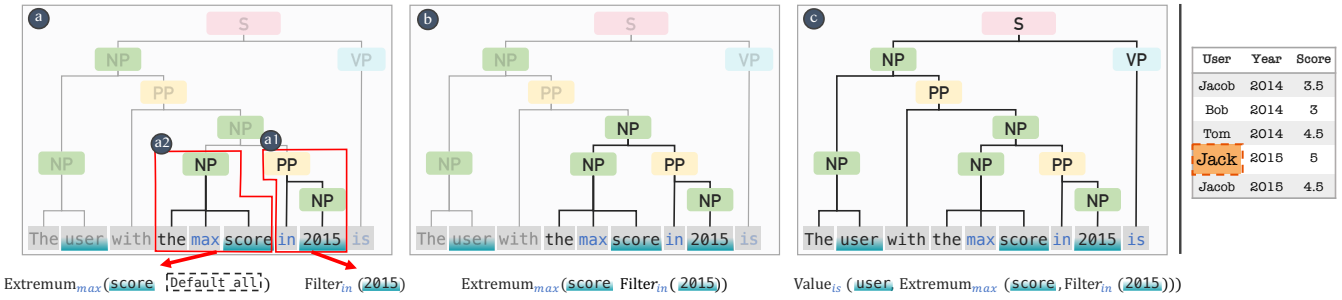

Figure 4: An example detailing how the Connection Engine infers the data operations and suggests dependent data phrases. The engine first parses the sentence into a constituency tree, each of whose nodes represents text phrases (e.g., noun/verb/proposition phrase) in the sentence. Then, the engine infers and assembles data operations in a bottom-up order (a - c). The output of the operation in the root node is returned as suggested dependent data phrases.

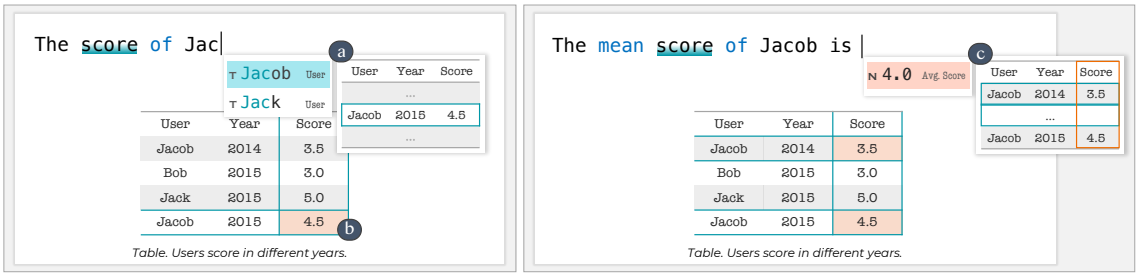

Figure 5: Retrieving Data and Computing Values. a) A list of independent data phrases (highlighted by the cyan background) are retrieved and suggested for the user. b) The data mentioned in the sentence is highlighted. c) The mean score is computed and suggested as a dependent data phrase (highlighted by the orange background) for the user. Detail information about each suggestion is provided to assist in resolving ambiguities.

Engine infers a find extremum operation for the keyword "*max*" on the "*Score*" column from the phrase "*the max score*" (Figure 4a2). According to our predefined rules, the operation finds the extremum in all rows by default. When process to its parent node (Figure 4b), the engine fills the default argument (i.e., all rows) with the output of the filter operation inferred in Figure 4a1 since its output is a list of rows. The engine recursively repeats this process and finally infers a retrieve value operation in the root node from the keyword "*is*", whose arguments are the phrase "*user*" and output of the find extremum operation (Figure 4c). As such, the dependent data phrase is computed from a compounded operation of the filter, find extremum, and retrieve value operations. The output of this compound operation, "*Jack*", will then be recommended to the user. Once the user selects "*Jack*" from the suggestions, a dependent phrase will be inserted, and a text-data connection will be established. The detailed rules for each operation and the pseudocode of the algorithm are provided in the supplemental materials.

6 LEVERAGING CONNECTIONS FOR DATA DOCUMENT AUTHORING

CrossData leverages the text-data connections found by the Connection Engine to provide novel interactions that address the issues identified in the formative study, thus enabling users to efficiently retrieve, compute, explore data, and adjust tables and charts during the writing of data documents, while automatically maintaining data consistency between the text, data, tables, and charts.

6.1 Connections for Inputting Data

The formative study found that data retrieval is tedious but repeated numerous times when authoring data documents (T1). Professionals manually retrieved data from data processing tools (e.g., Excel), leading to issues while application switching, navigating data, and transferring data into word processing tools (e.g., Word). To address these issues, several interactions that enable users to leverage the output of the Connection Engine were thus designed.

6.1.1 Retrieving Data. As a user types in the text editor, CrossData automatically runs the Connection Engine to detect the connections. The underlying data elements that the text potentially connects to are returned as suggestions for the user in a list (Figure 5a). Additional information (e.g., the data types, the context in the spreadsheet, etc.) about each suggestion is provided for each list item to help the user select the correct data and resolve ambiguities. If the underlying data table is also visible on the user interface, CrossData automatically highlights the corresponding row, column, or cell based on the data phrases the user is typing (Figure 5b). Such reference highlighting can help users efficiently locate the elements in tables. The user can select a suggestion from the list to insert it into the text editor or simply enter the text following the suggestion. CrossData will automatically maintain the connection between the text and data for later reuse.

6.1.2 Computing Values. Sometimes the user needs to compute and input values that do not exist in their dataset. CrossData detects these dependent connections and calculates their derived value using the Connection Engine. The derived value and the detailed information about the calculation are displayed as suggestions for the user (Figure 5c). The user can select and insert the derived data while preserving the connection.

6.1.3 Using Placeholders. An issue when retrieving or computing data in a written sentence, which differs from command-like sentences in other NLIs systems, is that the data that one may want to retrieve or compute could be input before its dependency is retrieved or computed. CrossData thus provides a set of placeholders, such as Diff, Ratio, and Count, that the user can employ to indicate

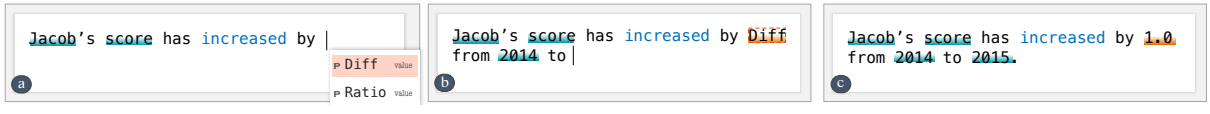

Figure 6: Using placeholders. a) There is not enough information provided in the sentence to calculate the difference between Jacob's scores in different years. b) CrossData allows the user to use a Diff placeholder to indicate the computation. c) CrossData updates the placeholder as more information is provided.

expected data types. For example, in Figure 6a, if the user wants to report the increase in Jacob's score while the year range is unknown, the user can press the Tab key to open the suggestion list to select and insert a placeholder (Figure 6b). Then, whenever new data phrases in the sentence are inserted or detected, the Connection Engine will attempt to evaluate and update the placeholders (Figure 6c). All placeholders are thus dependent data phrases.

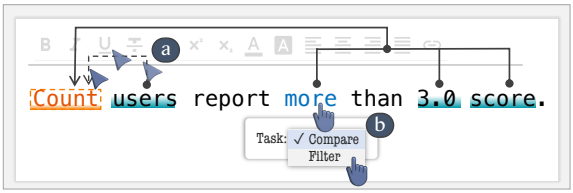

Figure 7: Fixing misdetections by a) hovering over the "*Count*" placeholder to visualize its dependencies and linking "*users*" to "*Count*" to fix the missing dependency or by b) hovering over the operation keyword "*more*" to display the task inferred from it. In this example, "*more*" should be interpreted as a filter instead of a comparison task.

6.1.4 Fixing Misdetections. It is not uncommon for CrossData to retrieve or calculate incorrect data for dependent data phrases. The incorrectness can be caused by mis-detected dependencies (i.e., wrong input) or operation keywords (i.e., wrong tasks). CrossData allows the user to interactively correct these misdetections by hovering over a dependent data phrase to visualize and modify its dependencies (Figure 7a) or hovering over operation keywords to refine their tasks (Figure 7b).

6.2 Connections to Maintain Consistency

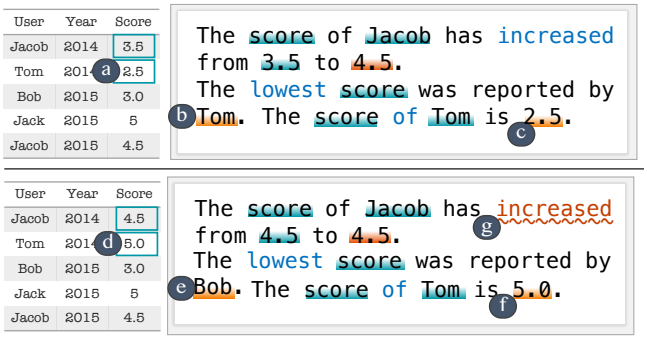

Figure 8: Maintaining consistency automatically. After changing Tom's score (a) from 2.5 to 5.0 (d), CrossData updates all related sentences, such as the user with the lowest score (b) and Tom's score (c). Problematic operation keywords caused by the updated data will also be highlighted, e.g., after changing Jacob's score (a) from 3.5 to 4.5 (d), the "*increase*" description (g) is incorrect.

The formative interviews demonstrated that most of the professionals manually maintained consistency between their text and data and considered this process to be time-consuming and error-prone (T2). With the help of preserved connections, CrossData can update data phrases and highlight problematic operation keywords to help users maintain consistency.

6.2.1 Data-driven Updates. Whenever a data element within the underlying dataset is updated, CrossData will automatically update all independent and dependent phrases that connect to the data element. For example, if the user changes the score of Tom from 2.5 (Figure 8a) to 5.0 (Figure 8d) in the table, CrossData will update Tom's score to 5.0 (Figure 8f) in the last sentence; meanwhile, Tom (Figure 8b) will be updated to Bob (Figure 8e) accordingly.

6.2.2 Operation Keywords Checker. Inconsistencies can also exist between the operation keywords and the data. For example, when changing the score of the first row from 3.5 (Figure 8a) to 4.5 (Figure 8d), the operation keyword "*increase*" is inconsistent with the data. However, different from data phrases, updating operations can be challenging because operation phrases are usually text descriptions. In such cases, CrossData will highlight the problematic operation keyword with red wavy underline (Figure 8g).

6.3 Connections for Interactive and Flexible Iteration and Exploration

When iterating on a data document, users frequently change various elements in their document (T3). While the interaction techniques introduced above can alleviate the overhead of retrieving values and maintaining consistency during iteration, a pressing and unaddressed challenge is the cascading effects that occur when changes are made to text, tables, and charts.

CrossData addresses this challenge by reifying text-data connections as interactive objects, which enable users to manipulate them to iterate on data documents and explore new insights directly in a document. Because the data phrases, tables, and charts are all connected with the underlying data, the necessary changes can be automatically performed without additional user effort.

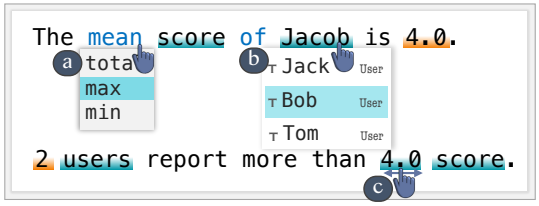

Figure 9: Interactive text. CrossData enables users to interactively iterate operation keywords (a) and independent (b, c) phrases. The interactions will trigger the related dependent data phrases to be updated.

6.3.1 Interacting with Data-Driven Text. Text phrases that are connected with underlying data can be interactively manipulated. As independent phrases represent an item (row), attribute (column), or value (cell) within the spreadsheet, CrossData allows the user to interactively change an independent phrase to other items, attributes, or values (Figure 9b, c). The interactions provided by an independent phrase depend on its data type, e.g., quantitative, nominal,

or ordinal. To avoid meaningless changes, CrossData only allows users to change item phrases to other items, attribute phrases to other attributes that have the same data type, and value phrases to other values in the same column.

Users often need to iterate on the metrics they use to report on their data, such as changing the average value to the median value or from a daily basis to a weekly basis. CrossData enables users to interactively alter operation keywords to achieve such goals. For example, in Figure 9a, the user can click and change the mean to other computations such as total, maximum, or median. The available operation keyword alternatives are predefined within a curated dictionary.

Changes to interactive text phrases are automatically propagated to other phrases according to the inferred data operation. For example, in Figure 9b, if the user interactively changes *Jacob* to *Bob*, CrossData will update the value *4.0* to Bob's mean score.

6.3.2 Automatic Adjustments of Tables and Charts. Because the text, tables, and charts embedded in a document are all connected to their underlying data, CrossData can automatically update tables and charts with the text to ensure the textual descriptions and data visualizations are consistent. CrossData supports three types of language-oriented manipulations of embedded data tables, based on the detected data operations in the text. First, when a dependent phrase is the output of a sort or find extremum task, CrossData will sort the table based on the column involved in the task. Second, if the user computes a dependent phrase by aggregating multiple rows (e.g., summation), CrossData automatically adds a new row that shows the aggregation results to the table (Figure 10a). Third, if the dependent phrase computes a new attribute for an item (e.g., the increase from last year), CrossData will attempt to calculate this attribute for all rows and add a new column to the table (Figure 10b). Changes in the tables are suggested to the user, which they can accept or reject.

Similarly, embedded charts are also synchronized with textual descriptions. CrossData automatically updates the charts if different data properties are reported in the text. For example, when the user switches the reporting of new infection cases from daily (Figure 10c) to weekly (Figure 10d), CrossData will automatically switch the underlying data source of the chart to synchronize with the change. CrossData will also automatically annotate the time period of the charts based on the dates reported in the text (Figure 10e). Since both the text and chart are connected to the underlying data, the user can directly manipulate the chart to adjust the text (e.g., dragging the chart overlay in Figure 10e), or vice versa, which can facilitate better authoring and reading experiences.

The supported editing operations are limited in the current implementation because the scope of this work is to demonstrate promising novel interactions and workflows enabled by text-data connections. Nevertheless, it is possible to extend CrossData to support more visualization editing operations and this is left for future exploration.

7 TECHNICAL EVALUATION OF THE CONNECTION ENGINE

The effectiveness of CrossData depends on whether the Connection Engine can suggest the correct data phrases to the user. Therefore, we conducted a technical evaluation to assess the accuracy and robustness of the Connection Engine. We report on the experiment settings, results, and investigation of the failure cases, as well as potential improvements to the Connection Engine.

7.1 Experiment Settings

7.1.1 Methodology. The goal of the evaluation is to assess whether the Connection Engine can suggest the correct data phrases based on the text in the writing process. Because independent data phrases are suggested based on string matching, which is usually highly accurate, we focused on evaluating the generation of dependent data phrases. Specifically, we gathered a corpus of sentences together with their corresponding datasets. For each sentence, we manually labelled all independent data phrases with the connections to the datasets as part of the input and all dependent data phrases as ground truth. We then input each sentence word by word into the Connection Engine to simulate a realistic writing experience and compared the suggested dependent phrases against the ground truth. The experiment was run on a Macbook Pro with a i7 2.2GHz Intel CPU.

7.1.2 Dataset. We collected sentences from 10 data documents from reputable public sources that cover multiple domains, such as World Health Organization [43], Bureau of Labor Statistics [41],

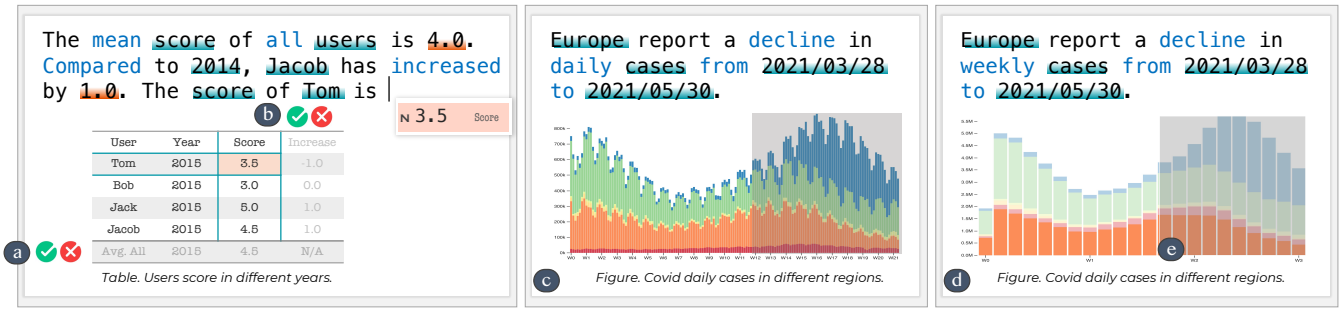

Figure 10: Adjustments of tables and charts based on the text. Based on the user's writing, CrossData adds a new row and new column to the table (a) and switches the data sources from daily (b) to weekly (d). Users can also directly manipulate the chart to update the text, e.g., by dragging the chart annotation (e).

Pew Research Center [9], National Center for Education Statistics [20], National Institutes of Health [40], California Department of Public Health [42], and a private company [23], as well as their corresponding datasets. We sampled the sentences by: 1) manually filtering all sentences that reported data in the documents and then 2) randomly sampling no more than 30 sentences from each document. For each sentence, we manually labeled the independent and dependent phrases. In total, the corpus contained 206 sentences (5398 words), with 807 independent phrases and 529 dependent phrases.

7.1.3 Metrics. We measured the ratio of correct dependent data phrases recommended by the Connection Engine to the total number of dependent data phrases. When the engine returned multiple candidates for a dependent phrase, we counted it as correct if the top 5 candidates contained the correct one. We also measured the time to compute the candidates.

7.2 Results

The accuracy of the dependent phrases was 88.8% (i.e., 470 corrects), which demonstrates the robustness and accuracy of the Connection Engine. Among these correct cases, the majority were computed by the compounded operation of filtering and retrieving values (i.e., 262 cases, 55.7%), the finding extreme operations (i.e., 62 cases, 13.2%), the compounded operation of finding extreme operations and retrieving values (i.e., 61 cases, 13.0%), and the compounded operation of finding extreme operations and comparing values (i.e., 48 cases, 10.2%). This echoes the findings from Section 3.2, reflecting that the data retrieval operation was prevalent in real-world data documents. The average time to generate candidates was 0.3 seconds, which was sufficient for interactive use cases and could be further optimized with better implementations.

7.3 Failure Cases Analysis

We further investigated the failure cases and identified three major reasons for these failures. Note that a failure may be caused by multiple factors.

7.3.1 E1: Lack of Context (i.e., 50.8% of cases). Among the failure cases, the majority of cases (i.e., 31) failed because certain expressions (e.g., it, these, previous years) referred to other data phrases. For example, with the sentence "*These three countries comprised 89% of all cases reported in the region*", to compute the "*89%*", the Connection Engine needed to know which countries "*These three countries*" referred to. In this example, the three countries were mentioned in previous sentences as independent phrases. This problem, however, can be addressed by employing co-reference resolution, i..e, finding expressions that refer to the same entity within or between sentences, which has been advanced in recent years. The Connection Engine can integrate co-reference resolutions models [35] to connect data phrases in previous sentences to the present one, thereby maintaining the context to infer text-data connections.

7.3.2 E2: Expect Textual instead of Numerical Outputs (i.e., 27.9% of cases). Seventeen cases failed because the expected output was a text description rather than a number. For example, in "*Two in five e-cigarette users reported usually paying for their own e-cigarettes*", the expected output was "*Two in five*" while the engine returned "*43%*". To address this issue, the Connection Engine could generate more candidates with different formats, or adopt more advance generative language models, such as GPT-3 [8]. Note that while the data formats of the suggested phrases do not match the ground truth, the underlying data operations inferred by the Connection Engine were correct. This means that the Connection Engine could accurately infer 91.9% of all data operations.

7.3.3 E3: Uncovered Operations (i.e., 21.3% of cases). Thirteen cases failed because the required data operations in the sentences were not covered by the 10 low-level data operations summarized by Amar et al. [2]. In the example "*Cases have decreased steeply for the past four weeks*", computing the "*four weeks*" is a high-level analytical task (i.e., given a column and a text description of the trend, report the range of rows that fulfill the trend), that is not supported in our prototype. Considering the rule-based nature of the Connection Engine, these cases could be addressed by extending the predefined operation dictionary and corresponding rules.

7.4 Summary

In summary, the analysis showed that the Connection Engine was robust enough to achieve a high accuracy when generating dependent phrases about a set of real-world sentences collected from multiple domains. The in-depth analysis indicated that most of the failure cases could be fixed by extensions to the engine.

8 EXPERT EVALUATION

We developed CrossData as a technology probe to explore the notion of language-oriented data bindings and recognized that it may, at first, create usability problems for users who are familiar with existing tools. To gain feedback about the effectiveness of our approach without being bogged down by the initial challenges some users may encounter with usability, we conducted an expert evaluation study. We focused on collecting experts' feedback about the usefulness of each interaction technique and how language-oriented authoring could facilitate the overall workflow of authoring data documents.

8.1 Participants and Apparatus

Eight participants were recruited to participate in the study (E1 – E8, 5 female, age 28 – 31), i.e., 1 auditor (accounting), 1 operation officer (internet services), 1 investment banking associate (financial services), 1 due diligence consultant (business services), 2 marketing managers (internet services and retail) and 2 researchers (data science and public healthy). E1-E5 participated in the formative study. All participants had more than 5 years of experience analyzing data and writing data documents as part of their daily work. The most used data processing and writing tools included Microsoft Excel, Sheets, Word, Google Docs, and Tableau. The study was conducted remotely with CrossData implemented as a responsive Web application that participants could directly access from their personal computers. Video conferencing was used to communicate with participants, share screens, and record the study. Participants received $60 (USD) for the approximately 90-minute session.

8.2 Procedure

Each session included the following phases:

8.2.1 Introduction and Training (30 mins). The experimenter first introduced the study protocol, research motivation, and concepts of CrossData. Then, the experimenter walked the participants through the system with an example that contained two datasets that were presented as a table and a bar chart, and five insights to report. Participants were encouraged to ask questions anytime during the process. Participants were then asked to replicate the example to become familiar with the system.

8.2.2 Reproduction Task (15 mins). Participants were asked to reproduce a given data document, which presented a USA COVID-19 dataset with a multiple line chart and six sentences, each of which reported an insight. The original datasets, a multiple line chart, and a choropleth map were provided as the context for the insights.

8.2.3 Creation Task (20 mins). Participants were asked to write a short document to report on three datasets about Global COVID-19 cases. Each dataset included one data representation (i.e., a chart or a table) and three insights. The short document needed to contain at least one insight from each dataset, and one data representation. To simulate realistic iterative processes, after the participants finished the document, the experimenter asked them to iterate on the document by 1) reporting two more insights, 2) inserting one more chart or table, and 3) changing the data phrases or operators in the documents. The changes to the data phrases or operators were selected to ensure that the participants experienced all of the proposed interaction techniques.

8.2.4 Semi-structured Interview and Questionnaire (25 mins). After the creation task, participants completed a questionnaire that probed the usefulness and usability of the techniques using a 5-point Likert scale (i.e., 1 – Strongly Disagree, 5 – Strongly Agree). Then, the experimenter conducted a semi-structured interview to further collect feedback about the utility of each interaction technique, CrossData's effectiveness in supporting realistic workflows, limitations of the proposed techniques, and potential improvements.

8.3 Results

All participants successfully finished the reproduction and creation tasks. On average, each participant wrote 12.6 sentences and 123.3 words, which contained 22.1 independent and 13.6 dependent data phrases. All participants experienced all the proposed interaction techniques. We report and discuss how the proposed interactions 1) addressed the issues identified in the formative study, 2) could improve participants' current authoring workflows, and 3) could be extended for data exploration and to enable new workflows that bridge the gap between the writing and data exploration stages. We also report on observed behaviors that suggested future improvements for real-world usage.

8.3.1 Utility of Text-Data Connections. The interaction techniques provided by CrossData were lauded and rated as useful by participants (Figure 11). Participants confirmed that these techniques addressed key pain points in their daily workflows and considered them to be "*killer features*" (E5) for writing data documents. Among the various techniques, participants appreciated the compute value (7/8 strongly agree, 1/8 agree) and retrieve value (6/8 strongly agree, 2/8 agree) techniques as they facilitated the inputting of data (T1) by "*enable[ing] computation using words (E8)*", "*reduc[ing] application switching*" (E8), and "*avoid[ing] typos*" (E3). As commented by E3, these techniques addressed some "*fundamental issues*" and thus brought "*fundamental improvements to the writing process.*"

Participants also responded positively (4/8 strongly agree, 4/8 agree) to the techniques designed to maintain consistency between data and text (T2). These techniques helped users "*ensure consistency*" (E3) with "*fewer manual efforts*" (E1). E6 believed that these techniques could help her company "*reduce human resource costs on the review team*".

The interactive techniques that facilitated iteration (T3) via interaction with data-driven text (5/8 strongly agree, 3/8 agree) and the automatic adjustments of tables (5/8 strongly agree, 3/8 agree) and charts (5/8 strongly agree, 3/8 agree) were also appreciated by participants because these techniques could "*significantly reduce working back-and-forth*" (E5) and enabled participants to "*rapidly refine the charts [and tables]*" (E7). Participants (E1, E4, E7) also remarked that the interactivity of the text, as well as the real-time synchronization between text, table, and charts, made the authoring process "*fun and engaging*" (E1), but also could assist in thinking

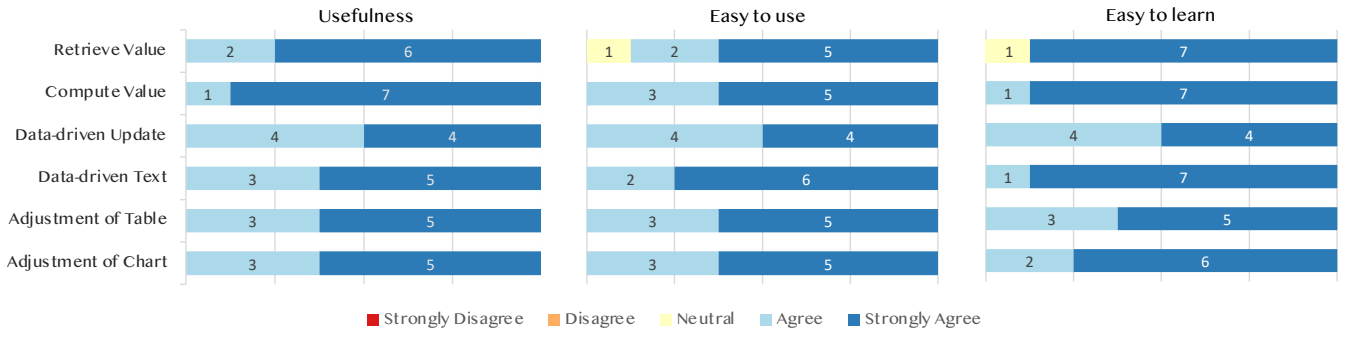

Figure 11: Likert-scale responses to "This technique was useful in my writing process.", "This technique was easy to learn.", and "This technique was easy to use."

processes and inspire more ideas during writing as the user can "*see what he is writing*" (E4).

8.3.2 Authoring Workflow vs Traditional Tools. All participants agreed that the interactions provided by CrossData would mesh well with their current workflows (4/8 strongly agree, 4/8 agree), e.g., "*you just need to write as usual*" (E1). They further commented that these interaction techniques did not require installing another application and could be easily integrated within existing tools by "*installing [them as] a plugin to my Word*" (E2).

All participants found that the interaction techniques could streamline their workflows due to "*less context switching*" and allow for efficient iterations of a document. E8 noted that she used to frequently switch between "*Excel, Word, and sometimes the calculator*" during the writing process, which was "*stressful and distracting.*" By integrating CrossData with the existing tools, participants could "*concentrate on her writing*" (E8), and "*focus on the current writing without worrying about refining or updating other sentences*" (E7).

Another improvement to participants' workflows that was mentioned was "*facilitating the process of getting feedback from others*" (E2). Mainstream tools such as Word and PowerPoint present reports in a static manner and thus hinder authors from addressing or responding to others' feedback immediately, whereas the features provided by CrossData "*make it very useful to answer ad-hoc questions during the discussions that would normally require some follow up work, e.g., swap out regions, look at percentage changes between different time periods, etc.*" (E5)

In terms of the negative impacts these techniques may have on their workflows, E7 noted that "*perhaps the only cost is to learn how to use [them]*". Specifically, "*you need to understand the concepts and get familiar with, for example, placeholders*" (E7). Nevertheless, all participants reported that the interaction techniques were easy to learn and easy to use (Figure 11), indicating that the downside of using them would be negligible.

8.3.3 Enabling New Workflows to Bridge the Gap between Data Exploration and Writing. While CrossData was designed to support the writing stage, the intertwined nature of exploration and writing inspired participants to imagine CrossData beyond the presented tasks, and they suggested several benefits that could be enabled by the language-oriented techniques to facilitate data analysis and exploration.

First, natural language enables one to express reusable high-level goals instead of performing transient low-level operations, thereby improving the efficiency of data exploration. E4 noted that with the compute value technique provided by CrossData, he could efficiently calculate a value by typing a sentence instead of "*scroll up and down in a sheet and brush and re-brush the cells.*" Moreover, E3 suggested that the exploration process could be easily reused for different data by copying and pasting the text, i.e., "*I can write text to retrieve and calculate values, and then copy the text to another sheet to get new values ... this is impossible in Excel since I cannot copy my interactions on one sheet to another.*"

Second, CrossData could facilitate active thinking during the exploration process. E1 found that the suggestion list and interactive operators inspired him to explore the data from new angles that he missed before. He remarked that the suggested text was similar to the query recommendations in search engines. E4 explained that sometimes he stopped data exploration because it required too many tedious operations with Excel, i.e., "*exploration is a process of thinking rather operating the Excel... I will definitely explore more if only a few clicks or types are required.*"

Third, language-oriented data exploration enabled users to "*record their exploration process as [a] draft*" (E1) and naturally "*shift from data exploration to writing.*" All participants confirmed that there was a gap between data exploration and presentation in their current workflow, which has been recognized in prior work as an important research direction to improve the workflow of data analysts [34]. E7 commented that these "*two interconnected stages [i.e., data exploration and communication,] were usually separated in two disconnected applications.*" With language-oriented interaction techniques, however, data exploration and data document authoring can be tightly integrated such that "*exploring [the data] is drafting [the document] and vice versa.*"

8.3.4 Observed Behaviors. We observed several interesting user behaviors that reflected participants' real-world writing practices that were not supported by our current implementation.

First, when the data operations were simple, participants tended to directly type the result, which could result in untracked connections. For example, when writing "*The U.S. reports the most new cases in America*", E3 manually typed "*The U.S.*" instead of using the placeholder feature. This was because that the participant already knew the desired data, and inserting a placeholder required more effort. The result, however, was that "*The U.S.*" text would not be updated when the participant was asked to modify "*America*" to "*Africa*", causing data inconsistency due to the missing connections. While the Connection Engine is currently designed to interactively recommend data phrases, to address this issue, it could be extended to detect and connect manually typed dependent phrases to ensure all data phrases would be connected with the underlying dataset.

We also observed that some participants reported approximate numbers instead of exact data values, which caused undesired suggestions from the engine. For example, E1 wrote that "*[Placeholder] countries in America report more than 10,000 ...*". He wanted to connect "*10,000*" with the new cases column. However, because "*10,000*" is an approximate number that did not exist in the new cases column, the Connection Engine could not return suggestions because it relies on string and synonym matching to suggest independent phrases. E1 then struggled to connect the "*10,000*" with the new cases column. Such behavior was also observed in other participants (e.g., E2, E5, and E7). While participants altered the approximate numbers to exact values to create connections, this issue could be common in real-world scenarios. To address this, CrossData could be extended to allow users to manually insert their desired connections or support fuzzy data value matching when certain keys are present, such as "almost" and "more than".

Third, the participants tended to write safe, simple sentences to ensure the connections would be created successfully during writing. Overall, the sentences were relatively simple and had similar structures to the sentences in the training and reproduction tasks. While this could be attributed to the limited time frame of the task, it is possible that participants faced a dilemma when guessing which written text the system could understand and establish connections with. Such an issue has been recognized as a

long-standing challenge for users of NLI systems [53]. To address this issue, we propose that the system could provide alternative methods (e.g., interface actions) to allow users to manually create text-data connections instead of fully relying on the auto-extraction of the connections from the text. Several participants confirmed this improvement would be useful and necessary in their interviews, indicating that "*the system should enable users to create or modify the connections after the writing.*" (E7)

8.3.5 Limitations. Participants noted several limitations of CrossData and suggested some improvements. Similar to other interactive systems that employ NLP, CrossData can misinterpret users' intentions due to the reasons discussed in Section 7.3 and Section 8.3.4 (e.g., lack of context, unrecognized approximate numbers). While CrossData allows users to correct misdetections caused by predefined rules, it does not support the correcting errors caused by NLP techniques. All participants expressed their concern regarding this and understood that they could be mitigated by further advancements of NLP techniques, more intelligent connection recognition algorithms, and by them being able to flexibly modify the suggested connections.

Participants also proposed several improvements with regards to extensibility and customizability. For example, E8 suggested that CrossData could support customized operators and calculations or enable users to import domain-specific operators from online libraries. E3 proposed that CrossData should enable users to share their customized operators with others to facilitate collaborative editing. E5 indicated that the system should enable users to "freeze" connections so that they could rephrase sentences without worrying about losing any connections.

Several participants also raised the concerns about scalability. For instance, E1, an auditor, who often needed to write data documents to synthesize findings from more than 50 datasets, noted that connecting a phrase to all underlying datasets could lead to too many possible connections. A potential solution to this could be to add a context-awareness mechanism to CrossData so that it could prune the search space based on one's writing context, e.g., the surrounding sentences, tables, charts, and section titles.

9 FUTURE WORK

Beyond Tabular Data and Basic Charts. CrossData currently supports the connection of text to tabular data, wherein each data item is represented as a row and its attributes are represented as columns. While tabular data is common in practice, it does not naturally contain information about the rich relationships that exist among data items and are often found within graph-based or tree-based data structures. One future direction for language-oriented authoring research could be to support users in connecting text to rich data structures. The data visualizations currently supported within CrossData are basic charts (e.g., line and bar charts), however, future work should explore how to support more customized, complex data visualizations. This, of course, would require the identification of mappings between the natural human language used in data documents and the domain-specific terms used during data analysis and visualization processes. To develop such mappings, we plan to collect and annotate existing data documents that describe or contain various data structures and visualizations.

Blurring the Line between Writing and Programming for Data Analysis and Visualization. In addition to graphical user interface applications, programming is another commonly used modality for data analysis and visualization. For example, computational notebook applications, which enable users to write programs to analyze and visualize data, are becoming increasingly popular. A common practice when using computational notebooks is to write explanatory textual descriptions alongside a program's code to facilitate documentation and collaboration. This presents an opportunity to extend the use of written text for data analysis and visualization. Thus, one future direction could be to integrate CrossData into computational notebooks, so that users can analyze and visualize data by writing descriptive and self-explanatory text without requiring programming skills.

Supporting Dynamic and Interactive Data Presentations. While CrossData leveraged text-data connections to support the authoring of static data documents, the data documents that results were interactive, suggesting opportunities to create interactive documents without any programming. We plan to expand CrossData to further support the creation of data-driven diagrams and simulations. Another future direction will be to explore the creation of other forms of dynamic and interactive presentations of data with text-data connections, such as data videos and data animations. Specifically, the connections between text with tables and charts could be directly employed to create animated changes in tables and charts that correspond with the narration of animation, videos, or slideshows.

10 CONCLUSION

Despite the proliferation of applications and systems that seek to support users while analyzing, visualizing, and communicating data, the authoring of data documents still remains a laborious process. Within this work, we conducted a formative study with eight professionals and found that a key reason for these tedious, repetitive, and error-prone workflows is the lack of connections that exist between text and data. As the process of writing data documents is implicitly the process of establishing connections between text and data, we thus developed a prototype system, CrossData, that would infer text-data connections within written text. The development of CrossData enabled for a systematic exploration of the power of identifying, establishing, and reifying text-data connections as persistent, interactive, first-class objects that could be used to assist in the authoring of dynamic, interactive data-driven documents. A technical evaluation demonstrated the effectiveness and robustness of the connection engine. Results from an expert evaluation found that CrossData not only reduced the manual effort required while writing data documents, but also provided new opportunities to leverage text to support the authoring of data-driven content.

REFERENCES

[1] Katrin Affolter, Kurt Stockinger, and Abraham Bernstein. 2019. A Comparative Survey of Recent Natural Language Interfaces for Databases. *VLDB Journal* 28, 5 (2019), 793–819. https://doi.org/10.1007/s00778-019-00567-8 arXiv:1906.08990

[2] Robert Amar, James Eagan, and John Stasko. 2005. Low-level Components of Analytic Activity in Information Visualization. In *Proc. of InfoVis*. IEEE, 111–117. https://doi.org/10.1109/INFVIS.2005.1532136

[3] Fereshteh Amini, Nathalie Henry Riche, Bongshin Lee, Andres Monroy-Hernandez, and Pourang Irani. 2017. Authoring Data-Driven Videos with DataClips. *IEEE TVCG* 23, 1 (2017), 501–510. https://doi.org/10.1109/TVCG.2016.2598647

[4] Sriram Karthik Badam, Zhicheng Liu, and Niklas Elmqvist. 2019. Elastic Documents: Coupling Text and Tables through Contextual Visualizations for Enhanced Document Reading. *IEEE TVCG* 25, 1 (2019), 661–671. https://doi.org/10.1109/TVCG.2018.2865119

[5] Michel Beaudouin-Lafon. 2000. Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces. In *Proc. of CHI*, Thea Turner and Gerd Szwillus (Eds.). ACM, 446–453. https://doi.org/10.1145/332040.332473

[6] Sonia Bergamaschi, Francesco Guerra, Matteo Interlandi, Raquel Trillo-Lado, and Yannis Velegrakis. 2013. QUEST: A Keyword Search System for Relational Data Based on Semantic and Machine Learning Techniques. *Proc. VLDB Endowment* 6, 12 (2013), 1222–1225. https://doi.org/10.14778/2536274.2536281

[7] Virginia Braun and Victoria Clarke. 2019. Reflecting on Reflexive Thematic Analysis. *Qualitative Research in Sport, Exercise and Health* 11, 4 (2019), 589–597. https://doi.org/10.1080/2159676X.2019.1628806

[8] Tom B Brown, Jared Kaplan, Nick Ryder, Tom Henighan, Mark Chen, Ariel Herbert-voss, Daniel M Ziegler, Gretchen Krueger, Amanda Askell, Christopher Hesse, and Sam Mccandlish. 2020. Language Models are Few-Shot Learners. In *Proc. of NeurIPS*. Curran Associates, Inc.

[9] Pew Research Center. 2021. What the 2020 Electorate Looks Like by Party, Race and Ethnicity, Age, Education and Religion. https://www.pewresearch.org/fact-tank/2020/10/26/what-the-2020-electorate-looks-like-by-party-race-and-ethnicity-age-education-and-religion/

[10] Angel X. Chang, Manolis Savva, and Christopher D. Manning. 2014. Learning Spatial Knowledge for Text to 3D Scene Generation. In *Proc. of EMNLP*. ACL, 2028–2038. https://doi.org/10.3115/v1/d14-1217

[11] Kevin Chen, Christopher B. Choy, Manolis Savva, Angel X. Chang, Thomas Funkhouser, and Silvio Savarese. 2018. Text2Shape: Generating Shapes from Natural Language by Learning Joint Embeddings. In *Proc. of ACCV*, Vol. 11363. Springer, 100–116. https://doi.org/10.1007/978-3-030-20893-6_7 arXiv:1803.08495

[12] Zhutian Chen, Wai Tong, Qianwen Wang, Benjamin Bach, and Huamin Qu. 2020. Augmenting Static Visualizations with PapARVis Designer. In *Proc. of CHI*. ACM, 1–12. https://doi.org/10.1145/3313831.3376436

[13] Zhutian Chen, Yun Wang, Qianwen Wang, Yong Wang, and Huamin Qu. 2020. Towards Automated Infographic Design : Deep Learning-based Auto-Extraction of Extensible Timeline. *TVCG* 26 (2020), 917–926. https://doi.org/10.1109/TVCG.2019.2934810

[14] Matthew Conlen and Jeffrey Heer. 2018. Idyll: A Markup Language for Authoring and Publishing Interactive Articles on the Web. In *Proc. of UIST*. ACM, 29–67. https://doi.org/10.1017/9781108657846.003

[15] Bob Coyne and Richard Sproat. 2001. WordsEye: An Automatic Text-to-Scene Conversion System. In *Proc. of SIGGRAPH*. ACM, 487–496. https://doi.org/10.1145/383259.383316

[16] 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 TVCG* 26, 1 (2020), 906–916. https://doi.org/10.1109/TVCG.2019.2934785

[17] Pierre Dragicevic, Yvonne Jansen, Abhraneel Sarma, Matthew Kay, and Fanny Chevalier. 2019. Increasing the Transparency of Research Papers with Explorable Multiverse Analyses. In *Proc. of CHI*. ACM. https://doi.org/10.1145/3290605.3300295

[18] Explosion.Ai. 2021. SpaCy. https://spacy.io/

[19] Marianela Ciolfi Felice, Nolwenn Maudet, Wendy E. Mackay, and Michel Beaudouin-Lafon. 2016. Beyond Snapping: Persistent, Tweakable Alignment and Distribution with StickyLines. In *Proc. of UIST*. ACM, 133–144. https://doi.org/10.1145/2984511.2984577

[20] National Center for Education Statistics. 2021. Condition of Education 2021. https://nces.ed.gov/pubsearch/pubsinfo.asp?pubid=2021144

[21] Tong Gao, Mira Dontcheva, Eytan Adar, Zhicheng Liu, and Karrie Karahalios. 2015. DataTone: Managing Ambiguity in Natural Language Interfaces for Data Visualization. In *Proc. of UIST*. ACM, 489–500. https://doi.org/10.1145/2807442.2807478

[22] Han L. Han, Miguel A. Renom, Wendy E. Mackay, and Michel Beaudouin-Lafon. 2020. Textlets: Supporting Constraints and Consistency in Text Documents. In *Proc. of CHI*. ACM, 1–13. https://doi.org/10.1145/3313831.3376804

[23] Bilibili Inc. 2021. Announces First Quarter 2021 Financial Results. https://ir.bilibili.com/node/7621/pdf

[24] Project Jupyter. 2021. Jupyter. https://jupyter.org/

[25] Kushal Kafle, Brian L. Price, Scott Cohen, and Christopher Kanan. 2018. DVQA: Understanding Data Visualizations via Question Answering. In *Proc. of CVPR*. IEEE, 5648–5656. https://doi.org/10.1109/CVPR.2018.00592

[26] Dae Hyun Kim, Enamul Hoque, and Maneesh Agrawala. 2020. Answering Questions about Charts and Generating Visual Explanations. In *Proc. of CHI*. ACM. https://doi.org/10.1145/3313831.3376467

[27] Dae Hyun Kim, Enamul Hoque, Juho Kim, and Maneesh Agrawala. 2018. Facilitating Document Reading by Linking Text and Tables. In *Proc. of UIST*. ACM, 423–434. https://doi.org/10.1145/3242587.3242617

[28] Nam Wook Kim, Nathalie Henry Riche, Benjamin Bach, Guanpeng Xu, Matthew Brehmer, Ken Hinckley, Michel Pahud, Haijun Xia, Michael J Mcguffin, Hanspeter Pfister, and Mathew Brehmer. 2019. DataToon: Drawing Data Comics About Dynamic Networks with Pen + Touch Interaction. *CHI* (2019), 12. https://doi.org/10.1145/3290605.3300335

[29] Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual Constituency Parsing with Self-Attention and Pre-Training. In *Proc. of ACL*. ACM, 3499–3505. https://doi.org/doi.org/10.18653/v1/p19-1340

[30] Donald E. Knuth. 1984. Literate Programming. *Comput. J.* 27, 2 (1984), 97–111. https://doi.org/10.1093/comjnl/27.2.97

[31] Nicholas Kong, Marti A. Hearst, and Maneesh Agrawala. 2014. Extracting References Between Text and Charts via Crowdsourcing. In *Proc. of CHI*. ACM, 31–40. https://doi.org/10.1145/2556288.2557241

[32] Chufan Lai, Zhixian Lin, Ruike Jiang, Yun Han, Can Liu, and Xiaoru Yuan. 2020. Automatic Annotation Synchronizing with Textual Description for Visualization. In *Proc. of CHI*. ACM, 1–13. https://doi.org/10.1145/3313831.3376443

[33] Shahid Latif, Diao Liu, and Fabian Beck. 2018. Exploring Interactive Linking Between Text and Visualization. In *Proc. of EuroVis*. Eurographics Association. https://doi.org/10.2312/eurovisshort.20181084

[34] Shahid Latif, Zheng Zhou, Yoon Kim, Fabian Beck, and Nam Wook Kim. 2021. Kori: Interactive Synthesis of Text and Charts in Data Documents. *IEEE TVCG* (2021). arXiv:2108.04203 http://arxiv.org/abs/2108.04203

[35] Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end Neural Coreference Resolution. In *Proc. of ACL*. ACL, 188–197. https://doi.org/10.18653/v1/D17-1018

[36] Can Liu, Liwenhan Xie, Yun Han, Datong Wei, and Xiaoru Yuan. 2020. AutoCaption: An Approach to Generate Natural Language Description from Visualization Automatically. In *Proc. of PacificVis*. IEEE, 191–195.

[37] Zhicheng Liu, John Thompson, Alan Wilson, Mira Dontcheva, James Delorey, Sam Grigg, Bernard Kerr, and John Stasko. 2018. Data Illustrator: Augmenting Vector Design Tools with Lazy Data Binding for Expressive Visualization Authoring. *Conference on Human Factors in Computing Systems - Proceedings* 2018-April (2018), 1–13. https://doi.org/10.1145/3173574.3173697

[38] Tanya Marwah, Gaurav Mittal, and Vineeth N. Balasubramanian. 2017. Attentive Semantic Video Generation Using Captions. In *Proc. of ICCV*, Vol. 2017-Octob. IEEE, 1435–1443. https://doi.org/10.1109/ICCV.2017.159 arXiv:1708.05980

[39] Arpit Narechania, Arjun Srinivasan, and John Stasko. 2021. NL4DV: A Toolkit for Generating Analytic Specifications for Data Visualization from Natural Language Queries. *IEEE TVCG* 27, 2 (2021), 369–379. https://doi.org/10.1109/TVCG.2020.3030378 arXiv:2008.10723

[40] National Institues of Health. 2021. Impact of NIH Research. https://www.nih.gov/about-nih/what-we-do/impact-nih-research/our-knowledge

[41] Bureau of Labor Statistics. 2021. Consumer Price Index News Release. https://www.bls.gov/news.release/cpi.htm

[42] California Department of Public Health. 2018. California Student Tobacco Survey 2017-18. https://www.cdph.ca.gov/Programs/CCDPHP/DCDIC/CTCB/CDPHDocumentLibrary/ResearchandEvaluation/Reports/2017-18CaliforniaStudentTobaccoSurveyBiennialReport.pdf

[43] World Health Organization. 2021. Weekly epidemiological update on COVID-19 - 18 May 2021. https://www.who.int/publications/m/item/weekly-epidemiological-update-on-covid-19---18-may-2021

[44] Fatma Özcan, Abdul Quamar, Jaydeep Sen, Chuan Lei, and Vasilis Efthymiou. 2020. State of the Art and Open Challenges in Natural Language Interfaces to Data. In *Proc. of SIGMOD*. ACM, 2629–2636. https://doi.org/10.1145/3318464.3383128

[45] Studio R. 2021. R Markdown. https://rmarkdown.rstudio.com/index.html

[46] Donghao Ren, Bongshin Lee, and Matthew Brehmer. 2019. Charticulator: Interactive Construction of Bespoke Chart Layouts. *IEEE TVCG* 25, 1 (2019), 789–799. https://doi.org/10.1109/TVCG.2018.2865158

[47] Steve Rubin, Floraine Berthouzoz, Gautham J. Mysore, Wilmot Li, and Maneesh Agrawala. 2012. UnderScore: Musical Underlays for Audio Stories. In *Proc. of UIST*. ACM, 359–366.

[48] Diptikalyan Saha, Avrilia Floratou, Karthik Sankaranarayanan, Umar Farooq Minhas, Ashish R. Mittal, and Fatma Özcan. 2016. ATHENA: An Ontology-Driven System for Natural Language Querying over Relational Data Stores Diptikalyan. *Proc. VLDB Endowment* 9, 12 (2016), 1209—-1220. https://doi.org/10.4324/9780203932148

[49] Arvind Satyanarayan and Jeffrey Heer. 2014. Lyra: An Interactive Visualization Design Environment. *CGF* 33, 3 (2014), 351–360. https://doi.org/10.1111/cgf.12391

[50] Vidya Setlur, Sarah E. Battersby, Melanie Tory, Rich Gossweiler, and Angel X. Chang. 2016. Eviza: A Natural Language Interface for Visual Analysis. In *Proc. of UIST*. ACM, 365–377. https://doi.org/10.1145/2984511.2984588

[51] Tianze Shi, Kedar Tatwawadi, Kaushik Chakrabarti, Yi Mao, Oleksandr Polozov, and Weizhu Chen. 2018. IncSQL: Training Incremental Text-to-SQL Parsers with Non-Deterministic Oracles. *CoRR* abs/1809.0 (2018), 1–11. arXiv:1809.05054 http://arxiv.org/abs/1809.05054

[52] Ben Shneiderman. 1982. The Future of Interactive Systems and the Emergence of Direct Manipulation. *Behaviour and Information Technology* 1, 3 (1982), 237–256. https://doi.org/10.1080/01449298208914450

[53] Arjun Srinivasan, Mira Dontcheva, Eytan Adar, and Seth Walker. 2019. Discovering Natural Language Commands in Multimodal Interfaces. In *Proc. of IUI*. ACM. https://doi.org/10.1145/3301275.3302292

[54] Arjun Srinivasan, Bongshin Lee, Nathalie Henry Riche, Steven M. Drucker, and Ken Hinckley. 2020. InChorus: Designing Consistent Multimodal Interactions for Data Visualization on Tablet Devices. In *Proc. of CHI*. ACM, 1–13. https://doi.org/10.1145/3313831.3376782 arXiv:2001.06423

[55] Arjun Srinivasan, Nikhila Nyapathy, and Bongshin Lee. 2021. Collecting and Characterizing Natural Language Uterances for Specifying Data Visualizations. In *Proc. of CHI*. ACM. https://doi.org/10.1145/3411764.3445400

[56] Nicole Sultanum, Zoya Bylinskii, and Zhicheng Liu. 2021. Leveraging Text-Chart Links to Support Authoring of Data-Driven Articles with Vizflow. In *Proc. of CHI*. ACM. https://doi.org/10.1145/3411764.3445354

[57] Anh Truong, Floraine Berthouzoz, Wilmot Li, and Maneesh Agrawala. 2016. QuickCut: An Interactive Tool for Editing Narrated Video. In *Proc. of UIST*. ACM, 497–507. https://doi.org/10.1145/2984511.2984569

[58] Prasetya Utama, Nathaniel Weir, Fuat Basik, Carsten Binnig, Ugur Cetintemel, Benjamin Hättasch, Amir Ilkhechi, Shekar Ramaswamy, and Arif Usta. 2018. An End-to-end Neural Natural Language Interface for Databases. *CoRR* abs/1804.0 (2018). arXiv:1804.00401 http://arxiv.org/abs/1804.00401

[59] Bret Victor. 2011. Explorable Explanations. http://worrydream.com/ExplorableExplanations/

[60] Bret Victor. 2013. Drawing Dynamic Visualizations. http://worrydream.com/DrawingDynamicVisualizationsTalkAddendum/

[61] Haijun Xia. 2020. Crosspower: Bridging Graphics and Linguistics. In *Proc. of UIST*. ACM, 722–734. https://doi.org/10.1145/3379337.3415845

[62] Haijun Xia, Bruno Araujo, Tovi Grossman, and Daniel Wigdor. 2016. Object-Oriented Drawing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 4610–4621. https://doi.org/10.1145/2858036.2858075

[63] Haijun Xia, Nathalie Henry Riche, Fanny Chevalier, Bruno De Araujo, and Daniel Wigdor. 2018. DataInk: Direct and Creative Data-Oriented Drawing. In *CHI*. ACM, 223. https://doi.org/10.1145/3173574.3173797

[64] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *CoRR* (2017), 1–12. arXiv:1709.00103 http://arxiv.org/abs/1709.00103