


Authoring Combined Textual and Visual Descriptions of Graph Data

Shahid Latif , Kaidie Su, and Fabian Beck 

University of Duisburg-Essen, Germany
paluno—The Ruhr Institute for Software Technology

Abstract

The interactive linking of text and visualizations supports easy and guided exploration of information and results in a coherent document. Authoring such documents for the web requires writing custom HTML and JavaScript. Existing research aims at reducing the effort by providing a declarative syntax. However, these approaches either do not support the interactive linking of text and visualizations or require advance programming skills to establish this linking. Targeting a specific type of data i.e., graph data, we introduce an approach that uses a declarative syntax to produce interactive documents and requires little to no programming. Based on the user specifications in an HTML file, the system queries the database to retrieve subgraphs and link them to the relevant text fragments. The resulting document consists of a node-link diagram and text; the two representations are closely linked via interactions and word-sized graphics, and provide an active reading experience.

CCS Concepts

• **Human-centered computing** → **Visualization application domains; Information visualization;**

1. Introduction

Bret Victor's [Vic11] essay *Explorable Explainables* suggests facilitating an active reading process that allows readers to interactively explore the content. However, creating interactive documents is often tedious and requires writing HTML and JavaScript. Many researchers [VZK16] [CH18] [LLB18] have explored ways to simplify this process by using a declarative syntax or a markup language. However, existing systems that allow for easy and fast creation of interactive visualizations do not provide support for establishing interactive linking between text and visualizations. Other systems often require initial training and advance programming skills to produce non-trivial visualizations and interactively link those to text. In this paper, we present a focused approach—aiming solely at graph data—for authoring web-based interactive documents. The approach uses a simple declarative syntax and does not require any programming.

Graph data consists of entities and relationships between them. It is often difficult to find important information in a graph visualization having many entities and multiple types of relationships. These visualizations benefit from a closely coupled textual description that can provide contextual information. Interactive linking of the relevant parts of the visualization with the text helps in better cross-referencing the two representations. This linking also allows authors of the document to highlight important points and helps in bringing readers' attention to those points. Besides

such visualization–text interactions, the use of word-sized graphics (sparklines) [Tuf06] helps in achieving a more coherent document.

We present an approach that facilitates the process of authoring Web-based interactive documents. The approach aims at describing and visualizing multiplex graphs—graphs having several types of edges. The data is visualized as a node-link diagram with a force-directed layout. We use a declarative syntax that only requires specifications in an HTML file. Our system uses a graph database as an underlying data source, parses the user-specified parameters, queries the database, creates components of the document, and establishes a linking between the textual and visual representations. The resulting Web document offers an interaction-enriched reading experience; Figure 1 shows an example.

2. Related Work

Creating interactive articles involves designing interactive visualizations, producing reactive elements, and establishing visualization–text linking. High-level JavaScript libraries such as *React* and *Polymer* allow users to quickly create reactive components and a user-interface. *Tangle* is a JavaScript library that helps in building interactive documents. However, these libraries do not include support for creating visualizations. D3 [BOH11] is a JavaScript library for developing highly customizable visualizations. Although these libraries reduce the effort, users still need to write JavaScript and go through complex application code. Computational environments such as *Jupyter Notebook*, *R Markdown*,

and *Observable* are systems that aim at creating and sharing computations or graphics in a reproducible way. It is possible to create interactive content using these systems, but they focus on interactive coding experience and target technical users.

Existing research aims at providing ways to author interactive documents with ease and using minimal programming. Mavo [VZK16] introduces a syntax that is an abstraction of HTML and allows users to bind data to templated HTML. The use of a declarative syntax such as Markdown has made it even easier to write for the web. Visdown [Kap16] uses Markdown and Vega-Lite [SMWH17], a JSON syntax, to specify visualizations. However, these systems do not provide a mechanism of linking text and visualizations. Idyll [CH18] introduces a markup language combined with reactive programming in JavaScript for publishing interactive articles on the web. The focus of Idyll is broad and it allows building custom visualizations using D3 or Vega-Lite and binding them to the text. This generalizability comes at the cost of programming custom visualizations. In contrast, we only focus on interactive articles describing graph data which offers an easy-to-use syntax that does not require programming knowledge.

The possible ways and interactions for close integration of text and graphics (especially in the context of word-sized graphics) are discussed by Goffin et al. [GBWI17] and Beck and Weiskopf [BW17]. Latif et al. [LLB18] describe an approach for implementing some of those interactions using a simple syntax that only requires modifying the HTML. However, their approach is limited to simple visualizations, i.e., bar and line charts. We use their syntax as a basis, and extend it to produce documents for graph data having a comparatively complex visualization and two-way interactions between text and graphics as suggested in Beck and Weiskopf's work [Section 5.3] [BW17].

3. Interactive Documents for Graph Data

Using our approach, authoring interactive documents only requires editing an HTML file using a declarative syntax. Figure 1 shows the components of the generated interactive document, namely static text, interactive text (boldface and clickable), word-sized bar charts, and a node-link diagram.

3.1. Data and Visualizations

We use a graph database, *Neo4j*, which models the data as entities (nodes) and relationships (edges). The nodes and edges can have properties defined by key-value pairs. *Neo4j* uses the Cypher query language for managing the data. We assume that entities can have multiple directed relationships and the relationships have labels. We use a dataset that describes the characters of a famous TV series *Game of Thrones* as our running example throughout this section. The dataset consists of 84 characters and seven different types of 216 relationships among the characters.

To visualize the data, we have used the open source JavaScript charting library *D3*. The graph data is visualized as a node-link diagram with a force directed layout; Figure 1 shows an example visualization. The different types of relationships are encoded using distinguishable colors. Word-sized bar charts are produced using

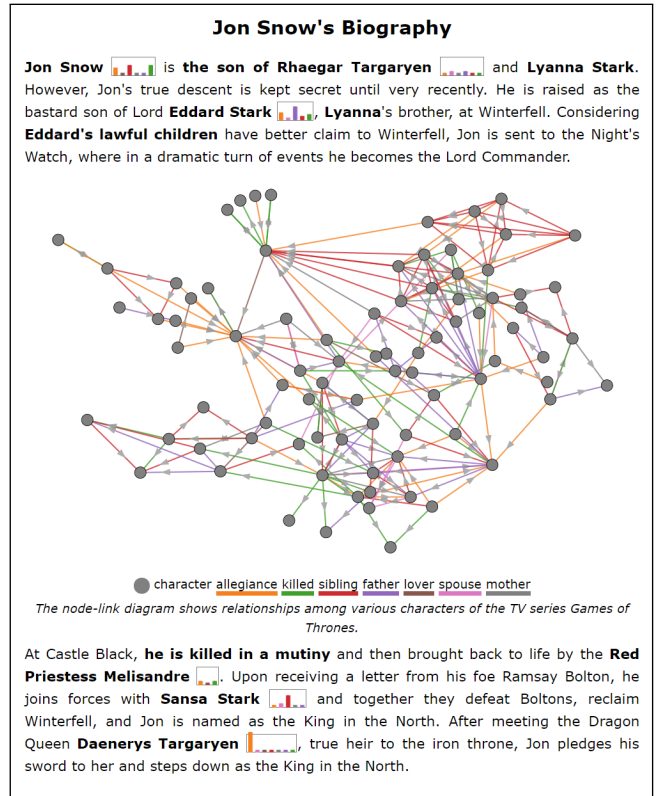



Figure 1: Interactive document describing the biography of a fictional character Jon Snow from the TV series *Game of Thrones*.

the jQuery plugin *jQuery Sparklines*. These small graphics reveal the degree of a node in the dataset with different types of relationships. For instance, the word-sized bar chart  following the character name Jon Snow in the first paragraph of Figure 1 shows that the character is linked to other characters with six types of relationships in different frequencies (number of edges per type).

3.2. Establishing Visualization-Text Linking

The linking between text and various parts of the visualization is defined as span tags in the HTML file. The user-defined parameters are specified using the custom data attribute option of HTML. The syntax of these tags is as follows:

```
<span class="linkedText"
      data-nodeID="ID" data-edgeSource="S"
      data-edgeType="R" data-edgeTarget="T"
      data-edgeDirection="both" data-cypherQuery="C">
text
</span>
```

All the span tags with class `linkedText` are parsed for the user-defined parameters. The `text` is produced as a clickable text and highlights the corresponding part of the visualization when clicked. The parameter values defined by the attributes are parsed and transformed into a Cypher query to retrieve the specified subgraph. Table 1 shows examples of various possible combinations of param-

Table 1: Custom parameters for querying the data. The check mark (✓) shows the presence of a string value and dash (–) means no value specified. The direction of the edge (*edgeDirection*) can be incoming (*in*), outgoing (*out*), or bidirectional (*both*). All other parameters accept string values. *S* and *T* are exemplary labels for source and target nodes.

Query	edgeSource	edgeType	edgeTarget	edgeDirection	Description
1	✓	✓	✓	<i>in</i> , <i>out</i> , or <i>both</i>	highlighting a specific relationship between <i>S</i> and <i>T</i>
2	✓	✓	–	<i>in</i> , <i>out</i> , or <i>both</i>	highlighting all the entities linked to <i>S</i> with a specific relationship
3	✓	–	✓	<i>in</i> , <i>out</i> , or <i>both</i>	highlighting all the relationships between <i>S</i> and <i>T</i>
4	✓	–	–	<i>in</i> , <i>out</i> , or <i>both</i>	highlighting all the entities linked to <i>S</i> with any possible relationship
5	–	–	–	<i>both</i>	highlighting the whole graph

eter values that can be used to filter the data for linking it to the text. It is not mandatory to specify the values of all the parameters; if not specified, the system uses default values. For example, the span tag `Jon Snow` will retrieve the subgraph containing all the nodes and edges associated with the node Jon Snow and link it to the text (Figure 3, A). The parameter *edgeDirection* has a default value set to *both*, but it is possible to specify *in* or *out* to restrict to only incoming or outgoing edges to or from a node. Although a small set of these parameters offers a selection of different types of subgraphs associated with one node, it does not support complex queries (e.g., compound queries, queries with conditions, etc.). To this end, the parameter *cypherQuery* allows specifying a Cypher query for retrieving more complex subgraphs from the dataset. For example, the parameter *cypherQuery* having the value:

```
"MATCH (c1:Character)-[r:killed]->(c2:Character) WITH a,
count(c2) AS kill_count WHERE kill_count > 3 RETURN c1, r, c2"
```

will retrieve a subgraph of all the characters who have killed more than three other characters in the TV series. In case both the *cypherQuery* and the other parameters for specifying the query are provided, the system uses *cypherQuery*.

Besides linking text to visualization, the entities (nodes) in the graph can also be linked to where they appear in the text. For this, the parameter *data-nodeID* is set to the label of that node. For example, the tag `Eddard's lawful children` links the node with label "Eddard Stark" to the text *Eddard's lawful children* (Figure 3, C).

The word-sized bar charts are defined using the span tag with class *sparkline* and setting the *edgeSource* to the label of the node. This will always send a Cypher query of type 4 (shown in Table 1) and return an array of edge types with the frequency of each relationship with the source node. The following span tag produces the first word-sized bar chart in Figure 1: ``.

3.3. Interactions

Through interactive linking, text and visualizations become an integrated unit. Our visualization–text interactions are adapted from

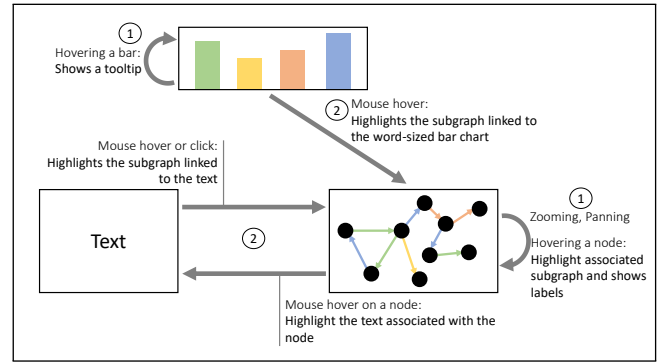


Figure 2: Interaction patterns between text, word-sized graphics, and the node-link diagram. (1) interactions influence only one component and (2) interactions connects two different components of the document.

the work of Beck and Weiskopf [BW17, Section 5]. Figure 2 explains the linking of text, word-sized bar charts, and the node-link diagram. The interactions marked as (1) only influence the entity which was interacted with. Interactions (2) describe the linking between two different components of the document. The *text* → *visualization* linking supports the standard reading strategy: reading the text first and exploring the visualizations while going through the text. The two-way interactions between text and the node-link diagram can support alternating reading strategies such as exploring the visualization first and then reading the corresponding parts in the text.

Hovering over the *interactive text* puts the linked part of the node-link diagram in focus by displaying the labels and reducing the opacity of the rest of the graph (Figure 3, A). This helps the reader to view the selected subgraph in context of the text and contrast it with the whole dataset. Clicking on the *interactive text* persists the selection and allows further exploration of the subgraph (Figure 3, B). Hovering over a word-sized bar chart highlights the relevant subgraph (similar to while hovering over *interactive text*) and hovering over each bar in it offers tooltips showing the name of the relationship type and its frequency (Figure 3, B).

The node-link visualization provides standard interactions such as zooming, panning, and resetting. In addition, when the user hovers over any node in the node-link diagram, it highlights the associated nodes and edges, and also shows their labels (Figure 3, C). If the hovered node has linked text, it gets highlighted with a yellow background.

3.4. Preliminary Feedback

To assess the ease-of-use of our system, we invited a bachelor student studying business informatics at our university. The participant was interested in writing an interactive article about the actor Tom Cruise. The participant was provided with a minimal version of the built-in *Neo4j* movie database. The dataset consists of three types of entities: actors, persons, and awards. These entities are associated to each other via ten different types of relationships.

The participant linked 13 text fragments to the graph visualiza-

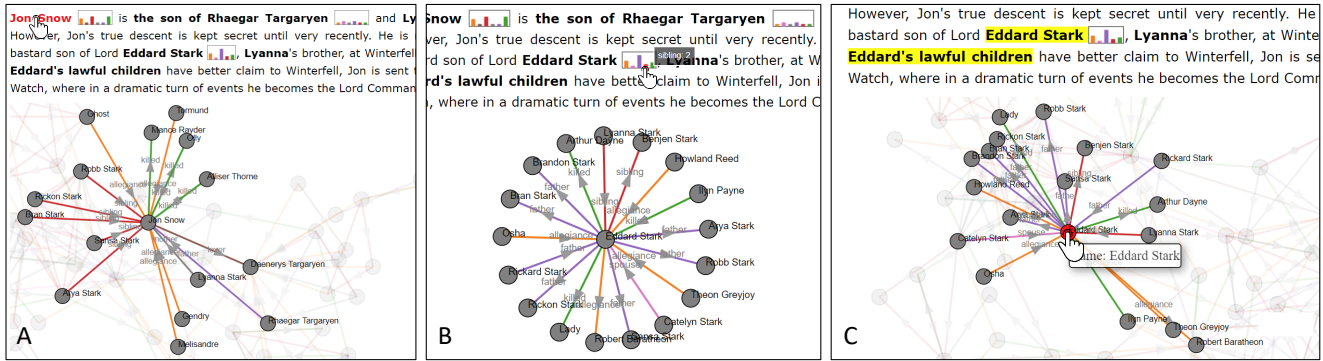


Figure 3: Supported interactions: (A) hovering over an interactive text, (B) effect of clicking on the interactive text “Eddard Stark”, hovering over a word-sized bar chart, and (C) hovering over a node in the node-link diagram.

tion using various query types from Table 1. In addition, the participant produced eight word-sized bar charts. After the participant was done, we asked questions about the ease-of-use and possible problems of the system. To the first, he responded, “This tool is quite simple to use, and people do not need the knowledge of SQL. The system is also understandable, and as a non-expert programmer, I [could] quickly get used to it”. The participant complained about filling in all the parameters in every span tag. It was still compulsory to define all the parameters in the version used by the participant. Based on the feedback, we have considerably simplified the syntax and made the user-defined parameters optional using default values. For instance, the query 4 (Table 1) can be specified by using only one parameter `text`.

4. Discussion

Our system allows for producing documents that describe graph data with only one type of visualization, i.e., a node-link diagram. Although our syntax scales to graphs having any number of nodes and edges, the visual clutter in the visualization is a bottleneck. However, it would be straight forward to replace the node-link diagram with another graph visualization. Our approach can also be extended to documents containing more than one visualization (of the same or different type) by including a unique identifier in the `` tags for the different figures. Moreover, the document layout to accommodate multiple visualizations in the interactive setting still remains an open question (e.g., relevant visualizations and text might get scrolled out). The support of more interactions such as brushing and linking between text and visualizations (drag-selecting a portion of visualization and highlight affiliated text) might be another useful extension.

5. Conclusion

We have presented an approach for producing interactive documents for graph data. It uses a simple syntax to establish the linking between text and various parts of the node-link diagram and do not require programming knowledge. The two-way visualization–text interactions not only provide an active reading experience but also support more explorative reading strategies.

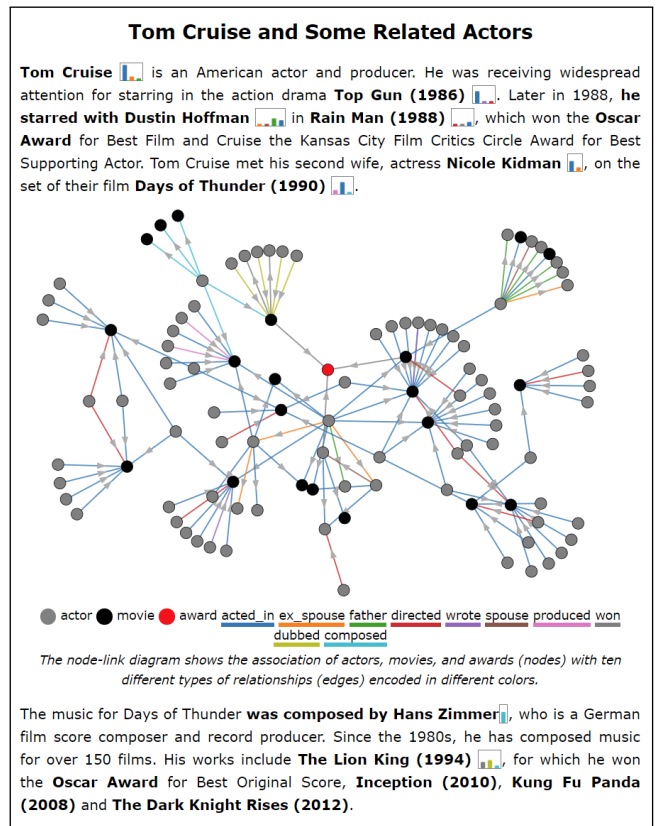


Figure 4: An interactive article authored by the participant during a preliminary feedback session. Using our system, only the HTML file was edited, and a variety of queries from Table 1 were used.

Acknowledgments

Fabian Beck is indebted to the Baden-Württemberg Stiftung for the financial support of this research project within the Postdoctoral Fellowship for Leading Early Career Researchers.

References

- [BOH11] BOSTOCK M., OGIEVETSKY V., HEER J.: D3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (Dec 2011), 2301–2309. doi:10.1109/TVCG.2011.185. 1
- [BW17] BECK F., WEISKOPF D.: Word-sized graphics for scientific texts. *IEEE Transactions on Visualization and Computer Graphics* 23, 6 (2017), 1576–1587. doi:10.1109/TVCG.2017.2674958. 2, 3
- [CH18] CONLEN M., HEER J.: Idyll: A markup language for authoring and publishing interactive articles on the web. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (2018), UIST '18, ACM, pp. 977–989. doi:10.1145/3242587.3242600. 1, 2
- [GBW17] GOFFIN P., BOY J., WILLETT W., ISENBERG P.: An exploratory study of word-scale graphics in data-rich text documents. *IEEE Transactions on Visualization and Computer Graphics* 23, 10 (2017), 2275–2287. doi:10.1109/TVCG.2016.2618797. 2
- [Kap16] KAPOOR A.: Visdown. <https://visdown.com/>, 2016. Accessed: 2019-01-13. 2
- [LLB18] LATIF S., LIU D., BECK F.: Exploring interactive linking between text and visualization. In *EuroVis 2018 - Short Papers* (2018), The Eurographics Association. doi:10.2312/eurovisshort.20181084. 1, 2
- [SMWH17] SATYANARAYAN A., MORITZ D., WONGSUPHASAWAT K., HEER J.: Vega-lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics (Proc. InfoVis)* (2017). 2
- [Tuf06] TUFTE E. R.: *Beautiful Evidence*, 1st ed. Graphics Press, 2006. 1
- [Vic11] VICTOR B.: Explorable explanations. <http://worrydream.com/ExplorableExplanations/>, 2011. Accessed: 2019-02-12. 1
- [VZK16] VEROU L., ZHANG A. X., KARGER D. R.: Mavo: Creating interactive data-driven web applications by authoring html. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (2016), UIST '16, ACM, pp. 483–496. doi:10.1145/2984511.2984551. 1, 2