G21 – 8-9

The Path to Web-based Visual Network Exploration

Dávid Debnár, Georgi Bonev, Sasha Gielis, Theo de Leeuw, Julian Dziegielewski and Jeroen Oerlemans

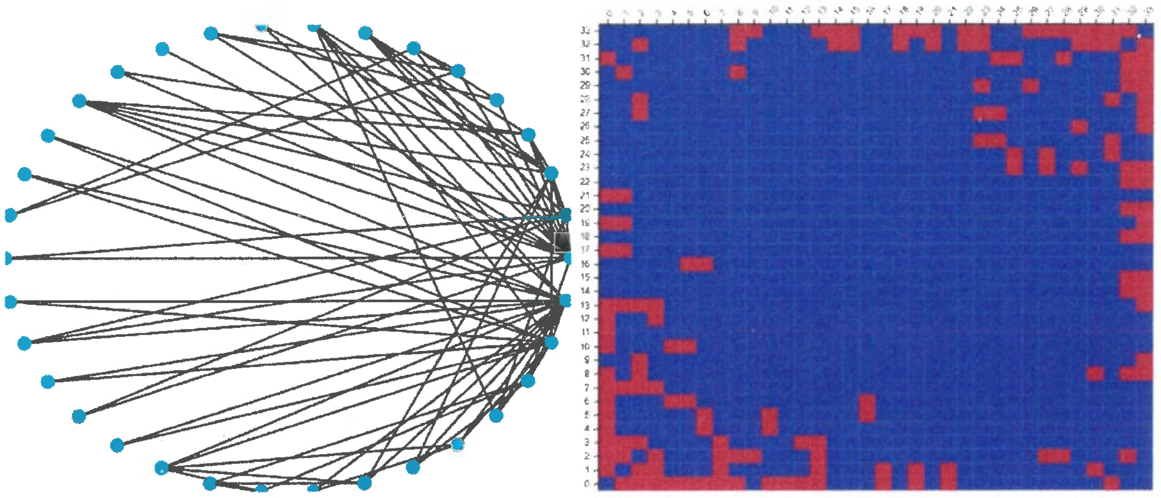

Fig. 1: Both supported views, the node-link diagram using a circular layout on the left and adjacency matrix on the right. → which data?

Abstract— Data exploration is best facilitated by the synthesis of algorithm-based representations and human-computer interaction, multiplied by the mobility of information gained through online collaboration and sharing of results. Given many emerging real-world systems reduce s to networks that impact daily life and decision-making, an open and accessible approach to information visualization is required. ~~The research~~ hence describes the techniques used in developing a web-based, interactive data exploration and visualization tool for ~~directed~~ network ~~graphs~~s. Drawing on ~~research into human~~-computer interaction for information visualization, and algorithms for representing network ~~graph~~s, specific choices of the tool design are discussed. The supported data format, for upload of datasets to be analyzed, is an adjacency matrix of weighted edges, based on the most wide-spread conventions. Networks are visualized using two metaphors, the node-link diagram, and the adjacency matrix. Interactions, as categorized by user intent, include exploration by pan and zoom, filtering, reconfiguring of elements in views, selection, and consequent, highlighting of connected elements. Users can reconfigure the vertices of the node-link view into a circle. This is achieved using the Bokeh library for Python deployed as an embedding into a Flask website application. The aim of this paper is describing techniques used in deploying a tool which can further expand the software landscape of online interactive data exploration.

2

Index Terms—Information visualization, ~~directed~~ Networks, Interaction, Collaboration, Visual exploration.

1 Introduction

Expanding the ability to visualize network data is an ever-increasing need in the era of being surrounded by various networks. Every system with a flow of information can be described by a network, from brain-connectivity related to Alzheimer's disease [10], to semantic networks in linguistics, or social networks of societies. Hence, a versatile tool can have far-reaching applications. With a personal computer in every pocket, one might be inclined to explore data of the physical or digital networks they are part of even outside academia, for their own daily decision-making.

The big-data collection has only been rising in recent years, and though paralleled by the increase in applications of machine learning or other statistical-learning methods, the human brain, when aided by interactive tools, still leads in terms of pattern-recognition [5]. Given the difficulty of anticipating possible uses for in-flowing data, the decision to gather is often made before intent of analysis, and so network data, among others, is being over-collected and under-analyzed.

Visual representations of network data coupled with interaction techniques, allow for a deeper understanding of behavior within networks, but often require a steep learning curve, are limited to simple networks or single-purpose [11]. Scaling to networks with a high vertex-count or with complex, dense connectivity is a key requirement when considering applications in areas like brain-imaging and social-networking. The analysis of networks is therefore bounded by the entry-requirements and the subsequent iteration-speed of individuals. 2

The paper will describe the techniques used in developing a web-based, interactive information visualization (InfoVis) tool for the exploration of ~~directed graph~~ networks. It aims to describe InfoVis approaches harnessed by the capability to share results via web-based tools, which open the field of data analysis to a wider range of actors and audiences thus giving rise to more opportunities for the synthesis between the data gathered and human brains available as pattern-recognizing machines.

2 Related Work

Visualization as a graph, network, or adjacency matrix is an integral part of data and computer science [3]. It is used in applications and domains [6].

Just like people ~~that~~ who are interacting with each other create networks, computer systems have the same relation between co-existing functions in the program. There are a countless number of examples in our world, all of which have a common structure of relations between themselves. They can be distinguished by their forms. When the relation holds in one direction we call it directed and if a relation is both-ways it is not directed. Although in graph theory we have edges as our relations and vertices which have relations between themselves when focusing on a visual representation of the data we refer to them as to nodes (vertices) and links (edges). This type of visualization is called a node-link diagram.

It has been invented by Leonard Euler when he was solving 'The Seven Bridges of Königsberg' problem [8]. Euler's discovery is considered to be the first theorem of graph theory. Even though during Euler's time it was the best visualization for this data, these days the node-link diagrams are not efficient due to constantly enlarging datasets which cause visual clusters of link crossings [7]. ?

There are visualization methods that resolve these problems. One of them is the adjacency matrix. It is a square matrix whose elements indicate whether pairs of vertices are adjacent or not. This type of graph eliminates line crossings and overlapping nodes. However, while using the adjacency matrix we have to first make an ordering of the data, otherwise, there would be a visual clutter effect.

text here

3 DATA

3.1 Data Format

As mentioned before, the type of data that will be visualized with this tool is network data. Within this data type, there can be some variations. Directed networks and weighted networks are network types one might encounter.

A directed network is a network with relations that may only hold in one direction. For example, Alice might have sent a message to Bob, but Bob might not have sent a message to Alice. Alice is then related to Bob, but Bob is not related to Alice.

A weighted graph is a ~~network~~ graph where the relations have weights. A relation between one pair of nodes might have a higher weight than another. To give an example, Alice might have sent ten messages to Bob and only one message to Charlie. The relation between Alice and Bob has a higher weight than the relation between Alice and Charlie.

There are several formats that one can use to represent all of these networks. In this section two of them will be discussed, and we will see which of the two formats was chosen to store the data to be visualized in.

The first format that will be discussed in this section is a compact one. It involves simply writing down the relations in a list:

A,B		A,B,C
A,C	or even more compact:	B,A
B,A		$\vdots$
$\vdots$		

The first node is the source of the relation, the other node(s) is the target of the relation. Weights can be implemented, for example, by repeating the same relation multiple times. Storing networks this way is very space efficient, however, it is not the easiest to work with. This is because one has to extrapolate how many nodes are included in the network. Therefore, it was chosen not to go with this format.

Another way of storing network~~graph~~s is by using a so-called adjacency matrix:

↓nodes→	A	B	C
A	0	1	1
B	1	0	0
C	0	0	0

Each source and target combination is displayed, however, only the cells where the cell is nonzero are included in the network. To implement weights, simply change the cell from a 1 to its weight. This format is less space efficient than the other one, however, it is easier to work with. This is because it is known beforehand how many nodes to include in the network. It has thus been decided to use this format to store the data in. Coincidentally, the datasets that were provided were formatted this way already.

3.2 Data Upload

To be able to create several visualizations from a dataset, the text in the uploaded file needs to be written in a specific format. The required format is the format of a comma-separated values (CSV) file, which stores tabular data in plain text. A CSV file contains the values of the cells of the table, separated by commas or semicolons. As already explained in Section 3.1, this format of data represents an adjacency matrix. The visualization tool is able to work only with datasets that are written in this specific format, so after the user has uploaded a file, the file will only be accepted if it contains a .csv extension. Since the provided datasets already have the correct format, uploaded files do not have to be processed before creating the visualizations.

Assuming the user has a dataset in the required format, it will be uploaded to the server. Once the upload process is finished, the filename will be displayed in the 'Uploaded Files' section. The upload functionality is achieved through a library called 'Flask' [1], in conjunction with the 'os' module [2]. The former of these is a web framework written in Python. It allows for full website functionality management. The os module is used for file management and interaction on the server. The upload page interface is written in HTML. The HTML code is called by Flask, which can also pass custom arguments to the HTML code. When the 'Upload' button is pressed, the HTML code calls on a specific part of the Flask code. Using the os module, a list of files on the server is supplied to the HTML template, so that these can be displayed in the 'Uploaded Files' section. ?x

4 VISUALIZATION TOOL

The library that is used by the tool to create the visualizations is called 'Bokeh' [4]. Bokeh is a library for Python and has a wide set of options for visualization. It not only allows for graph drawing and matrix creation but also allows for built-in interactions. The interactions that are implemented into this visualization tool will mostly consist of Bokeh's built-in interaction features. ?x ?x

4.1 Bokeh's Interface

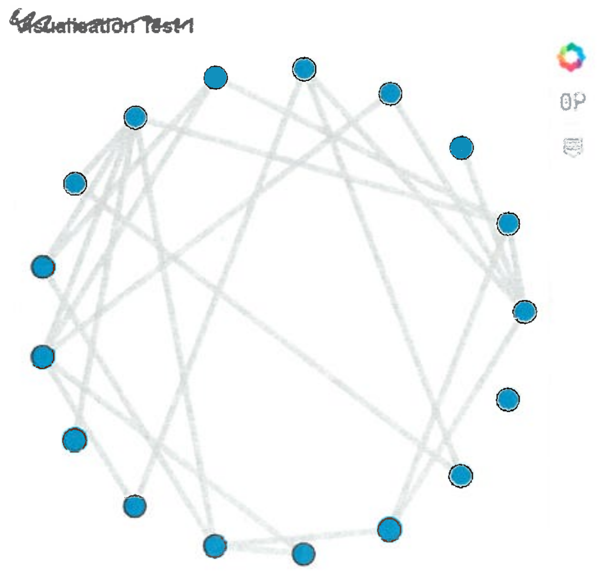

Fig. 2: Node-link diagram of a specific dataset. — more text —

Bokeh uses a special interface. The node-link diagram in figure 2 is the representation in Bokeh. On the right of the node-link diagram, we see an interface consisting of three icons, two of which are interactive tools provided by Bokeh. The top icon is the Bokeh logo itself, redirecting to Bokeh's website. In this example, a hover tool and a wheel zoom tool have been added to the interface.

Let us consider the hover tool again to explain the interactive tools.

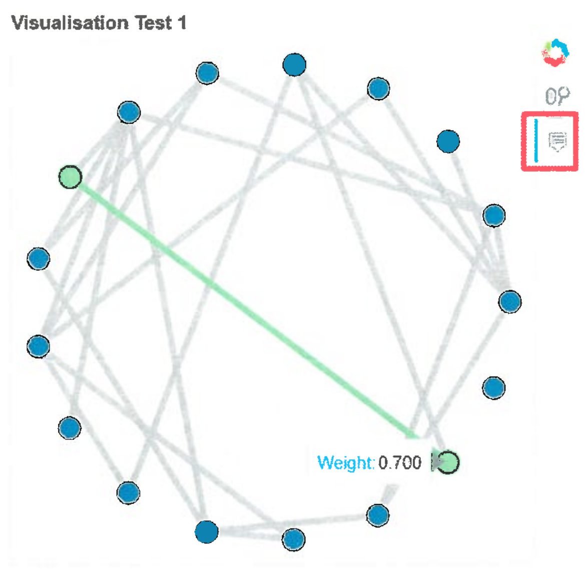

Fig. 3: Node-link diagram when using the hover tool.

As can be seen in figure 3, hovering over an edge or node will display what the developer of the tool wants to be displayed. We can add these tools easily. Suppose a variable *G* is set to be a HoloViews graph [9], then this is how we set the hover tool:

```
import pandas as pd
import numpy as np
import holoviews as hv
from holoviews import opts
hv.extension('bokeh', 'matplotlib')

h = hv.Graph()
opts.Graph(tools=['hover'])
```

Now there is a Bokeh interaction implemented for our visualization, but there are many more tools that can be used. Our tool uses the hover tool, as shown above in figure 3, which allows you to hover over edges, nodes, and the adjacency matrix. Another implemented interaction is the box select tool, which allows for selecting an area in the visualization resulting in selecting the corresponding edges and nodes in the node-link diagram or fields in the adjacency matrix.

4.2 The tool's Interface

As for the tool's interface, a toolbar is present which will be customized later to fit a user's wishes.

The toolbar on the left in figure 4 will be representing the interface, however, these buttons do not have any functionality yet. Again, on the top right of the adjacency matrix in figure 4 is the Bokeh interface.

5 Interactions

Among the main goals of this tool is facilitating exploratory data analysis. While ordering algorithms provide a necessary basis for data exploration in the form of representations, they are only part of a continuous back-and-forth process between a human user and the data analysis tools, compelling the need for further interactions in a data

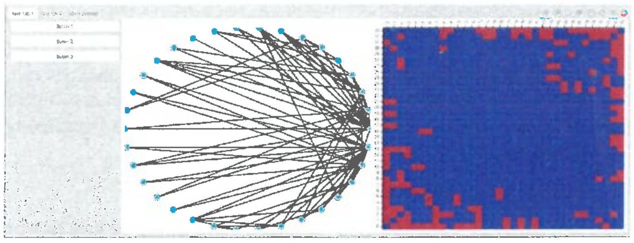

Fig. 4: The tool's interface for the visualizations.

exploration tool. Interactions considered in this tool can be categorized based on the model of notion of user intent, which include *filter*, *explore*, *select*, *reconfigure*, and *connect* [12].

As previously discussed, the graphical user interface of the visualization consists of a tabbed interaction panel docked to the left and two views side-by-side, each with their own visualization. The view on the left is a node-link diagram, which encodes relations of vertices and edges, and adds a spatial channel in the form of position of nodes, and by extension position of links. The view on the right is an adjacency matrix, which encodes edges as squares on a grid, and vertices as row and column labels. Both views share source data, which is reflected in several interactions, but the rest operate independently.

Filtering tools modify the source data loaded in memory, and as such are reflected on both views simultaneously. These include taking a random subset of vertices and hence edges prior to visualizing, for exploring large datasets dynamically, if limited by performance when viewing the entire network.

General exploratory interactions are pan and zoom, essential for discovering patterns at different layers of emergence, as well as making sense of large or complex networks in detail. These are provided as part of tools instantiated in Bokeh plots out-of-the-box, specifically panning a view by clicking and dragging over a view using a pointer, activated by default, and zoom using a rectangular selection. Further, the mouse scroll-wheel zoom Bokeh tool is instantiated, providing a more sensitive zoom alternative.

For selection, box select, and hover is available as part of Bokeh's toolset. Box select allows a user a rectangular selection of the nodes channel in the node-link diagram and edge channel in the adjacency matrix. The selected items are highlighted by a different colour and remain highlighted until another selection is made. The hover tool, activated by default, displays information about the underlying item in a floating bubble, e.g. vertex labels or edge weights. For each selected node in the node-link diagram, the connected links are highlighted as well, hence selection in this network visualization is inherently linked with finding connections.

6 Application Example

The goal of this visualization tool is to aid users in discovering more about their data. Network data can be messy to analyze, due to the many possible relations. This tool aims to make it simpler to do so.

A user may be wanting to find out which relation has the highest weight, which they can do using this tool. The adjacency matrix visualization combined with the hover tool makes finding the node with the most relations easy. The heaviest relation will have a reddish color, and by hovering over it the user can find out the exact weight of the relation. It also shows the index of the associated nodes, which makes finding the relation in the node-link diagram easier. This technique could be used in finding out whether the node with the heaviest relation is also well connected.

Finding out the node with the most relations is also something this tool allows for. In the node-link diagram, this node will have the most links going to that node. Users can click on this node to select it, which will make all the connected links stand out more by graying out the rest of the links. Once users know which node has the most relations, they can look for its index in the adjacency matrix visualization, where they can learn if all those relations are of high weight, or if the relations are all very light in comparison to the relation with the highest weight. This information could prove useful, for example, in analyzing a friend group, with the relation weights meaning the number of messages sent to each other. If the person that is messaging the greatest amount of people has a lot of relations of relatively high weights, this is probably a very popular person.

7 Discussion and limitations

This visualization tool has some limitations and therefore can still be improved. In this section, several limitations for some main parts of the tool will be described.

7.1 Performance

After the user has selected the file that is going to be visualized, the dataset is processed and after some time the visualizations will show up. When the dataset is very large, this waiting time can take up to a minute, which is lengthy.

This tool is an interactive visualization tool, meaning that the user should be able to obtain insight into a specific dataset using certain interactions. These interactions make the tool powerful but are also likely to influence its performance. This is another point of improvement because, after the visualizations of a large dataset are displayed on the screen, the tool responds quite slowly to the interactions. This problem is also related to the size of the selected dataset. To make the tool truly interactive, it should nearly instantly respond to commands by the user.

7.2 File Upload

When uploading a file to the tool, only CSV files will be accepted and all files containing a file extension other than .csv will not be uploaded. Even when the dataset in a non-CSV file is written in the correct format (and hence the tool will be able to process the data), the file will not be accepted and no visualizations will be created.

All CSV files uploaded by the user will be saved and their filenames will be listed vertically. By clicking on one of them, the corresponding dataset will be visualized. After restarting the server, the uploaded files will still be available on the website meaning that they are saved on the server, which is a problem because all people using the tool would be able to see the data uploaded by other users.

7.3 Visualizations

The tool supports a minimal amount of layouts for the visualizations. The only possible layout for the node-link diagram is radial/circular and the adjacency matrix cannot be reordered. Obtaining insight from only one view of a visualization can be difficult because some patterns in a dataset might not be clearly identified in a standard layout. These patterns will perhaps appear when using a different layout for the node-link diagram or after reordering the elements in the adjacency matrix.

7.4 Graphical User Interface

The graphical user interface is very basic. This is probably the least important limitation of the visualization tool since a basic GUI does not necessarily make the tool poor to use.

8 Conclusion and further work

This paper's purpose is to describe our web-based visualization tool, as well as everything behind it, in great detail. It explains how a dataset is accepted and transformed into an interactive visualization with two different overviews: a node-link diagram and an adjacency matrix. A lot of attention has been paid to the structure, implementation, and functioning of the application, and many of its purposes have been mentioned. Despite reaching a lot of the goals that were originally set, the tool is far from perfect and there is a lot of room for improvement.

For starters, only datasets of a specific file type and format are accepted hence adding more flexibility to the data upload would be beneficial. Visualizing with custom colors is another feature that is not supported. Different colors would make setting apart the data easier in addition to helping people with color vision deficiency. Lastly, the user may actually need to do a lot of work in order to obtain a visualization that provides information. Hence automatic filtering and suggestions that could aid the user would save time while making the tool easier to work with. While there are great ideas for further development it should be noted that what has been mentioned barely scratches the surface of what is possible.

Acknowledgments

References

[1] Flask (a python microframework). `http://flask.pocoo.org/`. Date accessed: May 2019.

[2] os - miscellaneous operating system interfaces. `https://docs.python.org/3/library/os.html`. Date accessed: May 2019.

[3] M. Burch and D. Weiskopf. An interactive visualization tool for dynamic graphs, node ...

[4] B. contributors. `https://bokeh.pydata.org/en/latest/`, Apr 2013.

[5] D. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):18, Aug 2002. doi: 10.1109/2945.981847

[6] T. v. Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. v. Wijk, J. Fekete, and D. Fellner. Visual analysis of large graphs: Stateoftheart and future research challenges, Apr 2011.

[7] R. Rosenholtz, Y. Li, Z. Jin, and J. Mansfield. Feature congestion: A measure of visual clutter, Jun 2006.

[8] T. Rz. Mathematical explanations in euler's konigsberg, 2014.

[9] J.-L. Stevens, P. Rudiger, and J. A. Bednar. Graph .

[10] F. Vecchio, F. Miraglia, F. Piludu, G. Granata, R. Romanello, M. Caulo, V. Onofrj, P. Bramanti, C. Colosimo, P. M. Rossini, and et al. "small world" architecture in brain connectivity and hippocampal volume in alzheimer's disease: a study via graph theory from eeg data, Mar 2016.

[11] M. Xia, J. Wang, and Y. He. Brainnet viewer: A network visualization tool for human brain connectomics. *PLoS ONE*, 8(7), 2013. doi: 10.1371/journal.pone.0068910

[12] J. S. Yi, Y. A. Kang, and J. Stasko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):12241231, 2007. doi: 10.1109/tvcg.2007.70515

[illegible]

more references

6/1

General Problems When Writing the Interim and Final Reports

Writing Quality:

X W1 Capitalize section, chapter, table, equation, figure ==> Section, Chapter, Table, Equation, Figure if it is followed by a reference number

W2 Typos

X W3 Many typos

W4 Really many typos

W5 Unacceptably many typos!!!

X W6 Look for consistency

X W7 Avoid textual redundancies

W8 Grammar problems

W9 Text not understandable

W10 Make paper ready for being published/submitted

X W11 Avoid same word occurring 2, 3, 4, or 5 times in the same sentence or placed next to each other

W12 Do not say: don't, won't, can't ==> say: do not, will not, cannot and so on

X W13 Do not be too wordy, come to the point earlier ==> Keep it short while still informative!

W14 If you have problems, ask a native English speaker or ask ME! (m.burch@tue.nl)

Structure and Layout:

X S1 Find a better structure of the paper

S2 Better balance of the sections and subsections required

S3 Take the correct template (fonts, layout, no diamondrule limiter, and so on)

X S4 Layout of the paper (avoid empty gaps, single words or single lines on a new page and so on)

S5 No single lines as a whole paragraph

S6 Avoid a subsection numbering only going to 1, like there is Section 6.1, but no Section 6.2

S7 Use correct reference style

S8 Add references for figures and research papers

S9 Place figures in the text (at the page where they are referenced for the first time)

X S10 Consistency, e.g., web-based vs. web based and so on

X S11 Consistency in capitalization, python vs. Python, java vs. JAVA and so on

X S12 Consistency in the references (use bibtex)

S13 Make figures fit on the width of the paper

S14 Avoid footnotes, put it to reference section

S15 Chose the correct citation style

Content:

C1 Not enough text/pages

X C2 Not enough figures

X C3 No text between section and subsection 3 + 3.1

C4 Better motivation in the abstract and introduction

X C5 Look for reproducability!!! A 1st year PhD student of computer science should be able to reproduce your work by just reading the final report

C6 Produce 3 pages of text and 1 page of figures (interim report)

X C7 Produce 6 pages of text and 2 pages of figures (final report)

X C8 Longer application example

X C9 If you have an algorithm, write enough content about the algorithm, e.g., add pseudo code and explain the steps

C10 If you have an algorithm, talk about runtime complexities in enough detail

C11 Make abstract longer, it is a summarization of the paper content

C12 Write a conclusion

C13 What is future work? More future work!

C14 Add your own names

C15 Find a better title, be creative!

X C16 For interaction techniques add some illustrating figures

Figures:

F1 Teaser too large

F2 Teaser too small

F3 Figures are blurry

F4 Figures have no legends

X F5 GUI screenshot full size

X F6 Annotate the GUI with components and describe them

X F7 Add an adequate number of illustrating figures to the sections

F8 Make figures of high resolution

F9 Use labels for the figures

F10 Make figures readable

F11 Avoid too many figures

F12 Add captions to the figures

X F13 Add longer/better captions to the figures

F14 Make figure captions shorter

F15 Figures are self-explanatory, captions should not be too long, but also not be too short

X F16 Show the GUI/webpage with visualization examples (in full size)

F17 Use the real-world dataset, any good reason not to use it?

F18 Annotate figures with numbers or capital letters and use those to describe them in the text

F19 Teaser figure should be horizontally longer than vertically and only up to one fifth of vertical page space

F20 Take your own teaser, take the best visualizations that you have (most aesthetically appealing)

62n

Implementation Details:

I1 Which programming language have you chosen?

I2 Which environments?

I3 Which libraries/frameworks?

✗ I4 Which components does your software have?

✗ I5 How are the components connected? Do they interplay? Show this process! UML? Graph diagram?

✗ I6 How did you get the server running? Which kind of server is this?

✗ I7 Any limitations in the implementation process?

Algorithms:

A1 Use pseudo code to explain algorithms

A2 Use input and output parameters

✗ A3 Talk about runtime complexities

Previous/Related Work:

P1 No related work (as separate section)

P2 Related Work/Papers not compared/discussed with new approach

✗ P3 More related work

References/Citations:

R1 Where are the figures from? Source? No reference!

✗ R2 References inconsistent

R3 References need proofreading

R4 Missing references in the text

R5 If you take text (or figures) from someone else, reference it! (no plagiarism)

✗ R6 Longer list of references

Special Extensions (only for this specific topic):

E1 Talk about previous/related work

E2 Talk about the data

E3 Talk about your tool

X E4 Talk about implementation details

X E5 Talk about the workflow

E6 Talk about visualizations

X E7 Talk about interactions

X E8 Talk about an application and which insights you find

X E9 Talk about scalability issues/performance issues/runtime complexities

X E10 Talk about data handling issues (data too large to be stored and so on)

X E11 Talk about the challenges and limitations

E12 Conclude the paper and talk about future work (do not tell in the future work what you really plan to do, tell what could be done but what you would never do!)

X E13 Talk about matrix reordering and algorithms for that

X E14 Talk about node-link layouts and algorithms for that

Others:

X O1 Add a workflow figure, a pipeline showing how the components are built on top of each other and how they are connected