Linnæus University
School of Computer Science, Physics and Mathematics

Degree project

Visualization of Relationships in Clustered Text Data

Author: Liu Jiayi
Date: May 26, 2013
Subject: Computer Science
Level: Master
Course code: 4DV01E

Abstract

There exist lots of text data. They are often extracted into topical tags for indexing. They are also connected by relationships like sharing the same authors, created at the same period. The entirety of above yields a big text network. In this thesis, a visualization tool to reveal information from big text data in the network is introduced. With clustering algorithm, the data are grouped. The groups with tag clouds show the overview of dataset. Edge halo, a new approach for bundling edges, represents the relationships of text data in and between the groups. An application prototype was developed to visualize clustered text data with their relationships and give an overview of the network in one view.

Keywords: information visualization, text, clustering, relationship, networks, tag clouds, edge bundling

Acknowledgements

Foremore, my gratitude goes to my supervisors, Prof. Dr. Andreas Kerren, Björn Zimmer and Ilir Jusufi, who guided me into the field of information visualization, a place full of treasures, wonder and magic. Their vision, enthusiasm and spirit will always inspire and enlighten me in a bright way. Also, my thanks extend to my parents and my family members for supporting me.

Contents

List of Figures

1 Introduction

Data containing information are around us and everywhere. Since data increase sharply, people realize that it is important to find a way to discover and understand them. One of the goals of information visualization is to make data visible and comprehensive. It is a process that transforms abstract or conceptual information into visual geometrical shapes [2] [1] [3]. Through human's visual system, the visualized data are convenient to be watched and analyzed [4]. It helps people who would like to get involved into a data set quickly such as searching, exploring and analyzing.

Nowadays, there are various data. Often they should to be visualized, due to their huge amount, especially in biology, climate, astronomy and so on. Text data are also enormous and people deal with them almost in every second. There exist a lot of text data, such as on web pages, in books, magazines and newspapers.

Knowledge is contained in text data, especially in documents [5]. Today the Internet is open and rich. People use search engines like Google to seek for information. Typing some keywords into an engine and getting a list of documents, they read summaries of the hit documents and fortunately can find the right ones [6]. To avoid verbose abstract, documents are often extracted into topical tags with a few words to summarize the contents [7]. Tags are helpful for classifying documents as well. Two documents sharing the same or similar tags can be put into the same group. Besides their contents, documents hold kinds of metadata such as author, creation time, or bibliography. Relationships between documents can be made based on those metadata. For example, two documents share the same authors; two documents are created at the same period; also a document makes reference to another one and so forth [7]. Then documents with tags and relationships can yield a big network of documents for indexing. We would like to visualize this network to put the clustered text data with their relationships in a clear overview.

Unfortunately, the details of clusters and relationships between clusters are usually visualized separately. Visualizing article relationships in a traditional node-link diagram does not display the internal tags for documents. A view to show only the detail of a specific clustered group does not support revealing the relationships for articles of other groups. Thus we would like to find a way to combine node-link diagrams and a view of the attributes of all clustered groups, to reveal the details of groups and document relationships.

1.1 Goal and Criteria

The Visualization of Clustered Text Data is a proposed master project under the ISOVIS group. Our interest as well as the goal is to build an application prototype to visualize the network with tags and relationships of documents. It can help users to get an overview of a collection of documents and to find interesting relationship patterns. In order to achieve this goal, the following features should be targeted:

- Read data from a file with tags and relationships of documents, cluster the documents by their tags and build a network of documents.
- Provide a clear overview of the network with some interactions to help users explore clustered documents and their relationships.

1.2 Thesis Outline

This report is organized into 5 chapters. Chapter 1 indicates the motivation and the goal of this project. Chapter 2 examines basic notations and background knowledge. It also surveys a few related works. Chapter 3 covers the design and implementation process for developing the prototype. Together with Chapter 4, which addresses the developed result and some discussions of use cases, these two chapters constitute the main part of this report. The last chapter concludes the whole project and provides some ideas for future work.

2 Related Work and Background

The chapter will begin with a few related works to get some inspirations and general ideas for the prototype development. Then some basic knowledge applied in this project will be mentioned. It introduces details about the visualization pipeline, data structure for text data, visualization rendering techniques and several human-computer interaction patterns.

2.1 Related Work

The aim of this project is to build a prototype to visualize the network of documents discussed later in Section 2.4. A network contains an underlying graph. There are also some attributes attached to the graph. Therefore drawing the nodes and edges in the graph with revealing their attributes is a main goal. In addition, the edges are for relationships and clusters. The relationship edges are pairs of nodes while the cluster edges are sets of nodes. They should be visualized in different ways. These two kind of edges will be discussed in Section 2.4 also. In the following related works and techniques used in our tool are discussed.

2.1.1 Tag Clouds

An impressive application generating tag clouds is Wordle [8] (see Figure 2.1). However, the clouds are usually large and occupy a lot of space. If we would like to show several clouds, then it requires a big screen to contain them.

ManiWordle presents an application based on Wordle with additional interactions [9]. The users can change the position of a tag. Then the application searches for new correct positions for other tags by a special spiral path. It adapts to the change and refreshes the view dynamically.

Another interesting project presents the application with euler diagram to show several tag clouds in one view [10] (see Figure 2.2). The authors considered that some related tags can be put into groups and be shown as sets.

Figure 2.1: A tag cloud generated by Wordle [8].

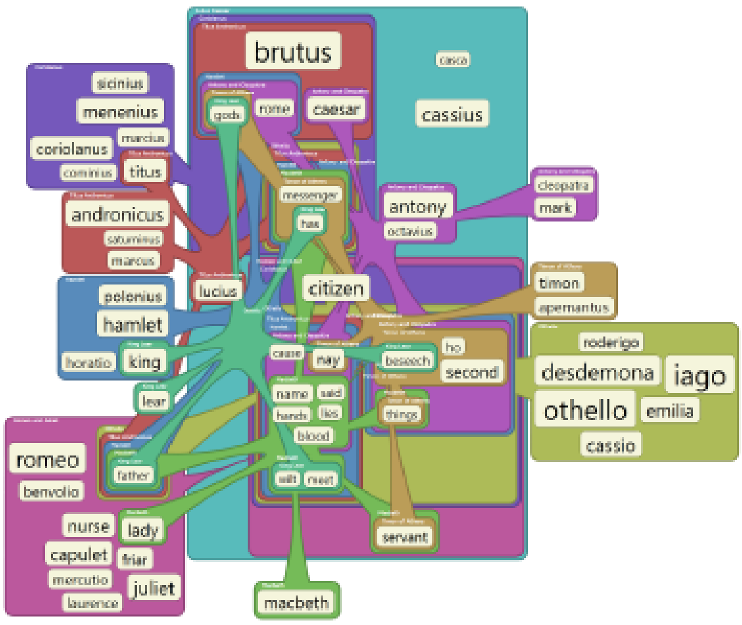

Figure 2.2: Tag clouds with their relationships are represented via Euler diagram [10].

However, sometimes the tag clouds with their relationships are overlapping, introducing undesired clutter.

2.1.2 Edge Bundling

Holten [11] addressed a method for edge bundling. The method is to rebuild a prepared graph with nodes and edges into a tree like using K-nearest neighbors algorithm [12], and then to route the paths between the nodes through the tree. It bundles the edges by paths and paints them with a parameter of density. Balzer and Deussen [13] provided another solution to visualize a graph (see Figure 2.3). They used large and complex clustered graphs. Their visualization presents a continuously adaptive and adjustable view. By zooming in, the users can discover the graph in dynamical views that range from rough to detailed representations.

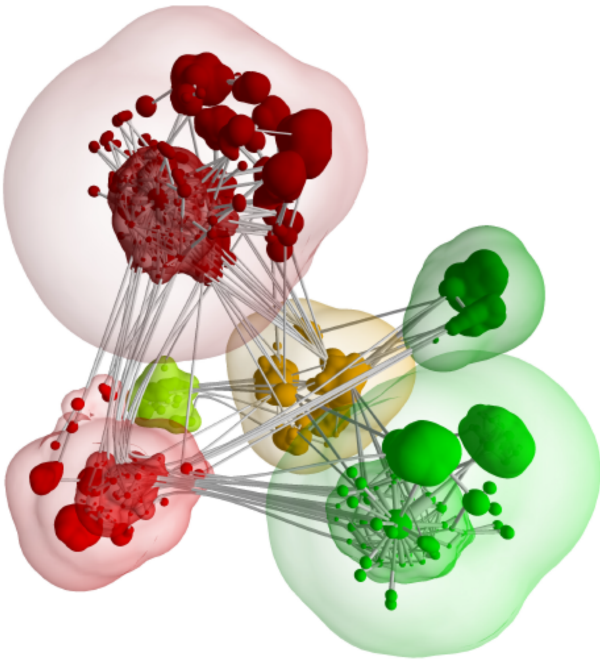

Figure 2.3: Level-of-detail visualization represents the clustered graph [13].

The geometry-based edge bundling [14] is another nice project. It calculates control points on a mesh and builds a graph for routing paths. The edges are clustered along the paths. Moreover, it also provides animation to show the process of the edge bundling algorithm (see Figure 2.4). Furthermore, there is also a force-directed edge bundling algorithm developed by Holten and Wijk [15].

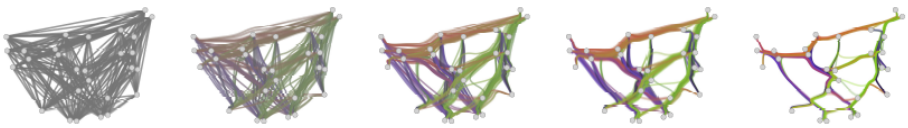

Figure 2.4: Morphing from a raw graph to the one with bundled edges [14].

The above solutions for edge bundling are without area restrictions and treat nodes as points. How can edges be bundled to avoid going through nodes that have a width and height? A geometry-based method is developed to solve the problem by Pupyrev et al [16] (see Figure 2.5).

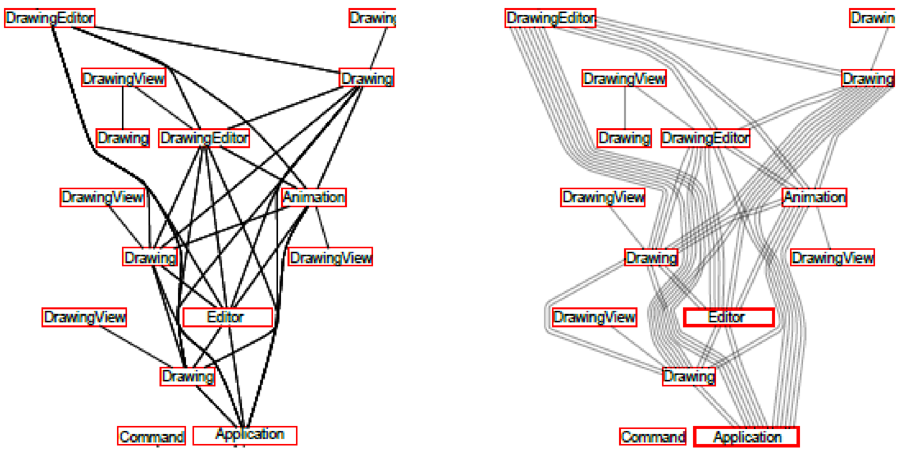

Figure 2.5: A graph with rectangle nodes and routed (left) and bundled edges (right) [16].

2.1.3 Others

There are also many projects on the visualization of text data. Bubble sets are in an aesthetic way to represent clusters [17]. Each cluster contains a

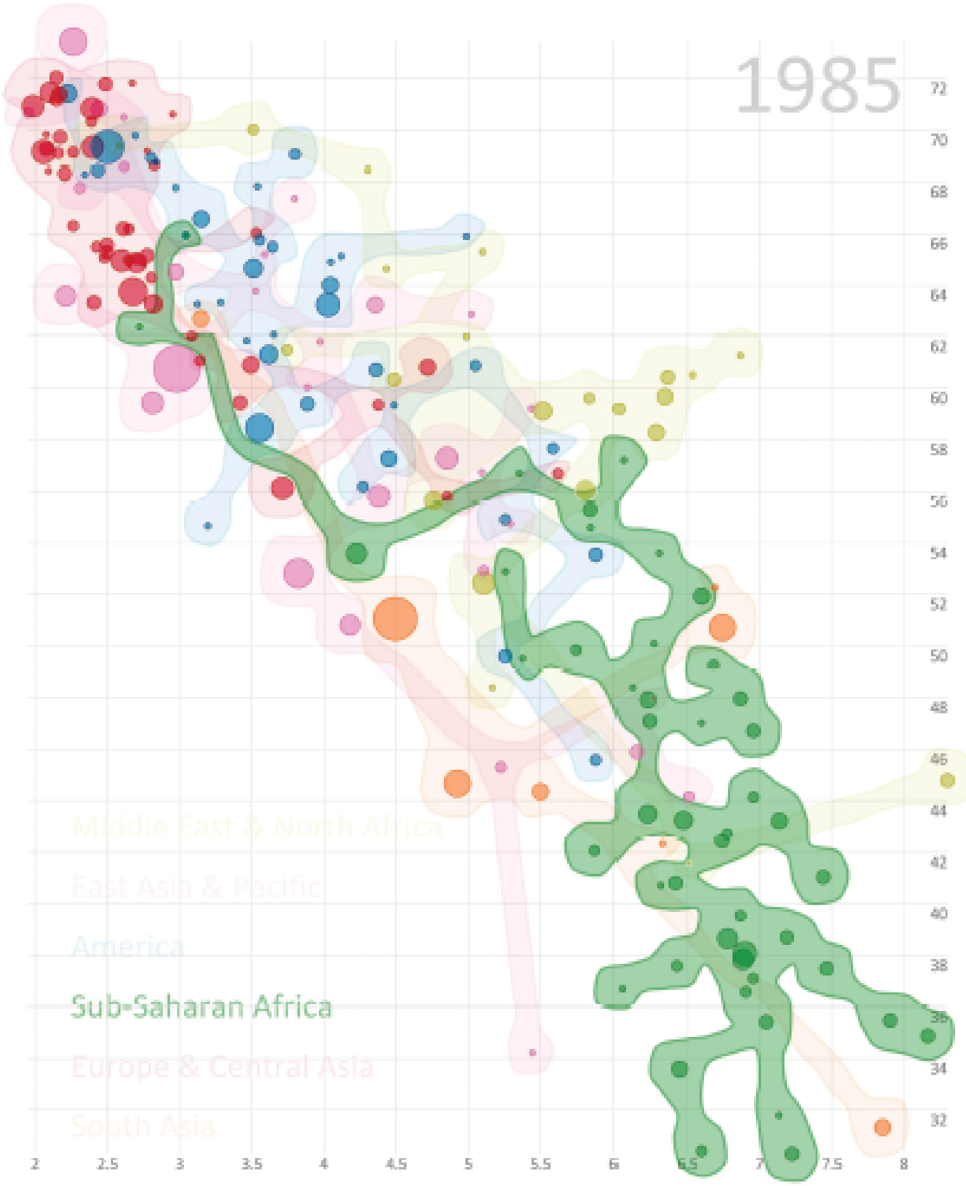

Figure 2.6: Bubble sets to show the topical trends [17].

few articles and they are displayed in an isocontour area like something is restricted in a bubble (see Figure 2.6). The visualization also implies that edge bundles can be confined in restrict areas.

Docubrust is a visualization tool with natural language processing to show the summary about a document [18]. It extracts the document into tags by word counts. Then it looks up the tags in an electronic dictionary and builds a semantic tree. The summary of the document will be shown in a radial layout. Users can analyze the semantic tree to know more about what the document is talking about (see Figure 2.7).

Word Bridge focuses on showing a few clusters from large documents [19]. It summarizes documents and classifies them into groups by topics. Then the groups are represented via tag clouds. If two groups are related, they are connected by a blue straight line (see Figure 2.8).

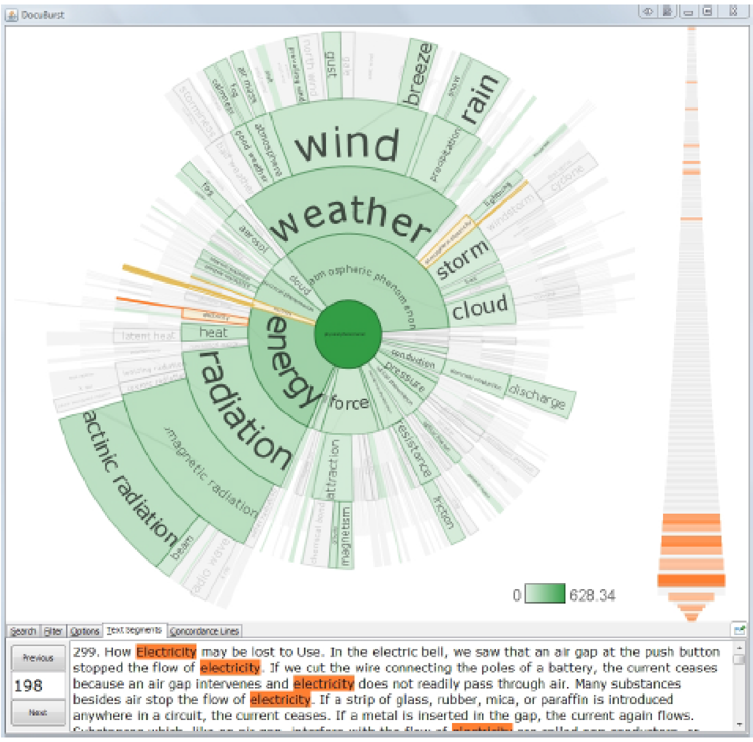

Figure 2.7: A document is visualized by Docubrust [18].

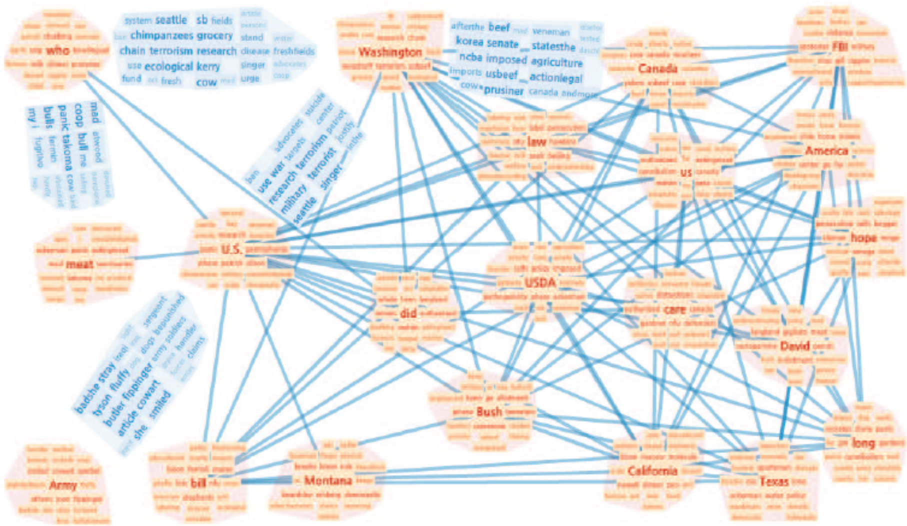

Figure 2.8: Word Bridge visualizes a set of documents [19].

JauntyNets is also a nice tool to visualize information of a collection of documents [20] [21]. It lists and shows all attributes extracted from documents as tags around a circle and puts documents as nodes inside the circle. The visualization combines multiple coordinated views with attribute-driven layout (see Figure 2.9). It can reveal the information from nodes, their attributes and relationships between nodes in a network.

For human-computer interaction, Bier et al [22] provided a solution of magic lens. The lens can show the detailed information about selected points in a dataset, while keeping the overview of the visualization of the dataset. It is nice way to reveal hidden information from contents with a huge amount of visual objects.

2.2 Visualization Pipeline

The visualization pipeline model is a primary framework (see Figure 2.10), which is described by Stuart [23] and modified by Chi [24]. This model is widely used and divides the visualization process into 2 segments. The first part is used to pre-process data. The raw data will be translated into struc-

Figure 2.9: The visualization is generated by JauntyNets [20] [21].

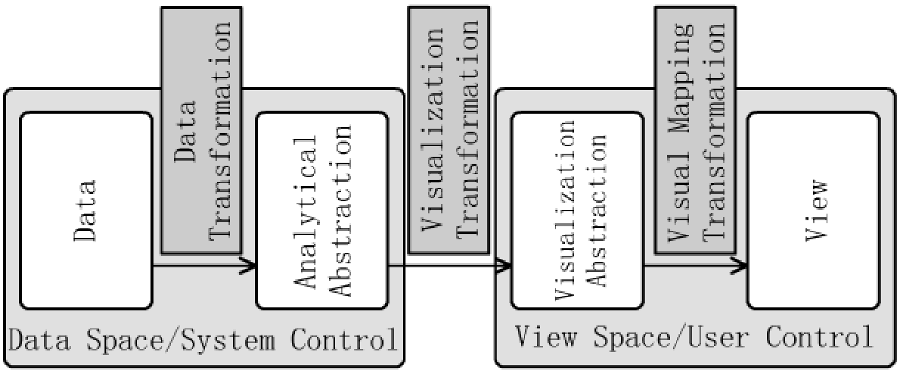

Figure 2.10: The visualization pipeline model.

tured data. The latter one is for rendering selected visual objects. It filters out some visual objects and presents the remainning ones on the screen. The connection between those parts is significant, which maps structured data onto visual shapes in general.

2.3 Tagging and Clustering

Following the visualization pipeline, the first step is to pre-process raw data. Many methods and algorithms can handle documents. There are two common methods of tagging and clustering.

Tagging (see Figure 2.11) is a process to tokenize text data, remove form words, do statistics for the count of each word, sort the counts and get some top ones as keywords [7]. It is useful to summarize documents automatically. With the help of algorithms of natural language processing, tagging will perform better to calculate and get the abstracts of text data [25].

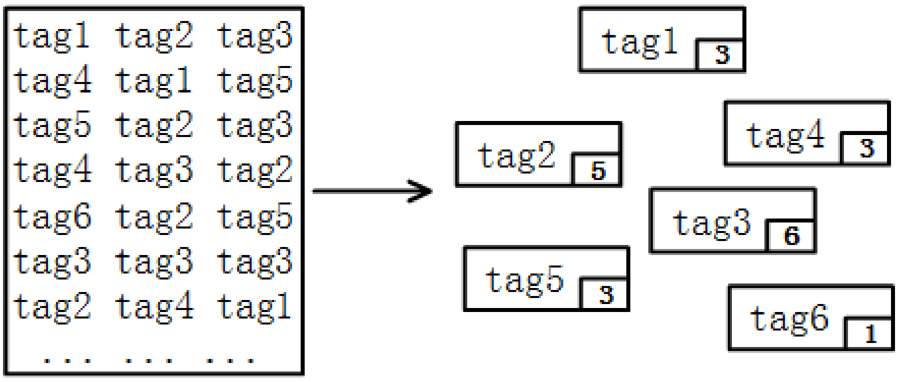

Figure 2.11: Tagging: to extract a document into tags.

Clustering (see Figure 2.12) divides documents into groups [7]. Considering that a document has many attributes of which each is with a value, the values can be described by a numeric vector. The distance between vectors is to aggregate the differences between the elements of two vectors. A group consists of documents whose vectors are close to each other's by distance. Thus before clustering, the vectors for the document should be defined. Then

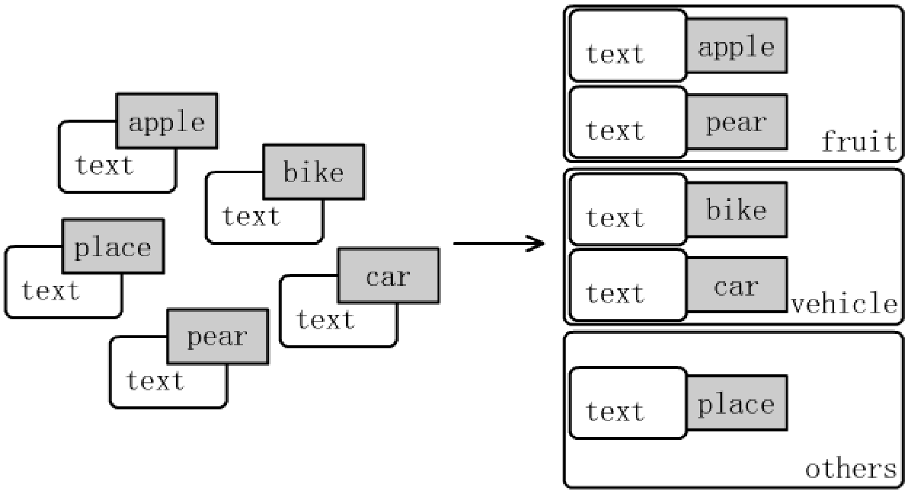

Figure 2.12: Clustering: to put documents into groups.

give an initial center for each group and repeat calculating the distance between the centers and vectors; putting the document, whose vector is closest to a center of a group, into the group and updating the center. While a simple network $N = (V, E, A, R)$ comprises 4 sets in order. The set V and E describe the underlying graph in the network. The additional attributes constitute the set $A = \bigcup_{x \in V \cup E} attributes(x)$, i.e. a closure set of attributes which nodes and edges have. Moreover, the set R explains which element in the graph has what attributes and their values, $R = \{(x, a, r) | x \in V \cup E, a \in attributes(x)\}$, with the r a real number.

2.4 Network

A network is a kind of data structure, which is a graph with additional attributes that are attached to its graph elements [26]. Mathematically, a simple graph $G = (V, E)$ is made of a set V of vertices together with a set E of edges that $E \subset V \times V$ [1].

We now discover how such a network is applied in the model of visualization pipeline. In the former part of the pipeline, raw data are scattered at first. After tagged and clustered, the raw data are translated into a network

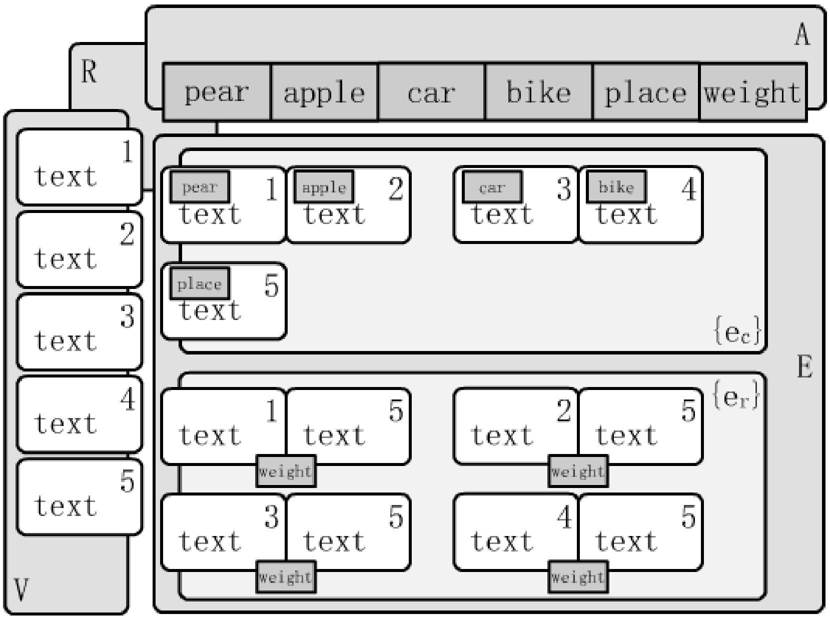

Figure 2.13: Network: the structure to store clustered text data with their relationships.

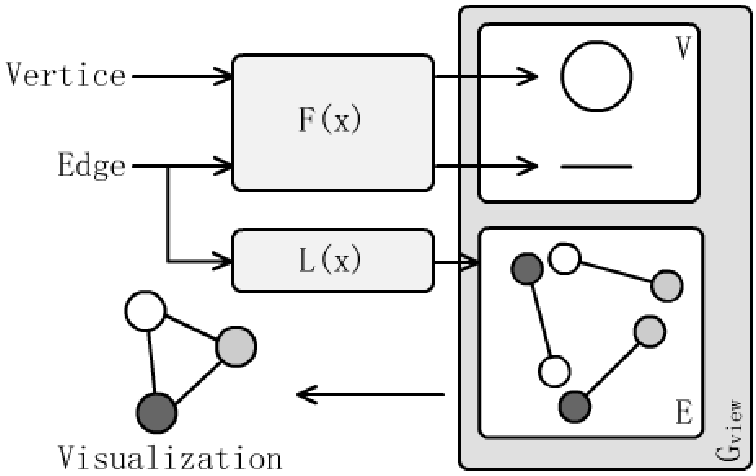

Figure 2.14: Visualization Graph: the structure for visualization rendering.

$N = (V, E, A, R)$ whose V is a set of documents. The set $E = \{all\ e_c\ and\ e_r\}$ contains edges in 2 types of cluster edge $e_c \subset V$ and relationship edge $e_r = (v_i \in V, v_j \in V | i \neq j)$. For example, Figure 2.13 shows a network with five documents which are consisted in the set V. They are tagged into *pear*, *apple*, *car*, *bike*, *place*. And the attribute *weight* is for relationship edge. Thus the 6 attributes belong to the set A. There are 3 cluster edges {text1, text2}, {text3, text4}, {text5} and 4 relationship edges (text1, text5), (text2, text5), (text3, text5), (text4, text5). Those edges are made of the set E. The set R is behind them to define the weights of attributes of documents and edges.

After visualization transformation, the pipeline produces visual objects. The elements of V_N and E_N in the network will be mapped into visual shapes through a function F. A function L is for mapping their layouts. At last, the pipeline renders a view on the screen. The view is also a graph $G_{view} = (V, E)$ consisting of a set $V = \{F(x) | x \in V_N \cup V_E\}$ of visual shapes and a set $E = \{L(x) | x \in V_E\}$ of layout information (see Figure 2.14).

2.5 Visualization Rendering

Rendering builds the network into a graph of view G_{view} and paints G_{view} on the screen. A renderer is responsible for it. The renderer orders visual shapes by their priorities and arranges them at correct positions with the function L. Then it fills the shapes with colored pixels by the function F. There are three techniques below to be discussed.

2.5.1 Tag Clouds

Tag clouds put tags on a flat plane. It is clear to present the overview for the users. Important tags, whose count is relatively larger, are shown in different visual properties such as fonts, sizes and colors [27] (see Figure 2.15). The tag count was discussed in Section 2.3.

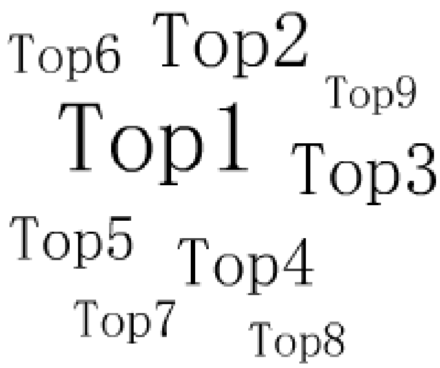

Figure 2.15: Tag Clouds: an example with tags in different font sizes.

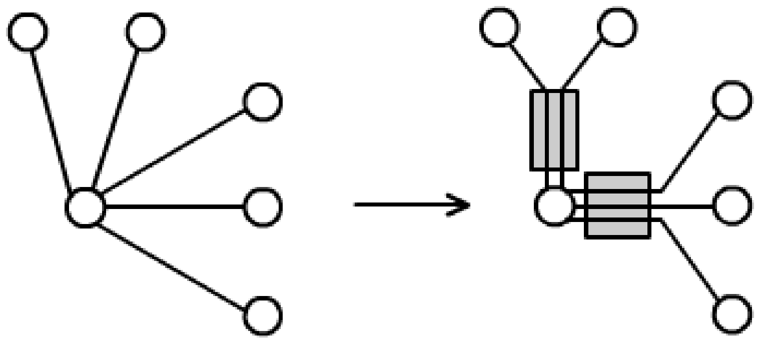

Figure 2.16: Edge bundles: to show the overview of a plenty of edges.

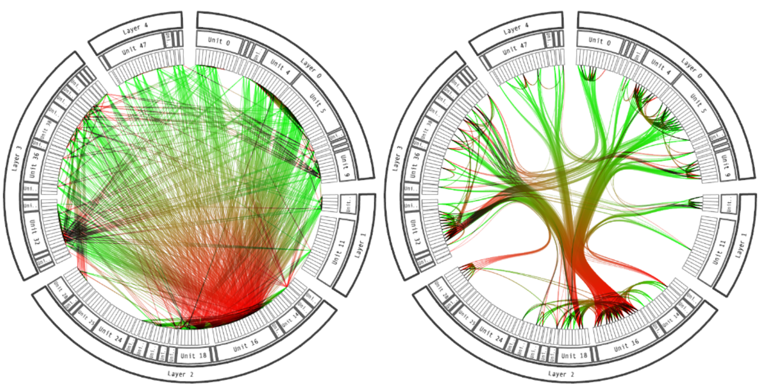

Figure 2.17: Edge bundles: reduce edge clutter from (left) straight to (right) bundled edges [11]

.

2.5.2 *Edge Bundling*

Edges rendered as straight lines bring overlapping and occupy much space in a view when the amount of them is large. Edge bundling partitions edges into groups and sorts them by their weights. The bundles of edges are drawn in different widths to represent the amounts of edges (see Figure 2.16). The edge clutter can decline and the bundles make visualization comfortable to present the overview of lots of edges [28]. Figure 2.17 shows an example of how edge bundling can reduce clutter.

2.5.3 *Double Buffering*

The processes of rendering and displaying are often assigned to two threads respectively. When those threads are not synchronous, more exactly, the displaying thread is faster than the rendering thread, the viewport acts as it flickers, i.e. it shows the view with or without contents unsteadily. This problem has a simple solution, which is called double-buffering. One buffer is filled by the rendering thread while the other one is displayed by the displaying thread. Once the rendering is finished, the rendering thread swaps the buffers. The other one gets filled and the one that was filled now is displayed. Then it repeats infinitely in program running (see Figure 2.18).

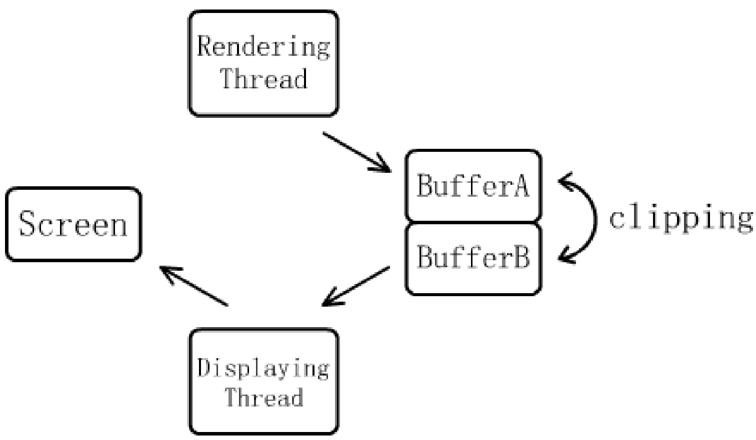

Figure 2.18: Double-buffering.

2.6 Human-Computer Interaction for Visualization

Human-Computer interaction deals with the communication between human beings and computers using input devices like mouse, keyboard and output devices such as monitor or printer. The computer takes tasks after users send them some commands. As for visualization, people would like to discover the visualized data sets. Therefore, the interactions are necessary. There are several interaction techniques: reconfiguring, encoding, filtering, abstracting/elaborating, connecting, exploring, selecting and others [29] [30]. We will discuss four of them here.

2.6.1 Encoding

Encoding is used to represent a same object in different ways [29]. For instance, the number 3 can be represented by 3 points, 3 lines, 3 squares, or 3 cubes. Color coding is another form of encoding. It can highlight a specific object with a unique color and with others colored completely differently. It can also distinguish a group of objects with different weights by painting gradient colors.

2.6.2 Filtering

Filtering removes visual objects from a view by matching some special attributes of the objects [29]. The most used example is book searching. To check if some books are available, a searching system does not list all books and make people look for what they want themselves. It requires the users to type some keywords and list a few books which are highly related to the keywords. Filtering accelerates finding something people are interested in.

2.6.3 Selecting

Selecting enable users to mark an visual object of interest to keep track of it. [29]. For example, users can make an interesting rectangle selected by highlighting it. On a screen there are lots of rectangles. The users use a mouse to click on a rectangle to select it. The system will repaint it differently, like coloring it specially, to make it stand out. Multiple selecting is used to mark some of visual objects as interesting elements.

2.6.4 Connecting

Related items in a view can be highlighted or de-emphasized by connecting [29]. Brushing and linking are the classical techniques of connecting.

- **Brushing**. Brushing means highlighting a subset of the data items or removing it in a view when the user would like to focus on others [29].
- **Linking**. Linking is a kind of technique to help the user trace a visual object in different arrangements synchrously [29] [30]. For instance, a box is displayed in four different windows with top, left, front, perspective view respectively. If a corner of the box is brushed in a view, the places referring to the same corner in the other views will be also brushed.

3 Prototype Development

The goal of this thesis is to build an application prototype to visualize the network with tags and relationships of documents. It can help users to get an overview of a collection of documents and to find interesting relationship patterns. This chapter will describe the process of design and implementation. The application prototype is written in Java, which makes it run on most platforms. To keep the project reusable, the source code was refactored for several times and the classes were classified carefully into different packages. Some mathematical models, like the shape mapping function F and the layout mapping function L, are discussed in Section 2.4. The visualization is realized by creating instances for those models with following the visualization pipeline.

3.1 Design

Before programming work, we should know what shapes refer to which elements in a network of documents. The first sections address the design of mapping from network elements into visual shapes. The middle sections present the idea of the global layouts for visual shapes. The last section provides the design of interactions about how the users interact with and explore the network of documents.

3.1.1 Visual Object: Document

The visualization tool should present a document network which contains documents, document clusters, and relationships between documents. Document can be represented by many shapes. There are several ideas to have rectangles, triangles, pies and small circles to represent documents (see Figure 3.1). We prefer to use pies, for they are realized more easily. Java internal libraries provide the method to draw or fill a pie efficiently. And a group of pies could be put into a circle and drawn in the same size. Although there is also a method to fill a rectangle and one to fill a circle, they are required more calculation to define the position of each element. Thus it is defined as,

$$F(document) = pie \tag{3.1}$$

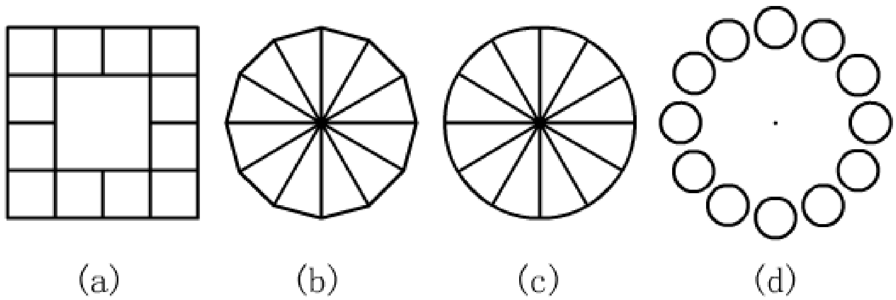

Figure 3.1: Documents in (a) rectangles, (b) triangles, (c) pies and (d) small circles.

3.1.2 Visual Object: Tag

Documents are summarized by keywords, which are printed as tags. A Tag cloud is used to indicate a few tags at the same time. To highlight important tags, we apply different font sizes to the tags and follow the rule that bigger font sizes represent more important tags. Moreover, limiting the maximal font size allows us to have more clouds in one view. A tag is represented as a box with a word or phrase printed. Tags are restricted in a specific area, especially inside a specific shape. It is better to put the tags inside a circle since the layout of tags can be decided by a fewer equations. A spiral equation with a constant c and a parameter t:

$$\begin{cases} x &= c * t * \cos(t + \frac{\pi}{2}) \\ y &= c * t * \sin(t + \frac{\pi}{2}) \end{cases} \tag{3.2}$$

is used to determine the positions of tags (see Figure 3.2). Additionally, there should be no collision between tags. Thus,

$$F(tag) = rectangle \tag{3.3}$$

and the layout of a tag cloud in a cluster,

$$L_{TagCloud}(cluster) = \{(tag, T) | T(x, y), F(tag_i) \cap F(tag_j) = \phi \text{ and } i \neq j\} \tag{3.4}$$

Here we assume that T(x,y) is a translation transformation where the x and y follows the Equation 3.2 and ϕ is an empty set.

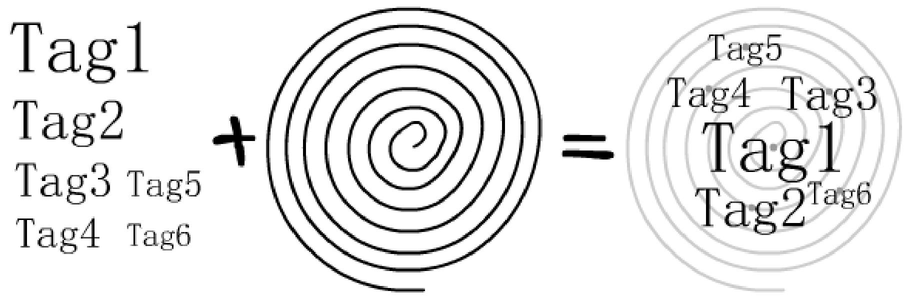

Figure 3.2: Tags in a tag cloud.

3.1.3 Visual Object: Cluster

A cluster is a big set that contains some documents and tags. We use a circle to represent a cluster. The visual pies of documents are put together to form a circle. Every document has a few tags. A cluster of documents collects the tags, each of which appears in different counts. The tags are sorted by count and the top 10 tags remain to make a tag cloud. The tag cloud is displayed in the center of the cluster circle (see Figure 3.3).

Considering the documents can have different attribute values, we have two ways to deal with this case. One is using gradient colors for pies to distinguish documents with different values (see Figure 3.4 (a)). The other one

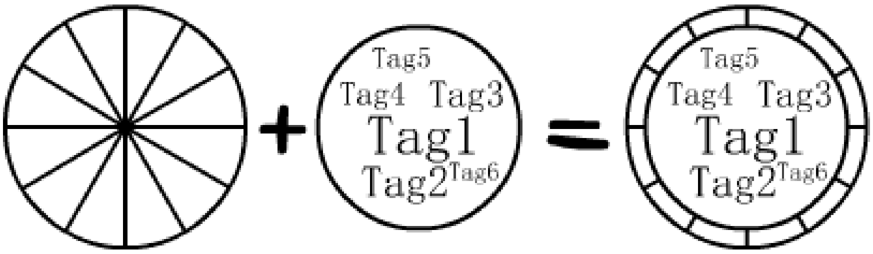

Figure 3.3: Document pies and the tag cloud are combined into a clusters.

is applying different angles to pies (see Figure 3.4 (b)). The former method performs well when there are many documents, but sometimes users could not distinguish too close colors. The latter one indicates the documents in different attribute values remarkably. However, when a document has an extremely large value, the others will be drawn in very small pies as if they were straight lines. Users cannot see them clearly. Thus, we prefer the first method.

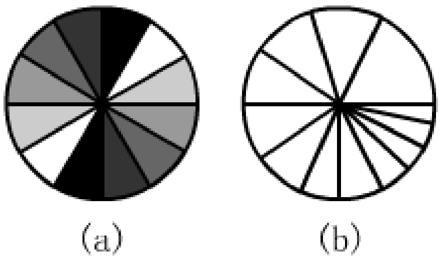

Figure 3.4: Documents pies in (a) gradient colors and (b) different angles.

A cluster with N documents and rotation transform R in the graph of view is defined as

$$\begin{aligned} F(cluster) &= circle \\ L_{TextRing}(cluster) &= \{(text_i, R(a)) | a = \frac{2\pi i}{N}, i = 0, 1, 2, ..., N\} \\ L(cluster) &= L_{TextRing} \cup L_{TagCloud} \end{aligned} \tag{3.5}$$

3.1.4 *Visual Object: Relationship*

The documents are connected by relationships. There are two kinds of relationships which where discussed in Section 2.4. For the documents are clustered, *external* and *internal* relationships should figure in the discussion. *External* relationships are the ones that connect a document in a cluster and another one outside the cluster. The connections between documents in the same cluster form *internal* relationships. The simplest way to draw relationships is using straight lines. However, the view will be messy when the number of relationships is huge. Straight lines will also go across the clusters. Thus some techniques should be applied to reduce the clutter.

We compare two possible approaches of polylines and Bezier curves [31] in general (see Figure 3.5). A polyline is a path of straight lines along all control points. It is easy to draw and the painting just takes a slice of time. It is also easy to check if it goes across some area. It is very quick to judge whether a point is on it. It, however, just looks not so well.

Bezier curves use tangents to define a path. It is smooth and beautiful but it consumes much more time to check that a point hits it than for a polyline.

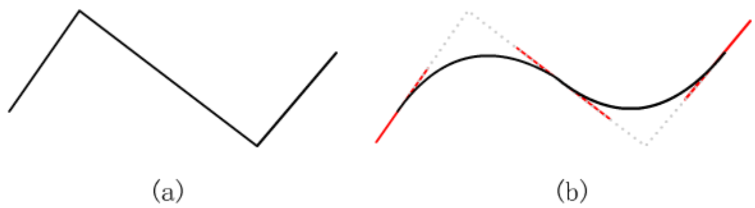

Figure 3.5: External relationships in (a) polylines and (b) Bezier curves.

We would like to use quadratic Bezier curves for *external* relationships in the circular cluster layout. The layout will be discussed in next section. Three control points A, B, C describe a curve. The curve starts from the center middle between A and B with the tangent line AB; ends in the point at the center middle between B and C with the tangent line BC. The representation of *ex-*

ternal relationship defines with a set of control points $\{P_i | i = 0, 1, 2, ..., N\}$ as

$$
\begin{aligned}
&F(external) = \{B_i(t)| \\
&\quad B_i(t) = (1-t)^2 \frac{P_{i-1} + P_i}{2} + 2(1-t)tP_i + t^2 \frac{P_{i+1} + P_i}{2}, i = 1, 2, ..., N-1\}
\end{aligned} \tag{3.6}
$$

For *external* relationships in the free cluster layout and *internal* relationships in both layouts, we would like to use another form. The design of *external* relationships in the free cluster layout will be discussed in Section 3.1.6. We mix straight lines and arcs to build a curve (see Figure 3.6). The representation of *internal* relationship defines with four control points P_0, P_1, P_2, P_3 as

$$F(internal) = \overline{P_0 P_1} + \overset{\frown}{P_1 P_2} + \overline{P_2 P_3} \tag{3.7}$$

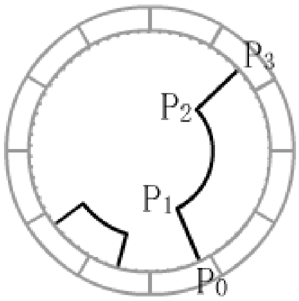

Figure 3.6: Two internal relationships in solid black curves.

3.1.5 Circular Cluster Layout

In the previous sections, we discussed the design for visualizing the elements of a document network. These two sections examine two layouts to arrange the visual objects respectively. In general, every layout has three layers. We put all *external* edges displayed on the bottom layer. The middle layer is for drawing the clusters with *internal* edges. The top one is reserved to show extra information for interactions.

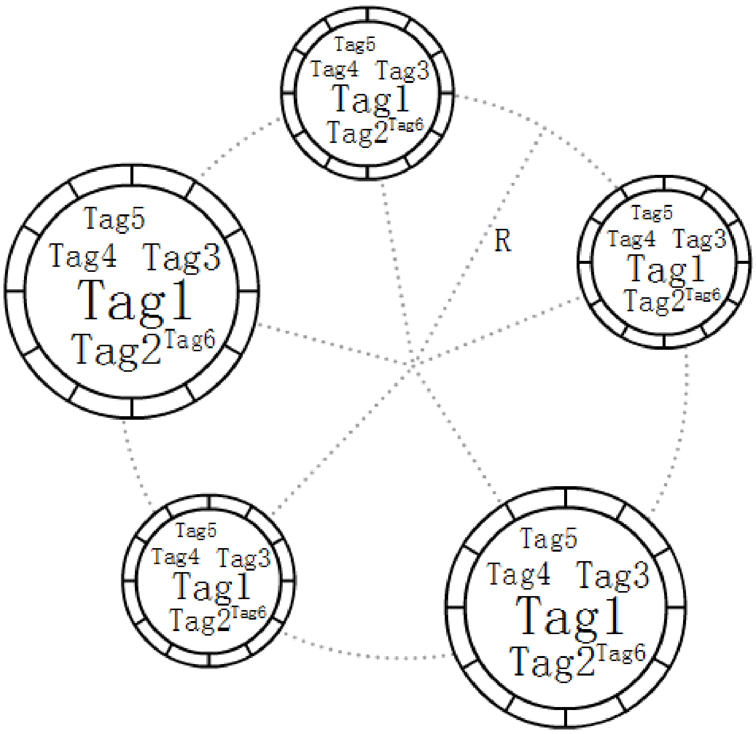

Figure 3.7: Several clusters in the circular cluster layout.

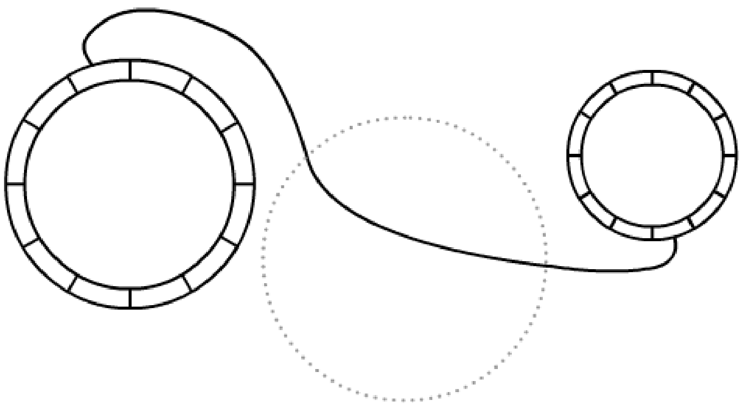

Figure 3.8: An edge for external relationship in the circular cluster layout.

In circular cluster layout, the clusters are fixed around a circle with a radius R (see Figure 3.7). We treat the center of the circle as an absolute center. An edge for *external* relationship starts from or ends at a document pie. The edge should not overlap on any clusters while it should go across an invisible circle in the center of view (see Figure 3.8). Edges for external relationships are bundled in the invisible circle. The entire layout with N clusters is described as

$$\begin{aligned} L(Circular) =& \{(cluster_i, T) | T(R\cos a, R\sin a), a = \frac{2\pi i}{N}, i = 0, 1, 2..., N\} \\ & \cup \{(external, T) | T(0, 0)\} \end{aligned} \tag{3.8}$$

3.1.6 Free Cluster Layout

Initially the clusters are around a circle and the user can drag them to other positions in free cluster layout. When doing edge bundling in space, people cannot trace an edge exactly but can get a first overview [11]. The whole free cluster layout is

$$\begin{aligned} L(Free) =& \{(cluster_i, T) | T(x, y) \text{ determined by dragging}\}, i = 0, 1, 2..., N\} \\ & \cup \{(external, T) | T(0, 0)\} \end{aligned} \tag{3.9}$$

Free cluster layout is similar to circular cluster layout. They are different from the representations of *external* edges. To give an equivalent overview of *external* edges, we employ a new approach of **edge halo** by our own. For one edge of *external* relationship, there are six control points from P_0 to P_5 to define it as (see Figure 3.9)

$$F(external) = \overline{P_0P_1} + \overset{\frown}{P_1P_2} + \overline{P_2P_3} + \overset{\frown}{P_3P_4} + \overline{P_4P_5} \tag{3.10}$$

Each cluster is surrounded by *external* edges bundled as a halo ring (see

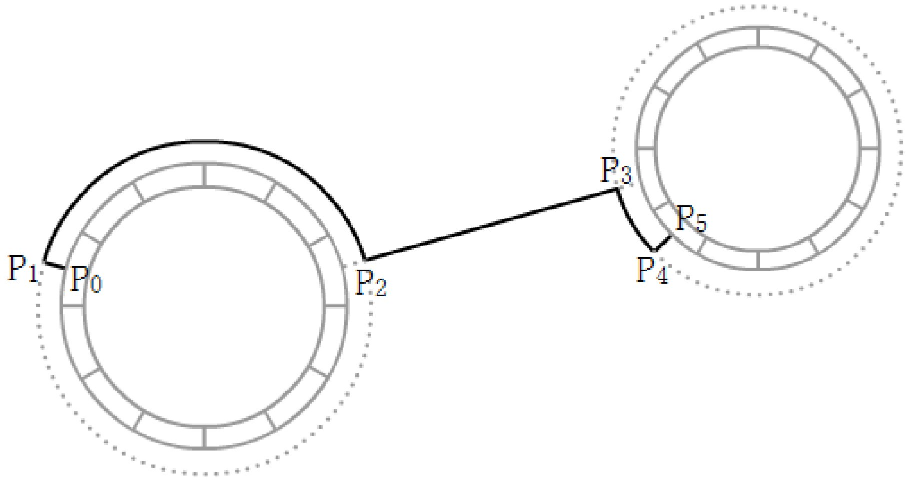

Figure 3.9: An edge for external relationship in the free cluster layout.

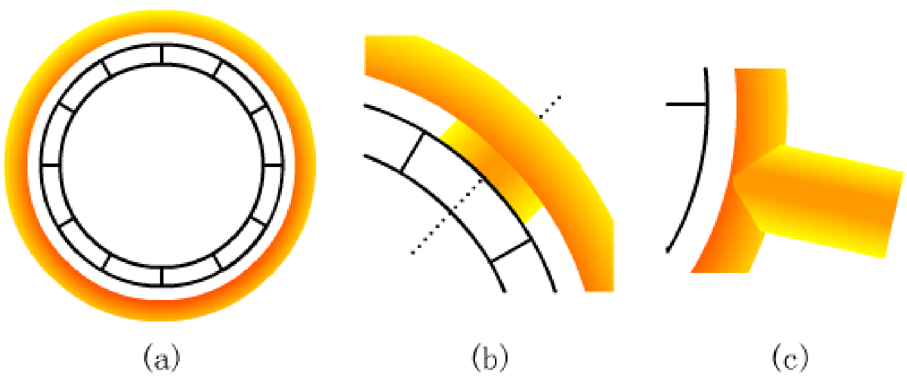

Figure 3.10: The design detail of edge halo with (a) halo ring, (b) node-halo connection and (c) halo-halo connection.

Figure 3.10 (a) and the orange area is for putting the edges with higher weight values). There are two arcs for each edge. The length of P_0P_1 and P_4P_5 is determined by the weight of the edge. The higher the weight an edge has, the smaller the length of the line segment is. Between a document pie and the halo ring, there is a strip consisting of $\overline{P_0P_1}$s or $\overline{P_4P_5}$s of the edges. They are arranged by the rule that the higher the weight value is, the nearer the edge is from the axis line of the document pie (see Figure 3.10 (b)). The strip as node-halo connection bridges a document pie and the halo ring in a cluster. $\overline{P_2P_3}$s of the edges go straight from one halo ring to another (see Figure 3.10 (c)). The halo-halo connection with edges looks like a light beam. The halo rings show which cluster has most relationships and even high-weighted ones. The edge beams imply which two clusters are most related.

3.1.7 Interactions

The application prototype has a simple and clean graphical user interface (see Figure 3.11). There is a button for opening a file and two buttons for switching the view mode between the circular and free cluster layout. There is also a selector to decide how many clusters the user would like to have and a filter to show edges conditionally by their weights. Moreover, a button on the left-bottom is to open the dataset explorer which will be discussed in Section 3.2.3. When loading a document network, the users can explore document information with their titles and authors in a list in the explorer. Those are all on the left side on a control panel. Besides, the view panel is on the right side. It presents the visualization to the user.

To help the user explore the relationships of clustered text data and the clusters, the prototype supports a few interactions. A gradient color map is applied to make interesting elements remarkable (see Figure 3.12). The user can choose an attribute of document or relationship to map the values onto colors. The larger the value is, the darker the color is. We use the colors from

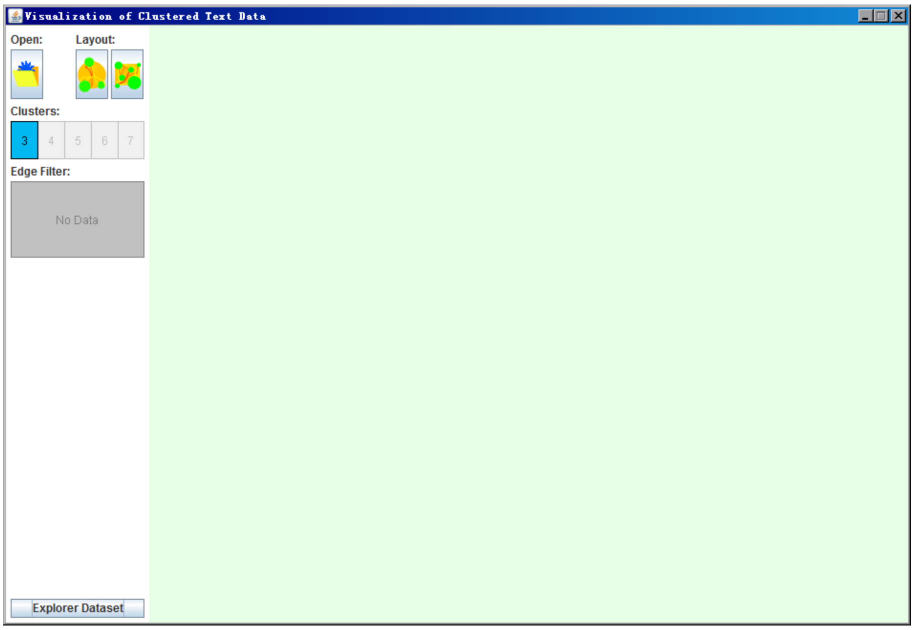

Figure 3.11: The graphical user interface of the application prototype.

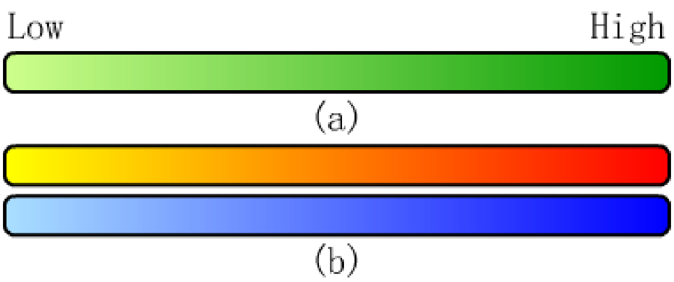

Figure 3.12: The gradient color map for (a) document pie and (b) edges in normal(up) and selected(down)

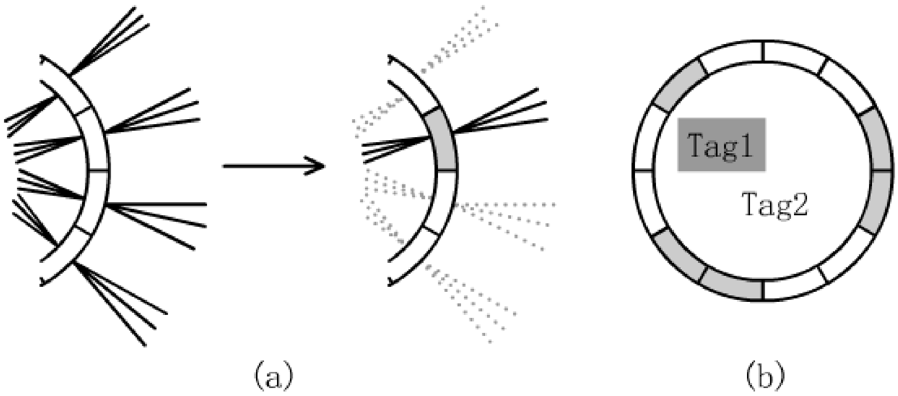

Figure 3.13: Single selecting for (a) document and (b) tag.

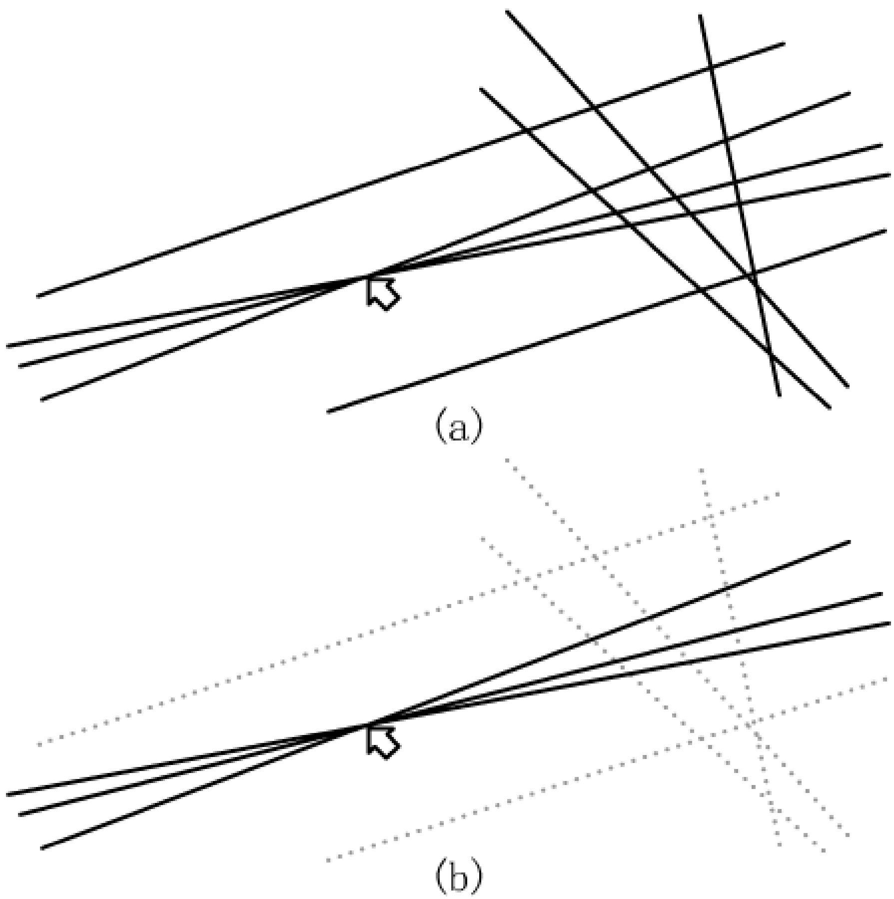

Figure 3.14: Multiple selecting for edges (a) before selecting and (b) selected.

light green to dark green to paint document pies. And the colors from yellow to red are for relationship edges. Single selecting is provided to mark a document or a tag. If a document is marked, the edges which are not connected to it will be filtered out (see Figure 3.13 (a)). If a tag is selected, the related documents in the cluster will be highlighted (see Figure 3.13 (b)). Multiple selecting is also available. If the user moves the mouse cursor over external edges, they will be colored by the gradient from light blue to dark blue. By clicking on that point, the user can select and highlight those edges and all other edges are filtered out (see Figure 3.14). By double-clicking a cluster, the user can remove all external edges connected with the cluster from the view. Moreover, in the free cluster layout the user is able to drag a cluster to other positions and the view will update automatically. To avoid the draggable clusters overlap each other, we decide to implement a force-directed algorithm to re-arrange the clusters when some of them cover the others.

3.2 Implementation

Three implementations where realized to keep the project clean and understandable. In the first implementation, the visualization of clustered text data without relationships between documents is achieved. The second implementation includes two experiments on the layouts and the solution to deal with edge bundling. The prototype is built with the designed graphical user interface in the final implementation.

3.2.1 First Implementation

In this iteration, we finished the tasks to show document pies, tags and clusters (see Figure 3.15). The single selecting for documents and tags are also complete. A special interaction is added to the first prototype. The user is able to roll the mouse wheel to change the size of a cluster. When a cluster gets smaller, some tags become invisible if they are not completely con-

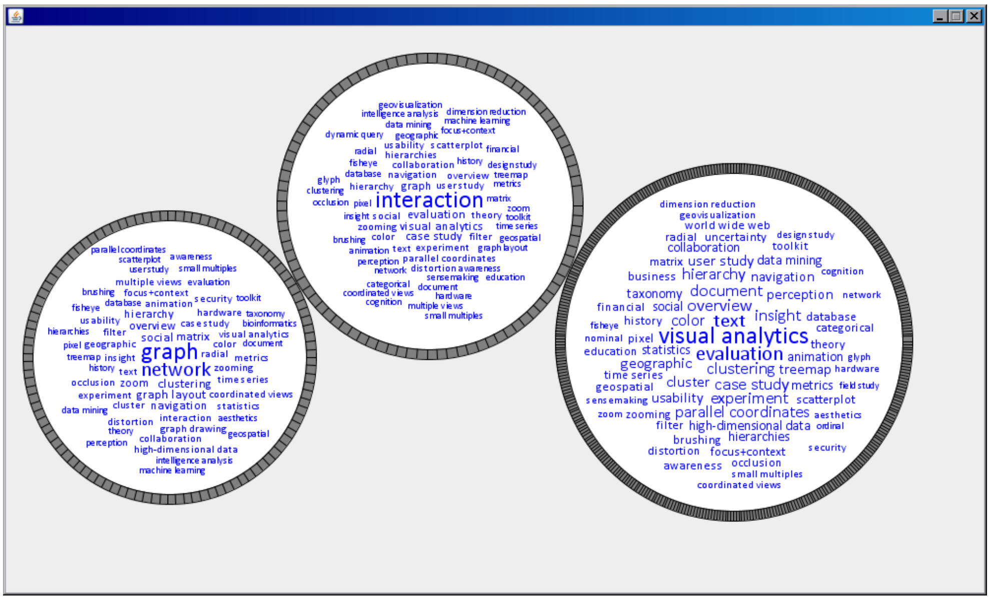

Figure 3.15: The first implementation for complete clusters without specific color coding.

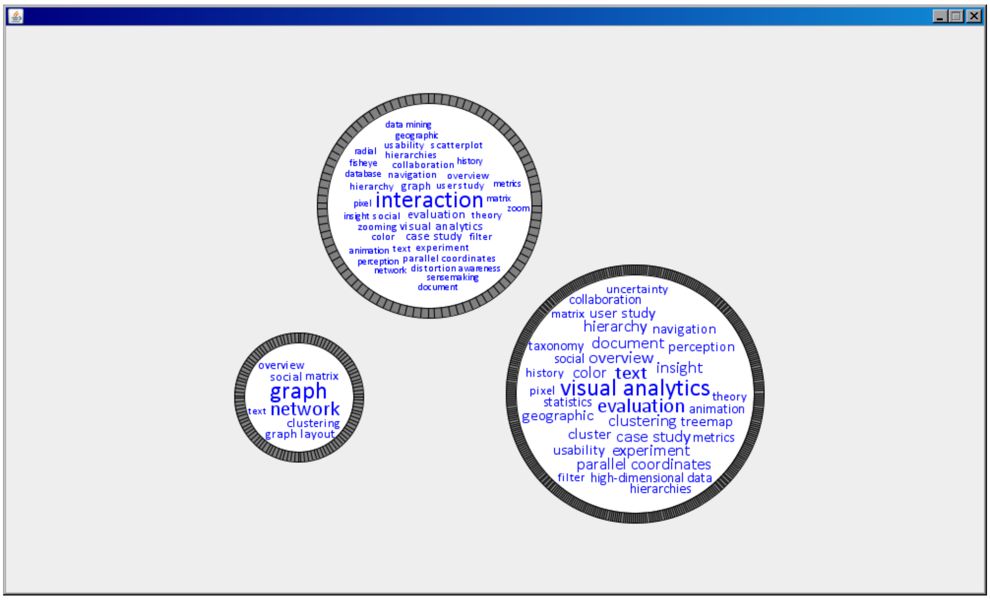

Figure 3.16: The clusters whose size are changed.

tained in the cluster circle (see Figure 3.16). However, in later development we found out that it will take much re-calculation because all edges for relationships should be rebuilt when the size of a cluster changes. Therefore, we removed that function after this iteration.

We manage to draw tag cloud by trial and error. At first, the application just ensured that there is no collision between tags and their centers were on the spiral via the Equation 3.2. However, they look scattered. Thus to make them tight, we wrote an algorithm which re-arranges tags towards the vertical axis of a cluster. It sorts the tags by y-axis value and then put them towards the center of the cluster by changing their x-axis value. This results in a nice layout and in limited space most of tags can be visualized.

3.2.2 Second Implementation

The iteration yields a full view of the design without the expected graphical user interface (see Figure 3.17). Our focus in this iteration is to discover how to display edges in the document network.

We did two experiments on the circular and free cluster layout respectively. They both are on routing a path from a point on a document pie to another. The only requirement is to make sure the path never goes across any clusters.

The first experiment deals with the circular cluster layout. First, it figures out the two tangent lines from the center of the view to a cluster. Second, the two half-lines from the center of the cluster to the two tangent points hold an angle. If a start point of an edge is contained in the angle range, it can go towards the center straightly; otherwise, it should add some control points to ensure the path starting from it does not overlap on any clusters (see Figure 3.18). We implemented a very simple algorithm for bundling the external edges which cross the invisible center circle. The external edges are sorted by their weights. If the weight is high, the control points of the edge with the weight will be close to the global center.

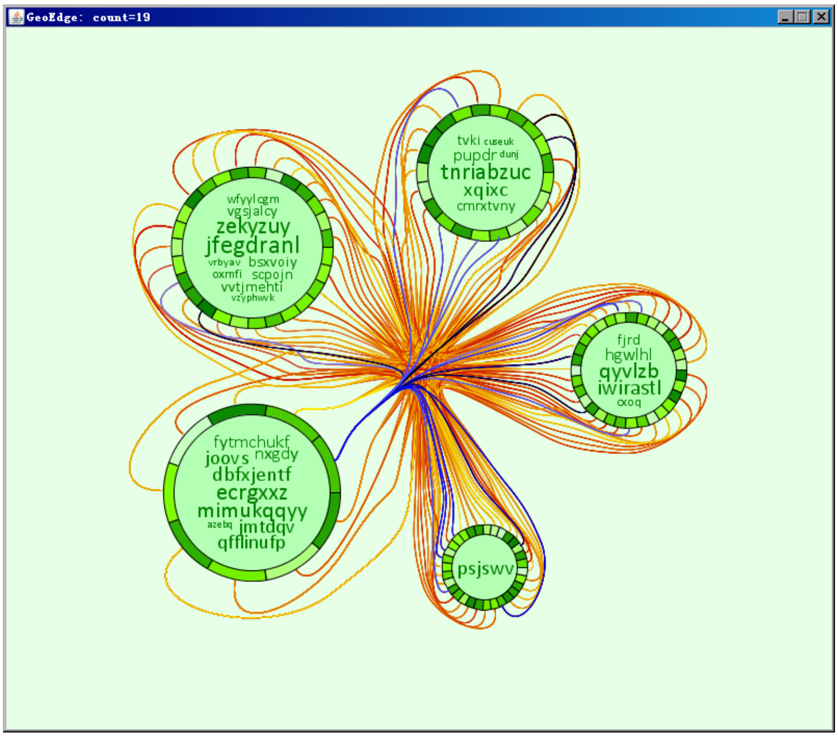

Figure 3.17: Second implementation with some edges highlighted.

It is a little more complex to deal with the routing problem in the free cluster layout (see Figure 3.19). In the first step, a grid will be built to cover the view completely. Then, the cells which do not contain in any clusters are stored in a list. Afterwards, we should know to which the two terminal points of an edge belong. Finally, in the list, the algorithm will find a path by moving a step cell by cell from a terminal to another one. If the algorithm fails to route the path, the two terminals will be connected by a straight line. After literature researching, we find out this problem is solved in 2011 [16]. Thus we consider it as a future feature and put more focus on other things like accelerating rendering.

3.2.3 Final Implementation

All things were complete in the last iteration. We continued the second implementation and improved it with the expected graphical user interface (see Figure 3.20 and Figure 3.21). We realized the edge halo in a simple way. And it shows the overview of relationship edges well (see Figure 3.21). During the procedure, we also meet some problems like the prototype cannot initialize the visualization in real time at first. Thus we create a thread to show a waiting view with simple animation. When the visualization is ready, the thread will be notified to stop and the visualization will be presented on the screen.

In addition, a data tip will be shown on the bottom of view panel when the cursor hovers on a document, an external edge or edges (see Figure 3.22). Before the document network is visualized, there is also a pop-up dialog to configure what attributes are about to be considered for the visualization (see Figure 3.23). The prototype will detect the type of attributes initially and the user can change the type of each attribute. The type *Tag* and *Node* are for document. If an attribute is a tag and it is used in high frequency, it will be displayed in tag clouds. A node attribute acts as the document weight. The colors of document pies depends on it. Similarly, an edge attribute is

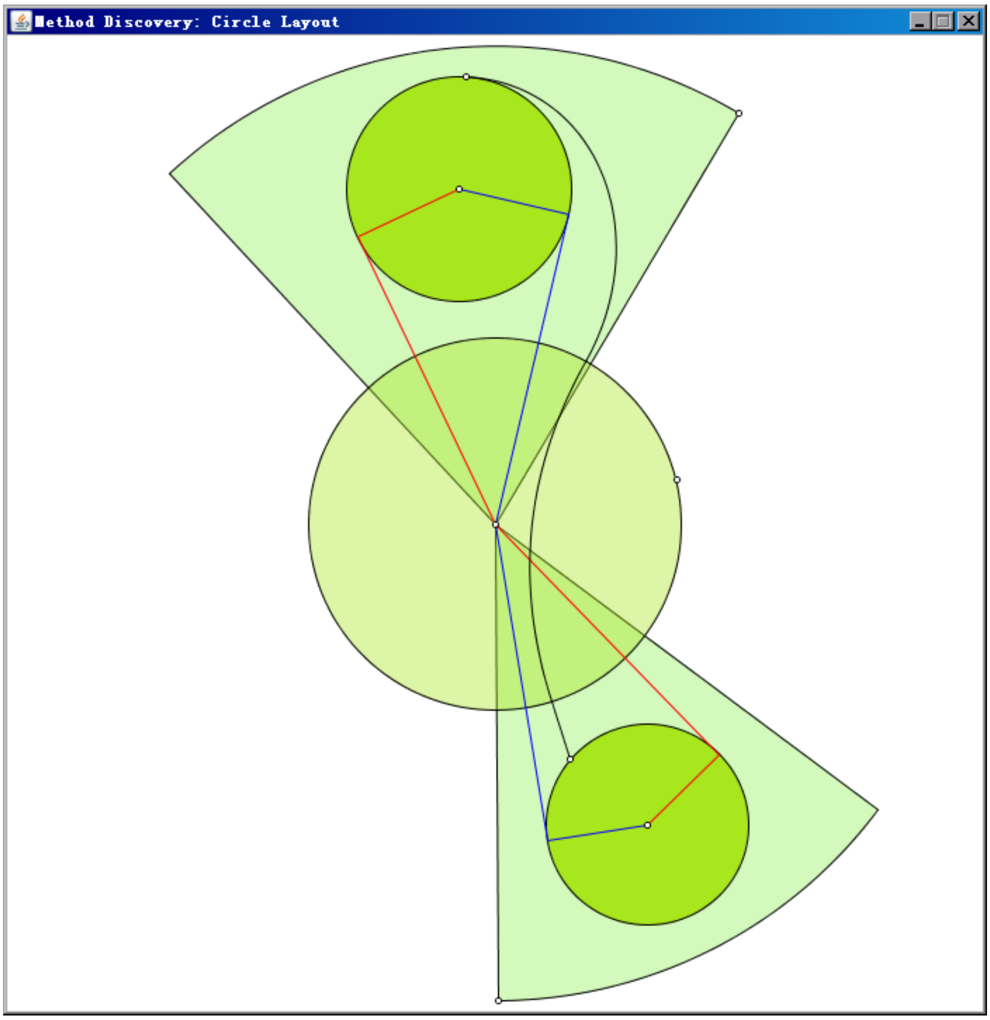

Figure 3.18: The experiment on routing for external edges in the circular layout.

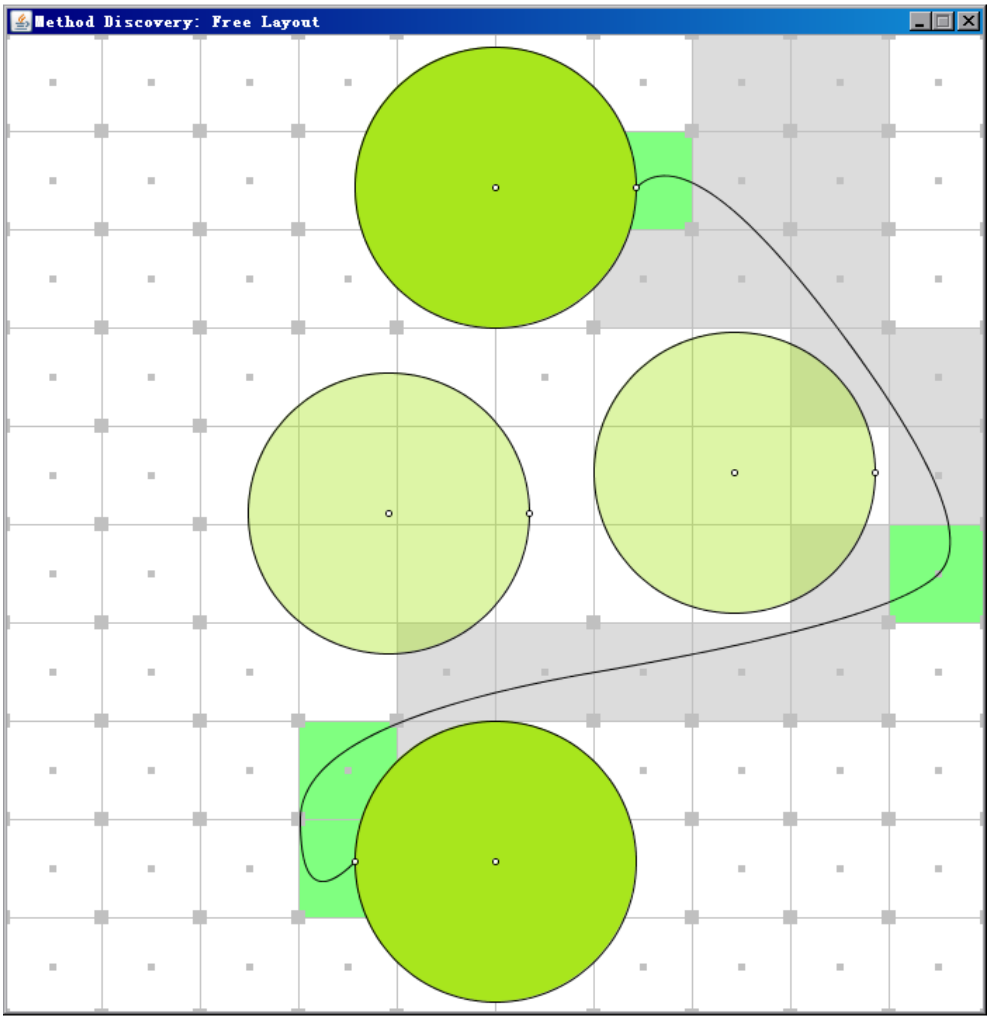

Figure 3.19: The experiment on routing for external edges in the free cluster layout.

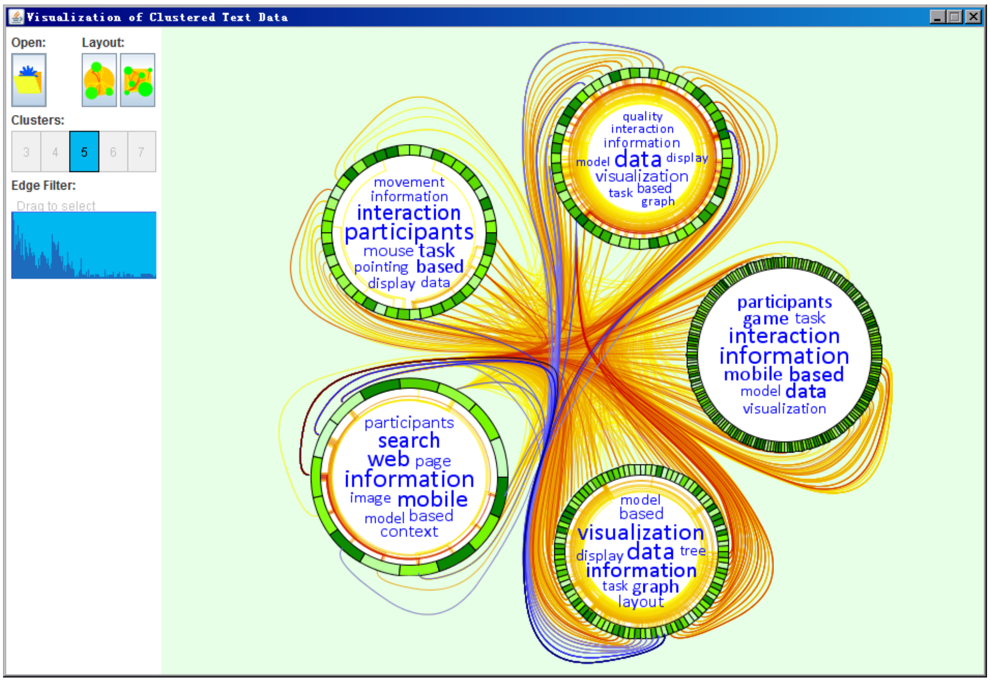

Figure 3.20: Third implementation in the circular cluster layout

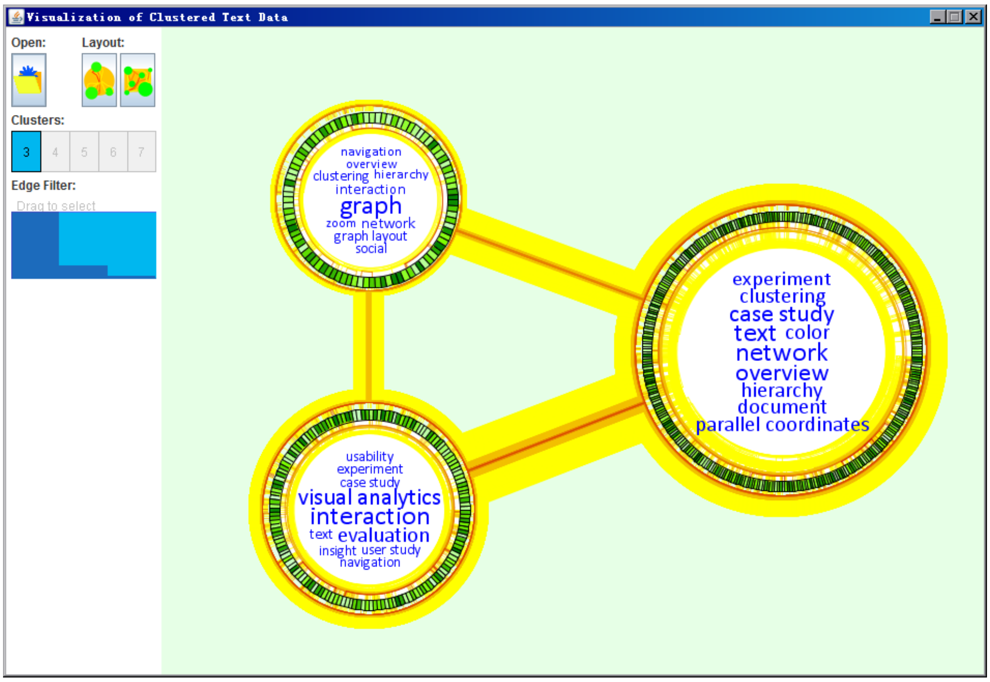

Figure 3.21: Third implementation in the free cluster layout

one of relationship weight, which decides the colors of relationship edges. Moreover, if the user switch an attribute as *None*, the attribute will be ignored by dataset loader. After loading a document network, the user can click the left-bottom button to open the dataset explorer. It lists document information with their titles and authors. When selecting a document in the view panel, the user can also see that the corresponding items are brushed in gradient colors by the relationship weight between each item and the selected one (see Figure 3.24).

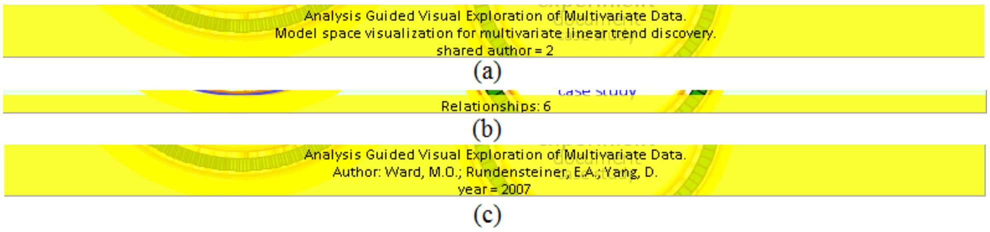

Figure 3.22: Data tips for (a) an edge, (b) edges and (c) a node.

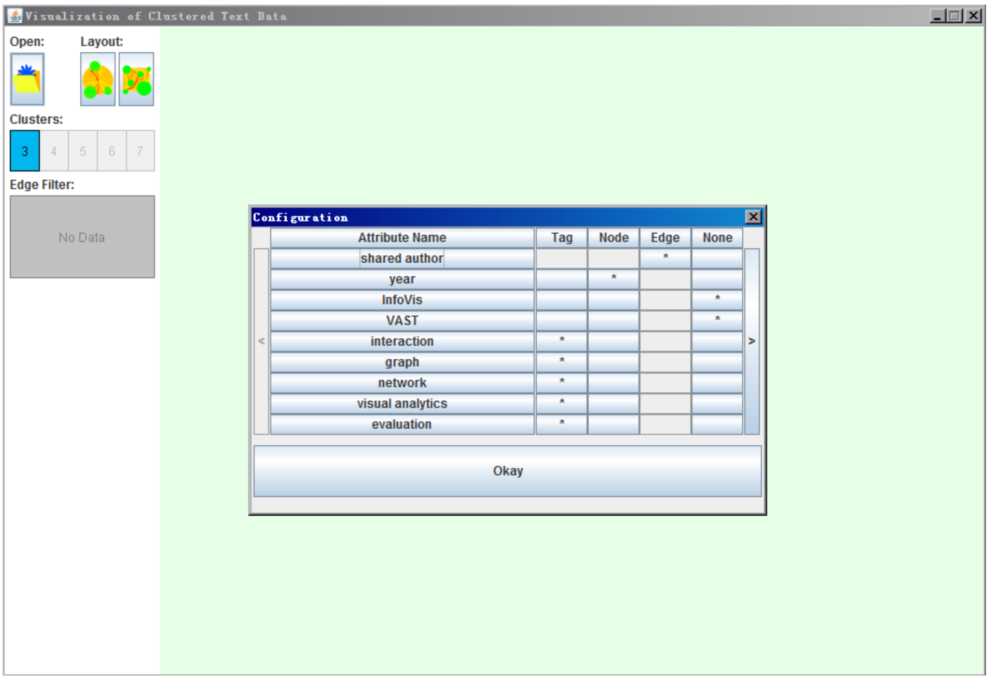

Figure 3.23: The pop-up dialog for configuration of visualization.

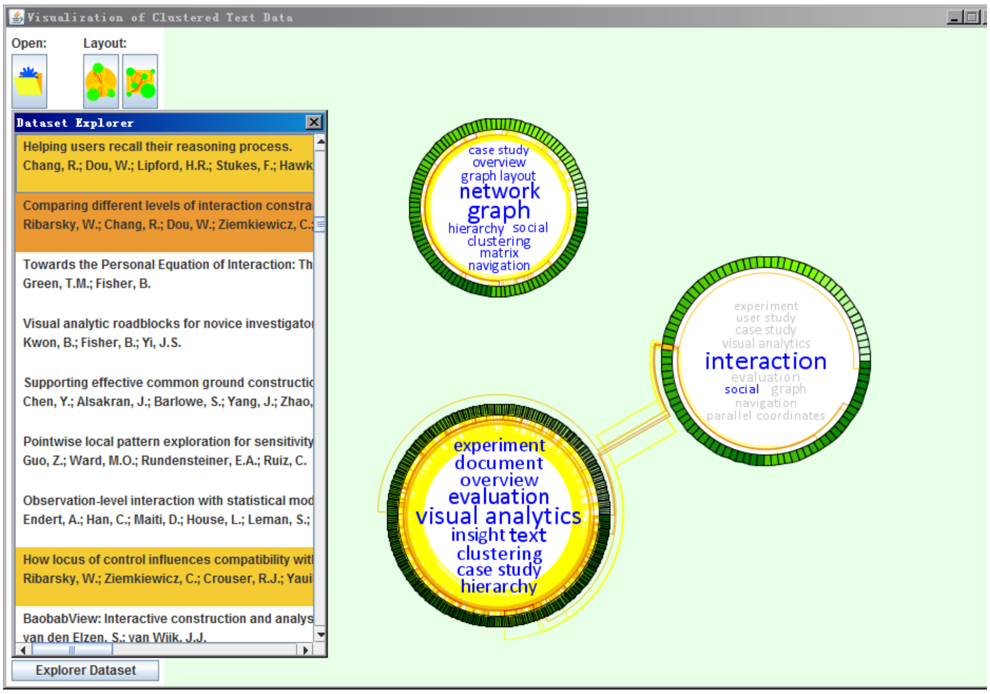

Figure 3.24: The dataset explorer list items in different colors with a document selected

4 Discussion

The chapter exhibits what we have done and how the application prototype performance is. Then it concludes our work. It also explores the advantages and disadvantages of our approach. We catch up with some ideas to improve the prototype.

4.1 Development Result

The result in concept is described as below. This description guides us create the hierarchy of the packages and classes in a clear and tidy way. It also helps others understand the implementation to continue to improve the application.

1. the application can parse a file with dataset which contains a network

$$\begin{aligned} N_{raw} =& (V_{raw}, E_{raw}, A_{raw}, R_{raw}) \text{ with} \\ & V_{raw} = \{document\} \\ & E_{raw} = \{(document_i, document_j) | i \neq j\} \\ & A_{raw} = \bigcup_{document \in V_{raw}} tag(document) \\ & R_{raw} = \{(x, a, r) | x \in V_{raw}, a \in tag(x), r \in \Re\} \\ & \quad \cup \{(x, weight, r) | x \in E_{raw}, r \in \Re\} \end{aligned} \tag{4.1}$$

2. the document vertices are clustered by tags with the weight in R_{raw} using K-means algorithm [6] and get a new network

$$\begin{aligned} N_{new} =& (V_{new} = V_{raw}, E_{new}, A_{new} = A_{raw}, R_{new} = R_{raw}) \text{ with} \\ & E_{new} = E_{raw} \cup \{e_c | e_c \subset V_{raw}\} \end{aligned} \tag{4.2}$$

3. the network is finally visualized follow the view graph which consists of the function F and L explained in previous chapter. The equations

contains the structure of packages and classes; also the detail of algorithms. For instance, the Equation 3.1 means there is a class which extends $java.awt.geom.Arc2D$ in Java to represent a document, since $Arc2D.PIE$ can define an $Arc2D$ instance as a pie. The Equation 3.2, 3.3 and 3.4 can show the procedure of arranging tags: for each tag in a cluster, it searches for a position along the specific spiral and the tag should not overlap the others in the cluster at the position.

Without any zooming, the prototype can display seven clusters in one view. The users can click the *open* button and there will be a pop-up dialog for choosing a file with a dataset to load. Then a waiting animation is shown in the view panel. In the background, it is initializing the visualization by parsing the selected file calculating clusters and rendering the dataset in memory. When everything is ready, the view panel presents the rendered result.

Figure 3.20 and Figure 3.21 show the results of our implementation in different layouts. In Figure 3.20, there are 421 documents, 1421 edges for relationships and 56 different attributes in 5 clusters in the circular cluster layout. In Figure 3.21, there are 578 documents with 1938 relationship edges and 81 different attributes in 3 clusters in the free cluster layout.

If the users click on the number of *clusters* on the control panel, a waiting animation will appear again until the visualization result is prepared. The *edge filter* is useful when the users would like to show some edges with weight values in a specific range. Right-clicking on removes the filter effect. The other interactions are supported as following the design discussed in Section 3.1.7 in the previous chapter.

4.2 Use case

This section presents an example to show how to use our tool. Every year there will be many conferences for people to discuss specific topics. The IEEE Information Visualization Conference (InfoVis) and the IEEE Conference on

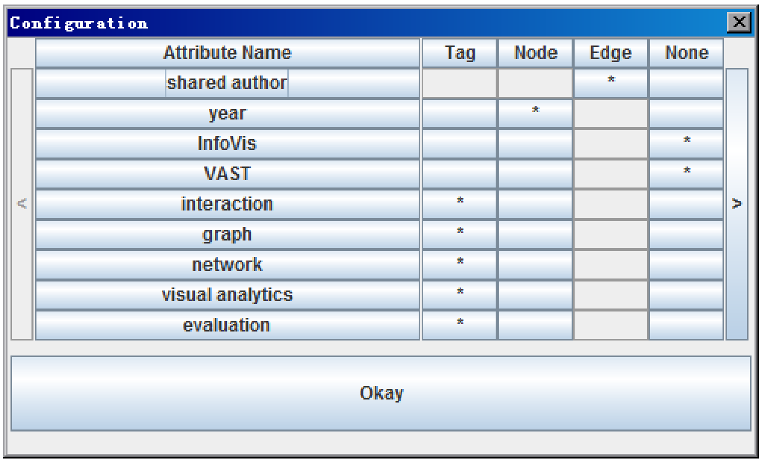

Figure 4.1: Use case: The visualization configuration dialog.

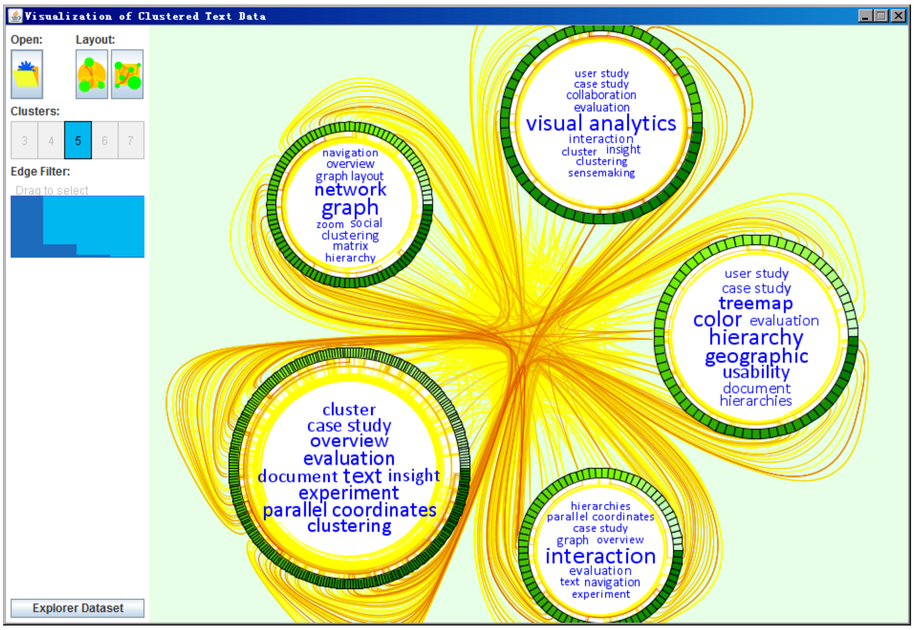

Figure 4.2: Use case: The visualization of the network of InfoVis and VAST papers in the circular cluster layout.

Visual Analytics Science and Technology (VAST) are for people publishing their fresh research results about visualization.

We consider a dataset with papers from the InfoVis conferences between 1995 and 2011; also from the VAST conferences between 2006 and 2011. Title, authors, keywords, published year are basic information. We collected the basic information of most papers from both conferences. Each piece of information can be treated as a node in a network. Then we compared authors between papers. If sharing at least one author, two papers should be connected by an edge with a weight which refers to how many authors they share.

Loading the file in the application, we should decide what to be read before visualizing (see Figure 4.1). The *InfoVis* and *VAST* can be switched off because we know all papers are from these two conferences and are not interested in both keywords. Then press *Okay* to visualize the dataset.

In a few seconds, the dataset is presented on the screen (see Figure 4.2). The documents are colored by published year. The newer published papers are displayed in darker color. The biggest tag refers to the topic of a cluster in general. If the sizes of tags in a cluster are almost similar, the cluster can be treated as *Cluster 'Others'*.

We can quickly grasp that the topic about visual analytics does not have a long history, since the documents from *Cluster 'Visual analytics'* are darker than the others. If interested in the topics about visual analytics and clustering, we can find the *Cluster 'Visual analytics'* and click the tag *clustering*. Then the related documents will be painted in red (see Figure 4.3).

We can get the interesting documents with them highlighted. If selecting one document, we will see the activities of its authors (see Figure 4.4). The authors who wrote the document were also active in some other projects. In the free cluster layout, the user can know which topic is important with edge halos. Ignoring *Cluster 'Others'* which has the tags of *text, experiment,*

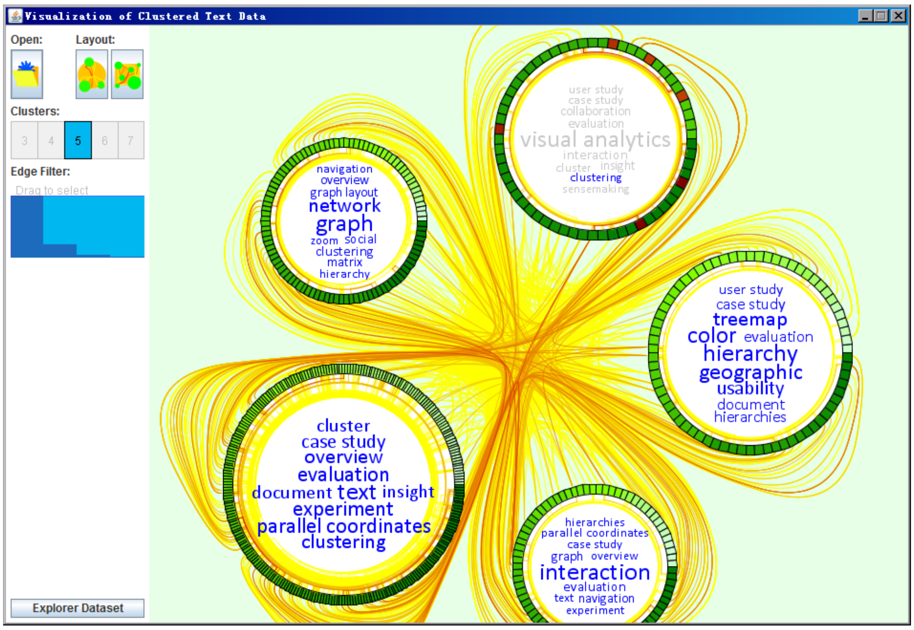

Figure 4.3: Use case: A user-interested tag is selected.

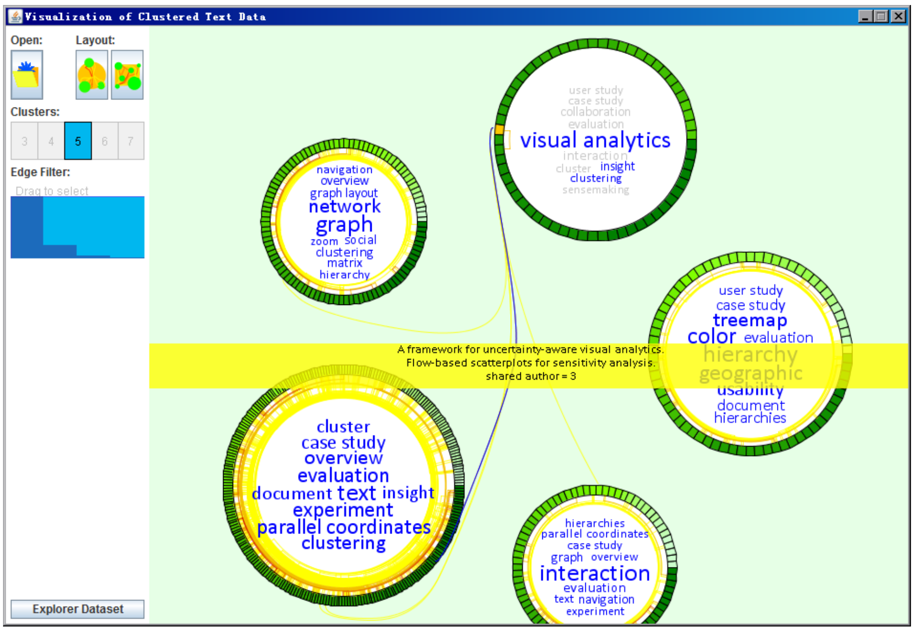

Figure 4.4: Use case: A user-interested document is selected.

parallel coordinates, evaluation and etc., we can guess the topic of *interaction* is important in the field of visualization since there are most relationships connected to *Cluster 'Interaction'* (see Figure 4.5).

To consider another dataset with papers from ISOVIS group, we calculate how much two documents are related as edge weight. By visualizing the document network, we find out that a document *57260736.pdf* has high relations with many articles. It implies the document is likely to be most worthy to read (see Figure 4.6).

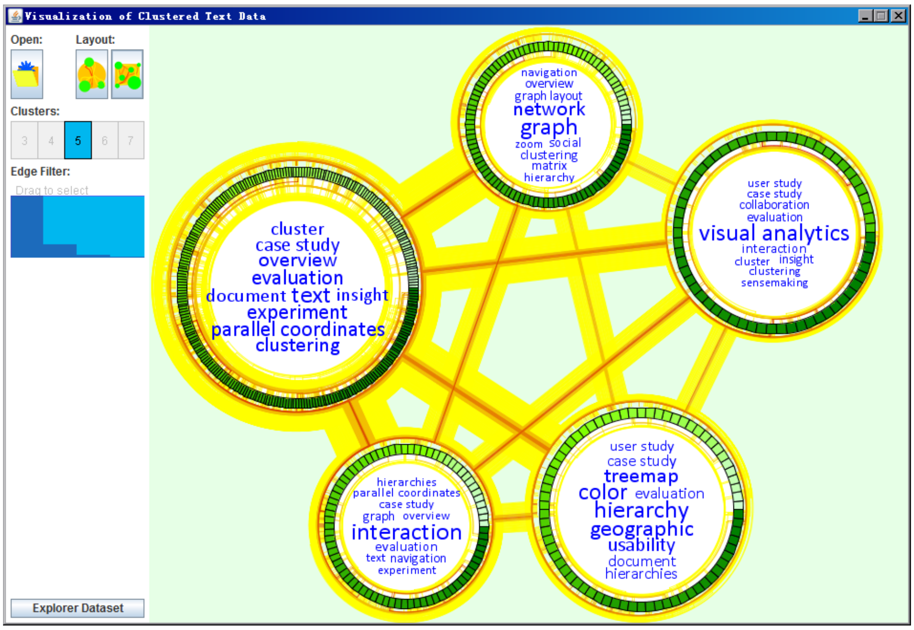

Figure 4.5: Use case: The visualization in the free cluster layout.

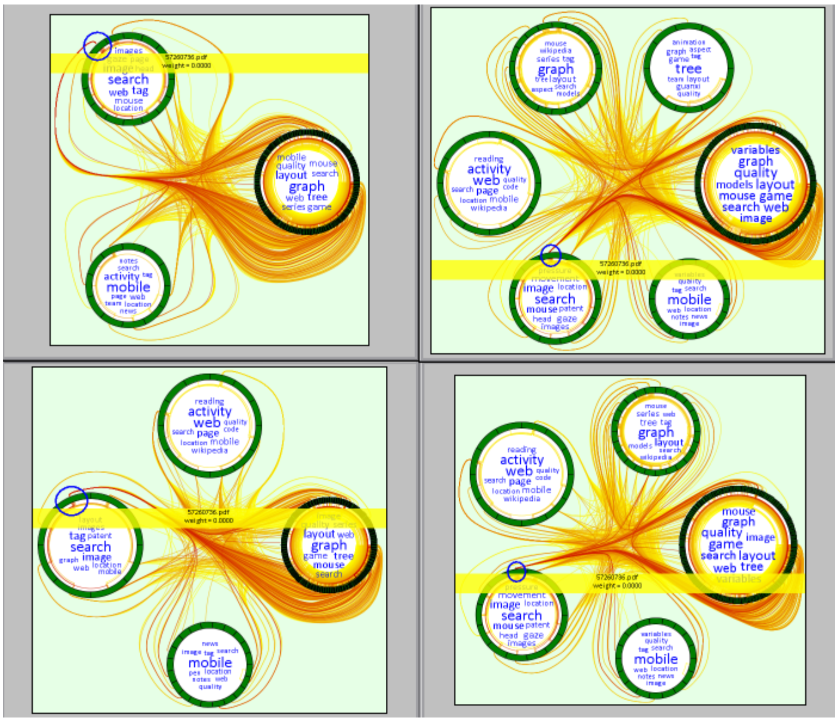

Figure 4.6: Use case: A dataset is visualized in the circular cluster layout by (up-left) three, (down-left) four, (down-right) five and (up-right) six clusters. A document worthy to read is marked by blue circle.

5 Conclusion

In Section 1.1, the goals are

- Read data from a file with tags and relationships of documents, cluster the documents by their tags and build a network of documents.
- Provide a clear overview of the network with some interactions to help users explore clustered documents and their relationships.

In Section 3.2.3, it is discussed that we realized the design of user interface which is explained in Section 3.1.7. There is a open button to enable users selecting a dataset and it is loaded into a network that described in Section 2.4. The prototype provide an overview of the network and we showed how to use it to explore clustered documents and their relationships in Section 4.1. Thus we achieved the both goals above.

After the development of design and coding with several times of testing and modifying, we created an application prototype to visualize the clustered text data with their relationships. The prototype can load a prepared dataset of collections of documents and reveal much information for the document network. The attributes of nodes, edges and clusters are visualized in one view. We also use a new approach to bundle edges called edge halo. It shows the overview of relationships inside a cluster and between clusters both well. Tag cloud represents the overview of a cluster clearly. The circular and free cluster layout with different edge bundling methods enable the users understanding document network via visualization by different feelings.

5.1 Advantages and Disadvantages

We are happy that the prototype provides a way to visualize all kinds of attributes for a network including the attributes of nodes, edges and even clusters in one view. It shows information contained in a network. It presents a good overview of relationships in clustered text data, especially with edge halo. The visualization displays the information about document, relationship and cluster in just one view. Our tool can show many clusters with tags to tell users what the documents surrounding are talking about. The documents will also be painted in gradient colors to give some information about the meta-data like published time, the number of authors, the length of document, and etc. The relationships between documents are represented by curve edges to help users look for some most related documents.

However, the prototype does not support zooming. When the network is huge and the users would like to see the exact detail about one document, it is not satisfied. Moreover, documents are displayed in pies with a very small angle when a cluster has too many nodes. Then the users cannot see them clearly and are also hard to select interested one. In the circular cluster layout, the edge bundling algorithm performs not so well. In the free cluster layout, edge halos go through the space with beams of stright lines. If there are six or seven clusters, edge crossing will make the view messy.

5.2 Future Work

As the existence of the disadvantages of our application prototype, we would like to give some ideas here.

- **Zooming**. Since the nodes and edges are displayed in order. In 2D real space, they are not overlapped on each other. Thus the users can see more detail about the document network by the interaction of zooming in. Magic lens is helpful to have a partial detail in enlargement without losing the overview of the whole network [22].

- **More attributes for nodes**. The document pies in a cluster can just show only attribute attached to nodes. It can add more pies to display more information about attributes of nodes (see Figure 5.1 (a)).

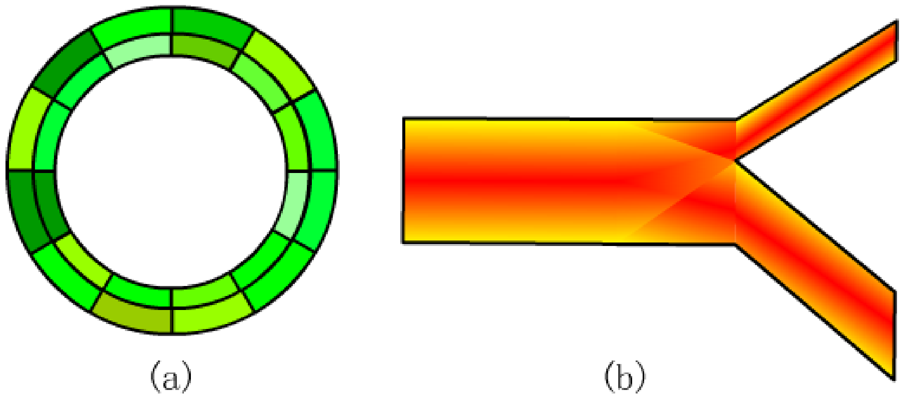

Figure 5.1: Future work: (a) more attributes for nodes and (b) advanced edge halo

- **Advanced edge halo**. The edge halos go through space by straight lines. The edge clutter is increasing while the number of clusters becomes more and more. The beams of straight lines can be replaced by smooth curves. And we can bundle the crossed edges first and then split them into several beams to reduce the clutter (see Figure 5.1 (b)). The solution of edge routing with ordered bundles [16] may be helpful to realize it.

- **Visualization Library**. The whole project does not import any third party libraries. The application is implemented by following the visualization pipeline model. It is easy to pick some of the packages and classes and refactory them into a visualization framework.

References

[1] A. Kerren, A. Ebert and J. Meyer (Eds.), *Human-Centered Visualization Environments, Volume 4417 of LNCS Tutarial*, NY: Springer, 2007.

[2] A. Kerren, J. T. Stasko, J.-D. Fekete and C. North, *Information Visualization: Human-Centered Issues and Perspectives, Volume 4950 of LNCS State-of-the-Art Survey*, NY: Springer, 2008.

[3] J. K. Gilbert, M. Reiner and M. Nakhleh, *Visualization: Theory and Practice in Science Education*, NY: Springer, 2008.

[4] C. Ware, *Information Visualization: Perception for Design*, MA: Morgan Kaufmann, 2000.

[5] A. Kerren, M. Kyusakova and C. Paradis (Eds.), *From Culture to Text to Interactive Visualization of Wine Reviews*. in F.T. Marchese and E. Banissi (Eds). NY: Springer, 2012, ch.5, pp. 85-110.

[6] B. Liu, *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data - Data-Centric Systems and Applications*. NY: Springer, 2011.

[7] G. S. Ingersoll, T. S. Morton and A. L. Farris, *Taming Text: How to Find, Organize, and Manipulate It*, NY: Manning Publications, 2013.

[8] F. B. Viegas, M. Wattenberg and J. Feinberg, ”Participatory Visualization with Wordle”, *IEEE Transactions on Visualization and Computer Graphics*, vol.15, no.6, pp. 1137-1144, 2009.

[9] K. Koh, B. Lee, B. Kim and J. Seo, ”ManiWordle: Providing Flexible Control over Wordle”, *IEEE Transactions on Visualization and Computer Graphics*, vol.16, no.6, pp. 1190-1197, 2010.

[10] N. H. Riche and T. Dwyer, ”Untangling Euler Diagrams”, *IEEE Transactions on Visualization and Computer Graphics*, vol.16, no.6, pp. 1090-1099, 2010.

[11] D. Holten, ”Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data”, *IEEE Transactions on Visualization and Computer Graphics*, vol.12, no.5, pp. 741-748, 2006.

[12] F. Nigsch, A. Bender, B. van Buuren, J. Tissen, E. Nigsch and J. B. Mitchell. ”Melting point prediction employing k-nearest neighbor algorithms and genetic parameter optimization”, *Journal of Chemical Information and Modeling*, vol.46, no.6, pp. 2412-2422, 2006.

[13] M. Balzer and O. Deussen, ”Level-of-detail visualization of clustered graph layouts”, in *The 6th International Asia-Pacific Symposium on Visualization*, Sydney, Australia, 2007, pp. 133-140.

[14] W. Cui, H. Zhou, H. Qu, P. C. Wong and X. Li, ”Geometry-Based Edge Clustering for Graph Visualization”, *IEEE Transactions on Visualization and Computer Graphics*, vol.14, no.6, pp. 1277-1284, 2008.

[15] D. Holten and J. J. van Wijk, ”Force-directed edge bundling for graph visualization”, in *Proceedings of the 11th Eurographics / IEEE - VGTC conference on Visualization*, Switzerland, 2009, pp. 983-998.

[16] S. Pupyrev, L. Nachmanson, S. Bereg and A. E. Holroy, ”Edge routing with ordered bundles”, in *Proceedings of the 19th international IEEE conference on Graph Drawing*, Eindhoven, The Netherlands, IEEE Computer Society Press, 2011, pp. 136-147.

[17] C. Collins, G. Penn and S. Carpendale, ”Bubble Sets: Revealing Set Relations with Isocontours over Existing Visualizations”, *IEEE Transactions on Visualization and Computer Graphics*, vol.15 no.6, pp. 1009-1016, 2009.

[18] C. Collins, G. Penn and S. Carpendale, ”Docuburst: visualizing document content using language structure”, in *Proceedings of the 11th Eurographics / IEEE - VGTC conference on Visualization*, Switzerland, 2009, pp 1039-1046.

[19] K. Kim, S. Ko, N. Elmqvist and D. S. Ebert, ”WordBridge: Using Composite Tag Clouds in Node-Link Diagrams for Visualizing Content and Relations in Text Corpora”, in *Proceedings of the 2011 44th Hawaii International Conference on System Sciences*, 2011, pp. 1-8.

[20] I. Jusufi. *Multivariate Networks: Visualization and Interaction Techniques*. Doctoral Dissertation. Linnaeus University, 2013.

[21] I. Jusufi, A. Kerren and B. Zimmer, ”Multivariate Network Exploration with JauntyNets”, in *Proceedings of the 17th International IEEE Conference on Information Visualisation*, London, IEEE Computer Society Press, 2013.

[22] E. A. Bier, M. C. Stone, K. Pier, W. Buxton and T. D. DeRose, ”Toolglass and magic lenses: the see-through interface”, in *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, Anaheim, CA, 1993, pp. 73-80.

[23] S. K. Card, J. Mackinlay, ”The structure of the information visualization design space”, in *Proceedings of the 1997 IEEE Symposium on Information Visualization*, Phoenix, AZ, 1997, p. 92.

[24] E. H.-H. Chi, ”A framework for information visualization spreadsheets”, Doctoral Dissertation, University of Minnesota, MN, 1999.

[25] J. Woch, *Integrated Natural Language Generation With Schema-Tags*. VA: Ios Pr Inc, 2003.

[26] G. Strang, *Linear Algebra and Its Applications*. KY: Brooks Cole, 2006.

[27] M. J. Halvey, M. T. Keane, ”An assessment of tag presentation techniques”, in *Proceedings of the 16th international conference on World Wide Web*, Banff, Alberta, Canada, 2007, pp. 1313-1314.

[28] S. Pupyrev, L. Nachmanson and M. Kaufmann, ”Improving layered graph layouts with edge bundling”, in *Proceedings of the 18th international IEEE conference on Graph drawing*, Konstanz, Germany, IEEE Computer Society Press, 2010, pp. 329-340.

[29] J. S. Yi, Y. ah Kang, J. Stasko and J. Jacko, ”Toward a Deeper Understanding of the Role of Interaction in Information Visualization”, *IEEE Transactions on Visualization and Computer Graphics*, vol.13, no.6, pp. 1224-1231, 2007.

[30] A. Kerren and F. Schreiber, ”Toward the Role of Interaction in Visual Analytics”, in *Proceedings of the Winter Simulation Conference*, Berlin, IEEE Computer Society Press, 2012, pp. 420:1-420:13.

[31] Wikipedia (2013, June). B*é*zier curve [online].
Available: https://en.wikipedia.org/wiki/Bezier_curve

[32] C. G*ö*rg, M. Pohl, E. Qeli and K. Xu, ”Visual Representations”, in *In Human-Centered Visualization Environments, Volume 4417 of LNCS Tutorial*, A. Kerren, A. Ebert, J. Meyer, Eds. NY: Springer, 2007, pp. 163-230.

Linnæus University
School of Computer Science, Physics and Mathematics

SE-391 82 Kalmar / SE-351 95 Växjö
Tel +46 (0)772-28 80 00
dfm@lnu.se
Lnu.se/dfm