8-12-2016

Text and Spatial-Temporal Data Visualization

Xi He

TEXT AND SPATIAL-TEMPORAL DATA VISUALIZATION

by

Xi He

Under the Direction of Ying Zhu PhD

ABSTRACT

In this dissertation, we discuss a text visualization system, a tree drawing algorithm, a spatial-temporal data visualization paradigm and a tennis match visualization system. Corpus and corpus tools have become an important part of language teaching and learning. And yet text visualization is rarely used in this area. We present Text X-Ray, a Web tool for corpus-based language teaching and learning and the interactive text visualizations in Text X-Ray allow users to quickly examine a corpus or corpora at different levels of details: articles, paragraphs, sentences, and words. Level-based tree drawing is a common algorithm that produces intuitive and clear presentations of hierarchically structured information. However, new applications often introduces new aesthetic requirements that call for new tree drawing methods. We present an indented level-based tree drawing algorithm for visualizing parse trees of English language. This algorithm displays a tree with an aspect ratio that fits the aspect ratio of the newer computer displays, while presenting the words in a way that is easy to read. We discuss the design of the algorithm and its application in text visualization for linguistic analysis and language learning. A story is a chain of events. Each event has multiple dimensions, including time, location, characters, actions, and context. Storyline visualizations attempt to visually present the many dimensions of a story's events and their relationships. Integrating the temporal and spatial dimension in a single visualization view is often desirable but highly challenging. One of the main reasons is that spatial data is inherently 2D while temporal data is inherently 1D. We present a storyline visualization technique that integrate both time and location information in a single view. Sports data visualization can be a useful tool for analyzing or presenting sports data. We present a new technique for visualizing tennis match data. It is designed as a supplement to online live streaming or live blogging of tennis matches and can retrieve data directly from a tennis match live blogging web site and display 2D interactive view of match statistics. Therefore, it can be easily integrated with the current live blogging platforms used by many news organizations. The visualization addresses the limitations of the current live coverage of tennis matches by providing a quick overview and also a great amount of details on demand.

INDEX WORDS: Text Visualization, Spatial-Temporal Data Visualization, Storytelling, Storygraph, Tree Drawing Algorithm, Line Crossing, Tennis Match Visualization.

TEXT AND SPATIAL-TEMPORAL DATA VISUALIZATION

by

Xi He

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2016

TEXT AND SPATIAL-TEMPORAL DATA VISUALIZATION

by

XI HE

Committee Chair:	Ying Zhu
Committee:	Eric Friginal
	Scott Owen
	Yanqing Zhang

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

August 2016

DEDICATION

This dissertation is dedicated to my always encouraging parents, Zhennan He and Zhongxian Gan. Thank you for all the unconditional love, guidance, and support that you have always given me.

ACKNOWLEDGEMENTS

This dissertation is the result of research carried out over a period of many years. During this period, many people supported my work and helped me to bring it to a successful conclusion, and here I would like to express my gratitude. This dissertation work would not have been possible without the support of many people. I want to express my gratitude to my advisor Dr. Ying Zhu for his support and invaluable guidance throughout my study. His knowledge, perceptiveness, and innovative ideas have guided me throughout my graduate study.

I also present my words of gratitude to the other members of my thesis committee, Dr. Eric Friginal, Dr. Scott Owen, and Dr. Yanqing Zhang for their advice and their valuable time spent in reviewing the material. I also would like to extend my appreciation to my colleagues in Graphics and Visualization Group for all the support and ideas and to everyone who offered me academic advice and moral support throughout my graduate studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

- NLP - Natural Language Processing

PART 1

INTRODUCTION

1.1 Data Visualization

1.1.1 What Is Data Visualization?

Data visualization refers to the techniques used to communicate data or information by encoding it as visual objects contained in graphics. People always look for structure, features, pattern, trends, anomalies and relationships in data, and data visualization supports this by presenting data in various forms with differing interactions. Data visualization can provide a qualitative overview of large and complex data sets, can summarize data, and can assist in identifying regions of interest and appropriate parameters for more focused quantitive analysis. On one hand, effective visualization helps users analyze and reason about data and evidence. It makes complex data more accessible, understandable and usable. On the other hand, data visualization can communicate information clearly and efficiently via statistical graphics, plots and information graphics. Numerical data may be encoded using dots, lines, or bars, to visually communicate a quantitative message.

Often, visualization is seen as a process of generating pretty pictures from the datasets. This is a myth. Good data visualizations provide great insights on the data. A fitting example of data visualization is John Snow's dot distribution map of 1854 Broad Street cholera outbreak. John Snow is an English physician who used dot distribution map to show that cholera is spread by contaminated water instead of air. In 1954 there was a severe outbreak of cholera that occurred near Broad Street in the Soho district of London, England. The dominant miasma theory at that time believed the cholera was caused by pollution or a noxious form of "bad air". Since the germ theory was not established at this point, so people was unaware of the mechanism by which the disease was transmitted. But Snow believe that the spread of cholera was not due to breathing foul air. He creates a dot distribution map to

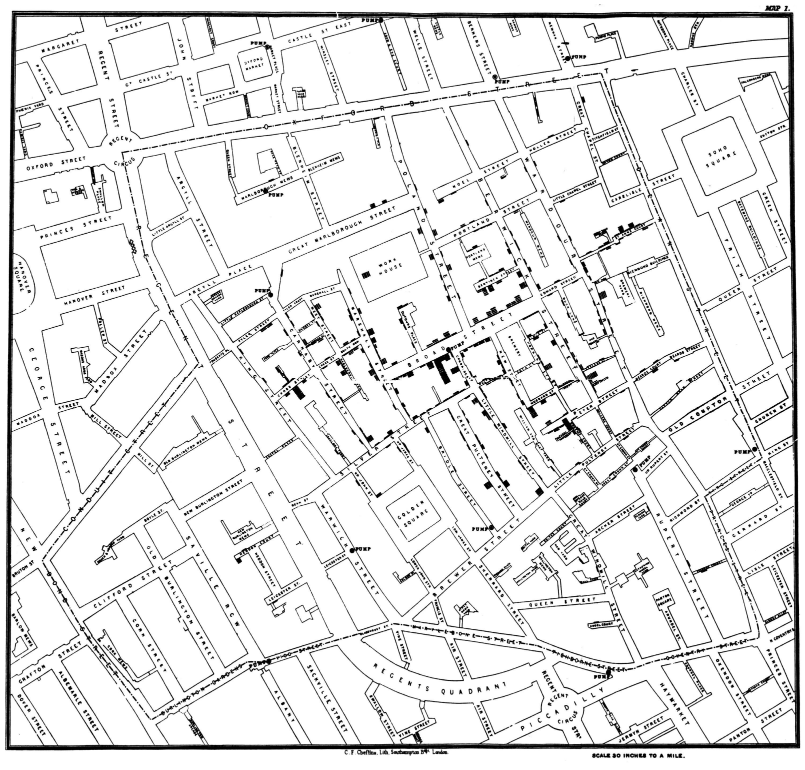
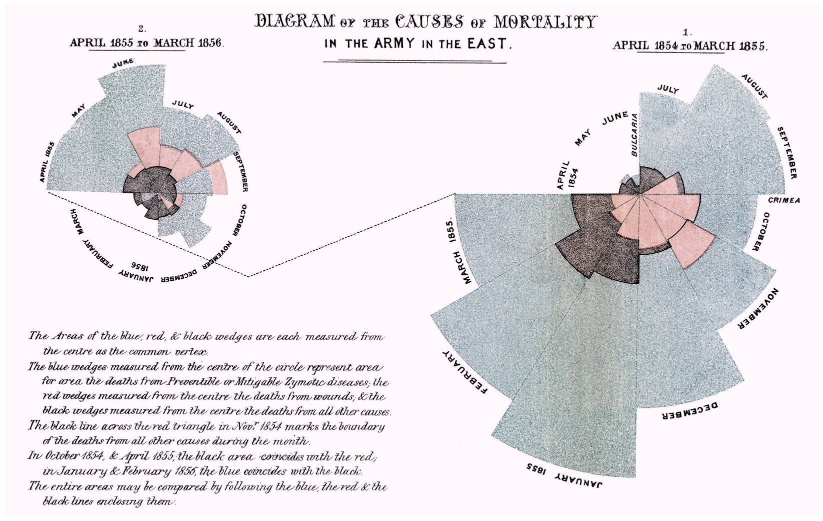


Figure (1.1) Original map by John Snow showing the clusters of cholera cases in the London epidemic of 1854. The pump is located at the intersection of Broad Street and Little Windmill Street [1].

study the pattern of the cholera cases. The base map of John Snow's dot distribution map (Figure 1.1) is a simple road network, with few buildings named or depicted. Cholera deaths are depicted along the road network in their correct locations by address, with quantities measured by parallel tick marks stacked next to the road. With this dot distribution map, Snow illustrated how cases of cholera were centered on the pump, and with a much more elaborate investigation he found out the the cholera is spread by the contaminated water in the pump.

Another important purpose of data visualization is to advocate and communicate effectively with a wide audience. For example, Florence Nightingale, who was a manager of nurses in the Crimean War, discovered some revealing facts with the data she collected in two years: the majority of solders were dying not because of wounds caused during the battle



The Areas of the blue, red, & black wedges are each measured from the centre as the common vertex.
The blue wedges measured from the centre of the circle represent area for area the deaths from Preventible or Mitigable Zymotic diseases; the red wedges measured from the centre the deaths from wounds; & the black wedges measured from the centre the deaths from all other causes.
The black line across the red triangle in Novr 1854 marks the boundary of the deaths from all other causes during the month.
In October 1854, & April 1855, the black area coincides with the red; in January & February 1856, the blue coincides with the black.
The entire areas may be compared by following the blue, the red & the black lines enclosing them.

Figure (1.2) Nightingale Diagram showing the causes of mortality in the army in the Crimean War. Blue represents deaths occasioned by diseases, red stands for the deaths due to wounds and black for all other causes of death.

but due to infections inside the hospital, triggered by neglecting hygiene conditions. When Florence Nightingale came back from the war and tried to show her findings and proposal, however, she faced many difficulties and challenges. She finally manages to receive an audition, but she needs to make her presentation concise but impactful. Florence Nightingale then came up to the idea of what we know today as the Nightingale Diagram or Nightingale Rose, a circular graph that shows the amount of deaths per cause throughout the months (See Figure 1.2). As we can see, the picture is self-explanatory of the issue Nightingale wanted to point out. The clarity of this diagram enabled her to convince the Authorities in London of the necessity of a change in the sanitary conditions, as the diagram left no space for misinterpretation.

The Naepolean's march to Russia in the eighteenth century (Figure 1.3) by Minard is another great example of data visualization and is considered one of "the best statistical drawings ever created" [3]. This visualization is impressive because of its ability to combine all of dimensions: loss of life at a time and location, temperature, geography, historical context, into one single graphic. It shows these various details without distracting text

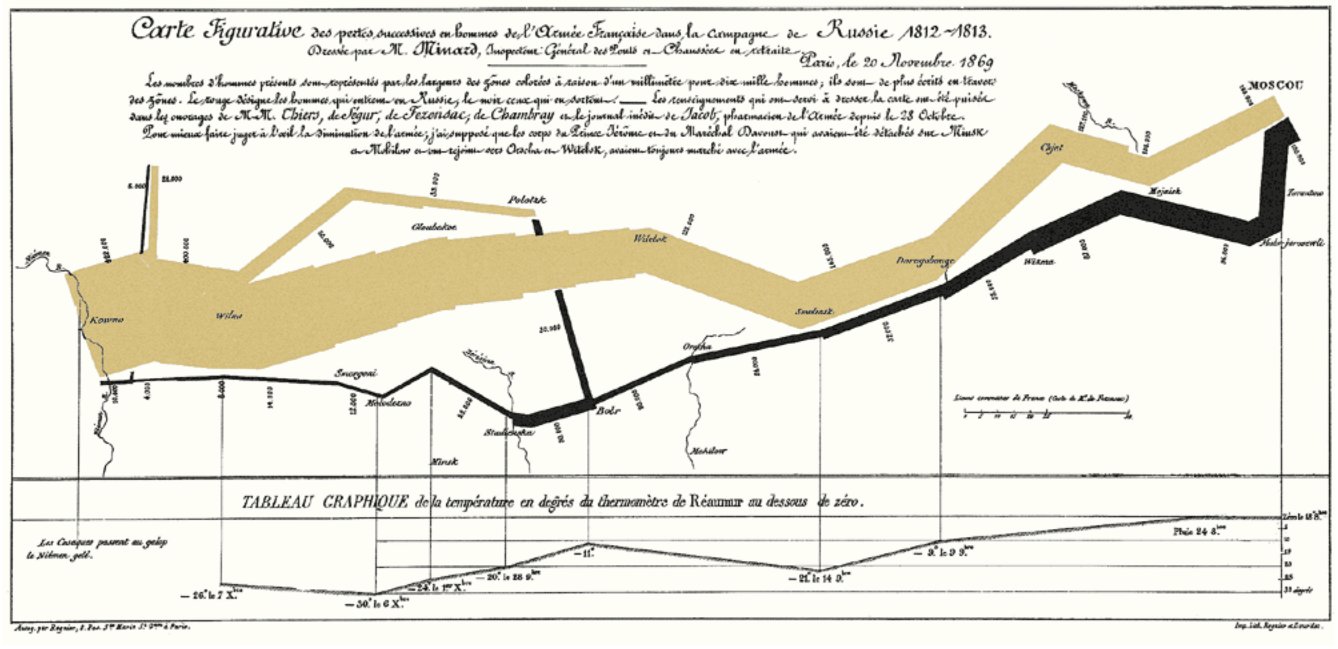


Figure (1.3) Naepolean's march to Russia by Minard. The pale yellow polyline denotes the advance march path towards Russia and the black polyline denotes the retreat path from Russia. The width of the lines corresponds to the size of the army at that point. The bottom line graph shows the temperature (inverse).

or labels as well. For example, It displays the points where Napoleon's troops divide into subgroups by breaking out the main bar into branches.

1.1.2 Visualization Types

According to [4], data visualization can be classified as the following categories and Figure 1.4 shows some examples:

- 1-dimensional: linear data types include textual documents, program source code, and alphabetical lists of names which are all organized in a sequential manner.
- 2-dimensional: planar or map data include geographic maps, floorplans, or newspaper layouts. Examples include Cartogram, Choropleth and Dot Distribution Map.
- 3-dimensional: real-world objects such as molecules, the human body, and buildings

have items with volume and some potentially complex relationship with other items. Examples include 3D computer models, surface and volume rendering and computer simulations

- Temporal: time lines are widely used and vital enough for medical records, project management, or historical presentations to create a data type that is separate from 1-dimensional data. Examples include Connected Scatter Plot, Polar Area Diagram and Time Series.
- Multidimensional: most relational and statistical databases are conveniently manipulated as multidimensional data in which items with n attributes become points in a n-dimensional space. Examples include Pie Chart, Histogram and Scatter Plot.
- Hierarchical: hierarchies or tree structures are collections of items with each item having a link to one parent item (except the root). Examples include Dendrogram, Ring Chart and Tree Diagram.
- Network: sometimes relationships among items cannot be conveniently captured with a tree structure and it is useful to have items linked to an arbitrary number of other items. Examples include Alluvial Diagram, Node-Link Diagram and Matrix.

1.2 Problem Statement and Motivation

1.2.1 Text X-Ray

A corpus is a large collection of texts. In general, corpora are not meant to be read, but to be processed and analyzed by computer programs, which are called corpus tools. For example, Google's Ngram Viewer is a corpus tool that allows users to search the Google Books corpus and visualizes the use of selected English words over many years. Corpus tools have been developed for many areas such as biomedical research, linguistic research, intelligence analysis, literature critic, marketing, etc.

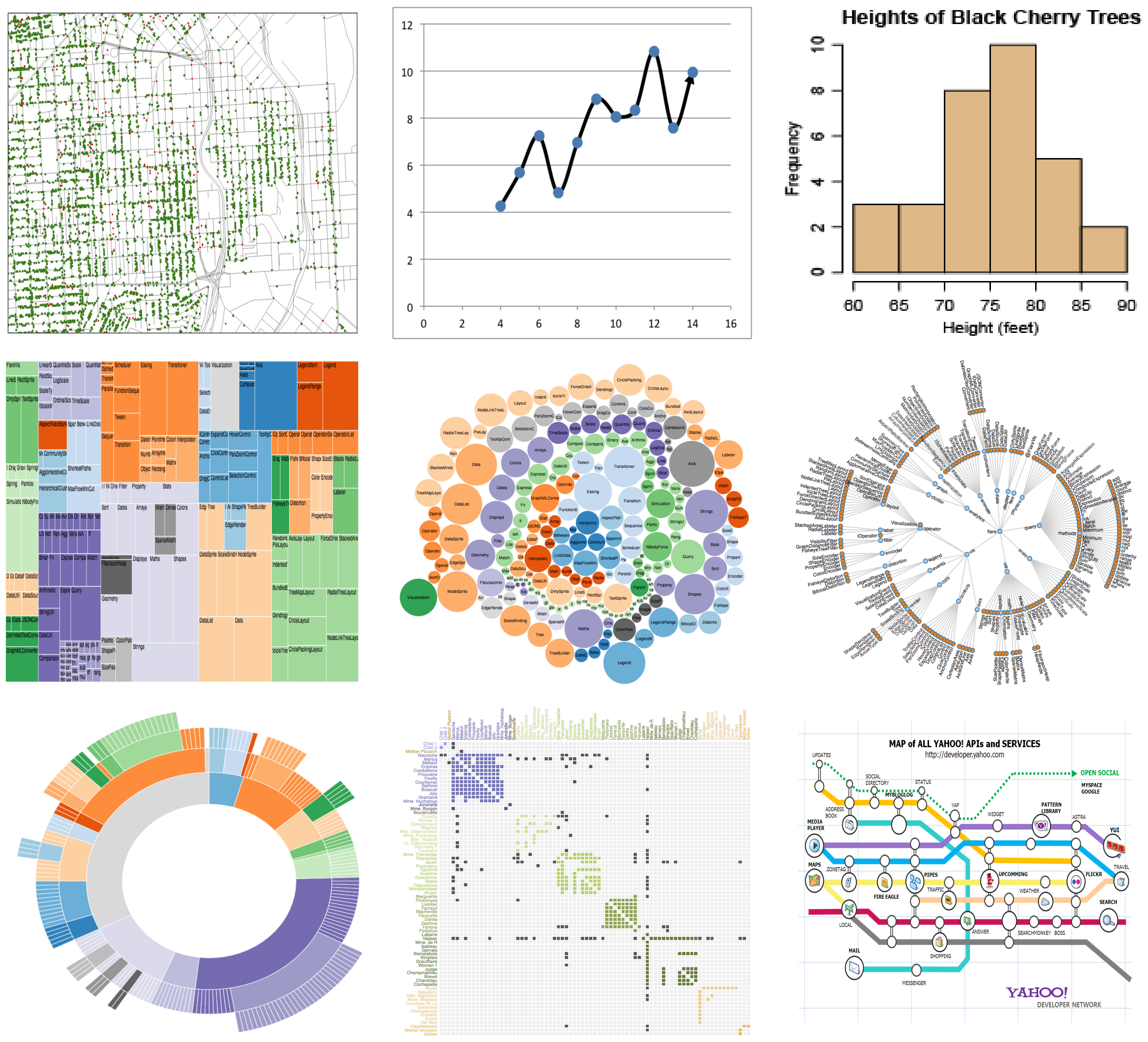


Figure (1.4) From top to bottom and left to right, dot distribution map (2-dimensional), connected scatter plot (temporal), histogram (multidimensional), tree map (multidimensional), bubble chart (multidimensional), radial tree (hierarchical), wedge stack graph (hierarchical), matrix (network), tube map (network). [2]

The value and effectiveness of corpus-based language learning and teaching have been established in previous studies [5–7]. For example, teachers can guide students to use a concordancer to study a corpus and build their knowledge about language use. Although many corpus tools have been developed for studying English language, they were mostly designed for linguistic research, not for language teaching and learning. For example, many corpus tools are designed for specific corpora and don't allow users to import their own corpus. In addition, the user interfaces of many corpus tools are designed for language specialists. Students and teachers often find them unhelpful in analyzing their own writings.

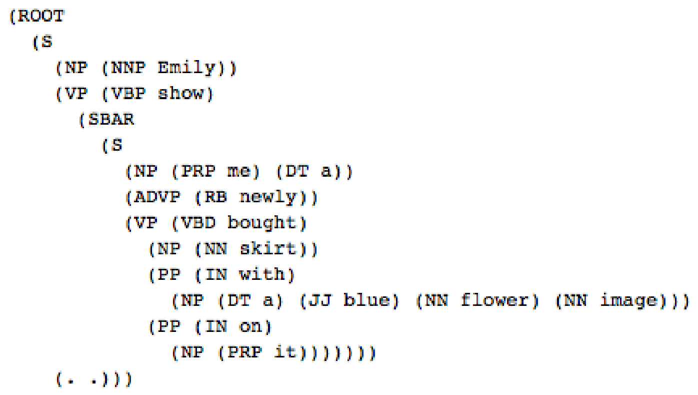
1.2.2 Tree Drawing Algorithm and Text Visualization

Tree drawing is one of the most researched areas in data visualization. They are widely used in biology, business, chemistry, software engineering, artificial intelligence, Web site design, data analysis, education, and social networks.

In graph theory, a tree is an undirected graph without simple cycles. A typical tree consists of nodes and edges. Each node represents an entity. Each edge represents the relationship between entities. A tree with a root node is called a rooted tree.

A tree drawing algorithm consists of a set of rules for placing the nodes and drawing the edges. Some tree drawing rules are introduced to address the characteristics of the structure of the data. Some tree drawing rules are aesthetic rules for the efficient use of space and clarity of presentation. The most important aesthetics of tree drawings include area, aspect ratio, subtree separation, closest leaf, and farthest leaf, etc. A new application may introduce new aesthetic rules that lead to new tree drawing algorithms.

I am developing a text analysis and visualization program for linguistic studies and language learning. One of the main features is the visualization of the parse tree for each sentence. A parse tree is a rooted tree showing the syntactic structure of a sentence or a string (Figure 1.5). Visualizing the parse trees can help researchers or students analyze the structure of the sentence and its complexity. The typical drawing of a parse tree is a top-down, level-based tree (Fig. 1.6). This type of drawing is intuitive and clear. But the drawback is that the aspect ratio of the tree does not fit the aspect ratio of newer computer displays. The tree grows vertically. The height of the parse tree visualization is usually larger than its width, particularly when the sentence is structurally sophisticated. However, the standard aspect ratio of computer displays after 2012 is 16:9, with the width larger than the height. With a dual monitor setup, the display is even wider. Therefore a traditional parse tree visualization does not make optimal use of the screen space. When multiple parse trees need to be displayed in a sequence, this problem becomes even more obvious.



```
(ROOT
  (S
    (NP (NNP Emily))
    (VP (VBP show)
      (SBAR
        (S
          (NP (PRP me) (DT a))
          (ADVP (RB newly))
          (VP (VBD bought)
            (NP (NN skirt))
            (PP (IN with)
              (NP (DT a) (JJ blue) (NN flower) (NN image)))
            (PP (IN on)
              (NP (PRP it)))))))
    (. .)))
```

Figure (1.5) The syntactic structure of the sentence “Emily show me a newly bought skirt with a blue flower image on it.”

1.2.3 Spatio-temporal Storyline Visualization

A story is an account of related events, and each event can be seen as a multidimensional data item. Storyline visualizations attempt to visually present the sequence of events and the relationships among multiple dimensions of these events. Therefore storyline visualization is a type of multidimensional data visualization.

Events in different stories may have different dimensions. But most events share some common dimensions such as characters, time, location, and actions. Many other dimensions can be added. For example, events can be classified into different types (such as topics or themes). The description of an event can be classified into different sentiment types. An event is often related to a set of other events, which form its context.

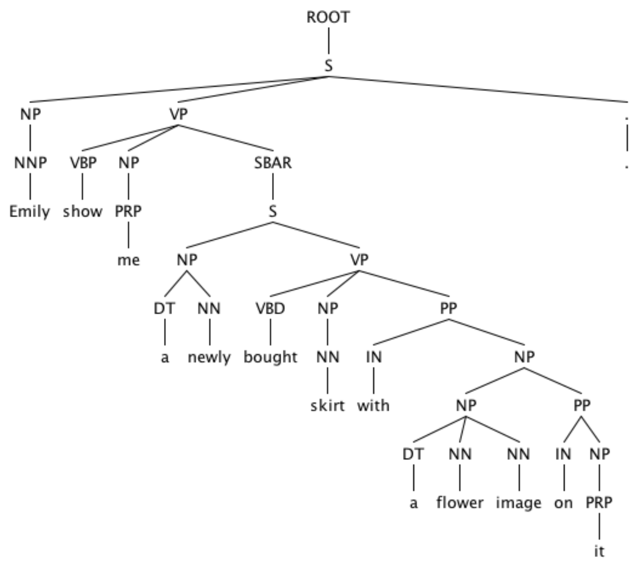


Figure (1.6) The traditional level-based parsed tree of the sentence “Emily show me a newly bought skirt with a blue flower image on it.”

The many existing storyline visualization techniques different in what dimensions they choose to depict and how each dimension is visually presented. In this work, I focuses on visualization techniques that try to integrate both time and location in one view. I call this type of problem integrated spatial-temporal storyline visualization.

Integrating both time and location in one view is often desirable. It allows users to correlate temporal and spatial information quickly. However, such integration has long present a challenging problem. Existing spatial-temporal storyline visualizations can be classified into four categories: map-based visualizations, timeline-based visualizations, unconventional, and hybrid approaches. Map-based visualization designs center around a map. The location information is visualized on the map, while other dimensions such as time are displayed either on top of the map in a 3D view or overlaid on the map in a 2D view. The timeline-based visualizations center around a timeline chart. Time information is usually mapped to the horizontal axis, while other dimensions such as location are encoded in the chart. Hybrid approaches use both map-based and timeline-based visualizations in a synchronized multiple view configuration.

Existing timeline-based methods maintain regular ordering for the temporal dimension but have irregular ordering for the spatial dimension. On the other hand, existing map-based methods maintain regular ordering for the spatial dimension but have irregular ordering for the temporal dimension.

1.2.4 Tennis Match Visualization

Currently, important tennis matches are streamed online or broadcasted on TV. Many news media also provide online live blogging on important matches (see [8]). There are two limitations to this kind of live coverage. First, they provide a linear, one dimensional narrative of the match. One knows what is happening now but it is difficult to find out quickly what has happened before. For online streaming or TV broadcasting, it's difficult to go back in time to watch earlier points. In live blogging, readers can go back and read comments on a previous game, but they have to page through many comments in between. One cannot quickly jump to a particularly interesting point. Second, current live coverage is not interactive. Users cannot get a quick answer to questions such as “How many Aces has Roger Federer served?” or “How many backhand winners has Stan Wawrinka hit?”

1.3 Text X-Ray

To address this issue discussed above, I developed a corpus tool, Text X-Ray, that helps teachers and students to analyze their own corpus and compare their own corpus with other corpora. Text X-Ray consists of a natural language processing engine, a visualization engine, a corpus analysis engine, corpus databases, and a visualization interface. The Web based visual interface provides visualizations of a corpus at every levels of detail: articles, paragraphs, sentences, and words. The goal of this tool is to make it easier for users to see the structures, patterns, and relationships that are not easily recognized in plain English text. This is why it is called Text X-Ray.

Comparing with other related corpus tools, Text X-Ray introduces a much more visual approach to language teaching and learning. Traditional corpus tools provide statistical analysis of the texts, which are useful but do not give the full picture. Text visualizations can fill that gap. For example, two articles may have similar readability indices but quite different writing styles. One article may use longer sentences but simpler words, while the other one uses simpler sentences but more difficult words. Such differences are readily visible in Text X-Ray's visualizations.

Text X-Ray is also more user friendly and versatile. Linguistic researchers can use it to analyze and compare articles in a corpus. Students can use it to compare their own writings with a corpus. To help teachers use it in classroom settings, Text X-Ray also supports corpus management and user management.

1.4 Tree Drawing Algorithm and Text Visualization

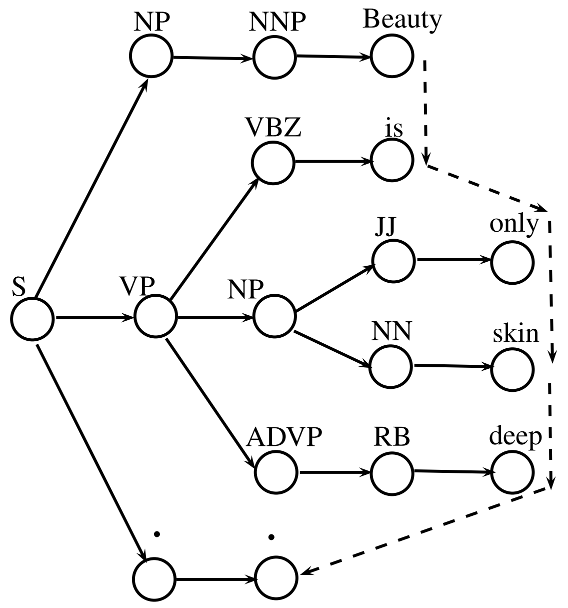


Figure (1.7) The visualization of a parsing tree for the sentence "Beauty is only skin deep.". Note that the words are displayed vertically and the sentence is hard to read.

To address the display issues in the application and many similar cases, it is more desirable to display the parse tree horizontally with the root node on the left for the optimal use of space. In addition, the leaf nodes (i.e. the words) should be placed in such a way that they can still be read from left to right as a sentence. This becomes a new aesthetic rule for tree drawing. Simply drawing a tree horizontally with a level-based algorithm is not

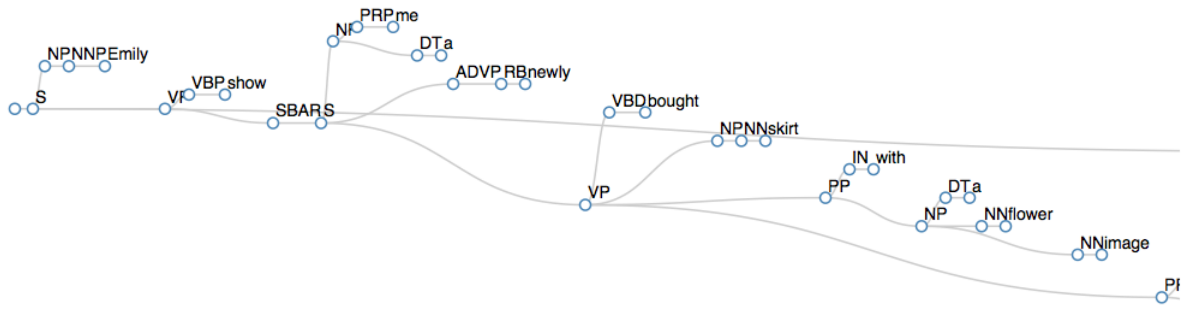


Figure (1.8) The simple modification of level-based parsed tree of the sentence “Emily show me a newly bought skirt with a blue flower image on it.”

user friendly because readers have to read the sentence vertically (see Figure 1.7). A simple modification of the existing level-based tree algorithm to create an indented display of words also does not work because line crossings make the tree difficult to read (see Figure 1.8).

To address this issue, I propose a new indented level-based tree drawing algorithm that preserves the grammatical structure of the parse tree but also allows users to read the sentence from left to right. The resulting tree visualization fits the aspect ratio of most computer displays better than the traditional parse tree visualization. I also analyze the time complexity of this algorithm and discuss the implementation of this algorithm using JavaScript and d3.js.

1.5 Spatio-temporal Storyline Visualization

I presents a novel hybrid approach for integrated spatial-temporal storyline visualization. My method include a timeline-based visualization and a synchronized map. The timeline-based visualization is based on our previously published work but with significant improvement. A major problem with timeline-based approach is the large number of line crossovers. Here I present a new heuristic algorithm to reduce line crossings for my timeline-based visualization technique. I also show that my timeline-based method maintains regular ordering for both temporal and spatial dimensions.

Through case studies, I further demonstrate that this hybrid approach is effective in providing a comprehensive visualization of the multiple dimensions of a story. Particularly notable is its effectiveness of depicting the context of events.

1.6 Tennis Match Visualization

Our visualization technique addresses these issues by presenting tennis match data in a 2D interactive view. This Web based visualization provides a quick overview of match progress, while allowing users to highlight different technical aspects of the game or read comments by the broadcasting journalists or experts. Its concise form is particularly suitable for mobile devices. The visualization can retrieve data directly from a tennis match live blogging web site. Therefore it does not require extra data feeding mechanism and can be easily integrated with the current live blogging platform used by many news media.

Designed as “visualization for the masses”, this visualization is concise and easy to understand and yet can provide a great amount of details on demand. Because it is a 2D view, users can quickly jump to an interesting point without paging or dragging slider bars. Users can interact with the visualization by typing a question and see the technical data being highlighted in the visualization. The visualization is designed primarily for general public, but serious fans or tennis experts can also use this visualization for analyzing match statistics.

PART 2

RELATED WORK

2.1 Corpus Tools

The most popular corpus tools for linguistic studies include the BYU corpora [9], AntConc, WordSmith, Sketch Engine, Sarah, Monoconc Pro, WMatrix, etc. [10]. There are generally two types of corpus tools: (1) corpus tools developed for a particular corpus or particular corpora; (2) corpus tools that can process different corpora. My tool, Text X-Ray, belongs to the second category.

The main issue with the most existing corpus tools is that they are not designed for language learning. In a recent review of corpus tools, Laurence Anthony [10], author of the popular AntConc program [10], pointed out that the current corpus tools are "... generally researcher-centric in that they do not always lend themselves to easy use in the classroom with learners." He also points out that students "need a corpus tool that gives them access to a corpus in an easy and intuitive way. They also need a tool to show them results that are immediately applicable to a given learning task ..."

A number of corpus tools, such as Compleat Lexical Tutor [11], AntConc [10], Word and Phrase [12], have attempted to address this issue by making them more useful for teachers and students. For example, Compleat Lexical Tutor [11], a Web based corpus tool, allows users to enter their own texts or a corpus and then make a concordance for all the words in the text. AntConc [10] also provides similar functions. Wordandphrase.info [12] allows users to load their own texts and then highlight all of the medium and lower-frequency words, based on selected corpora. This helps students focusing on learning the new, low frequency words.

The main difference between Text X-Ray and the above corpus tools is the visualization interface. The existing corpus tools display statistical analysis and highlight texts, but with limited user interactivity. Text X-Ray retains important functions found in most corpus tools but provides a much more visual and interactive user experience. In particular, Text X-Ray's interactive sentence bar and parse tree visualization are new features that are not seen in existing corpus tools.

Although many text visualization techniques have been developed [13], text visualization is rarely used in linguistic research and education. Siirtola, et al. [14] pointed out that "There are very few exploratory visualization and analysis tools that are linguistically motivated ..., and even fewer that allow rapid exploration of linguistic parameters." I examined recent surveys [13–15] on text visualization techniques and found only a few text visualization techniques for linguistic research [14], and no comparable work on language learning and teaching. This is an area that hasn't been sufficiently explored, and my work is an attempt to address this issue.

2.1.1 Characteristics of text visualization in language teaching and learning

Text visualization for language teaching and learning needs to meet particular requirements. These requirements are not unique to language teaching and learning, but they are emphasized more in this area than in others.

- The texts should be displayed along side the visualizations. In many text visualization techniques, the visualizations replace the texts. The original texts are often not displayed in the interface. But in language teaching and learning, the texts need to be examined at all times. Visualizations should support the examination of the original text, not replacing it. Text visualizations should be linked and synchronized with the text display.

- For language teaching and learning, highly innovative and abstract data visualization should be introduced with great care. Because users often switch between the original text and the visualizations, the form of the visualizations should be closer to the conventional textual display for easier mental transition and connection. Some text visualization techniques are difficult to use because they require too much mental adjustment from one form of display to another. Therefore the complexity and abstractness of the visualization needs to be controlled.

2.2 Tree Drawing Algorithms

2.2.1 Level-Based Tree Drawing Algorithms

The level-based tree drawing (also called layered tree drawing) is the most popular tree drawing algorithm. The five main aesthetic rules for level-based tree drawing include :

Aesthetic rule #1: Nodes of a tree at the same height should lie along a straight line, and the straight lines defining the levels should be parallel.

Aesthetic rule #2: A left child should be positioned to the left of its ancestor and a right child to the right.

Aesthetic rule #3: A father should be centered over its children.

Aesthetic rule #4: A tree and its mirror image should produce drawings that are reflection of one another; moreover, a subtree should be drawn the same way regardless of where it occurs in the tree.

Aesthetic rule #5: Small, interior subtrees should be spaced out evenly among larger subtrees. Small subtrees at the far left or far right should be adjacent to larger subtrees.

A tree drawing algorithm needs to calculate the position of each tree node in a way so that the resulting tree is aesthetically pleasing and conserves space.

Knuth first published an algorithm for drawing binary trees [16]. In 1979, Wetherell and Shannon [17] presented an $O(n)$ algorithm that satisfies aesthetic rules #1–3 while at the same time minimizes width. A similar algorithm was developed by Sweet [18]. In 1981, Reingold and Tilford [19] presented an algorithm that was inspired by the Wetherell and Shannon algorithm and addressed its flaws by satisfying aesthetic rule #4. Then in 1990, Walker [20] presented an improved method that satisfies aesthetic rule #5 for trees of unbounded degree. In 2002, Buchheim et al. [21] improved Walker's algorithm so that drawing trees of unbounded degree can be run in $O(n)$ instead of $O(n^2)$ in Walker's algorithm. For a recent survey on tree drawing algorithms, please refer to [22].

My proposed algorithm falls into the category of level-based tree drawing. In this algorithm, I introduce a new aesthetic rule:

Aesthetic rule #6: A tree should be draw from left to right, with the leaf nodes indented in such a way that they can be read from left to right as a sentence.

2.2.2 Non-Layered Tree Drawing Algorithms

The level-based tree drawing algorithms assume that nodes in the tree have uniform size. However, in many practical applications, these tree nodes may have varying sizes. Non-layered tree drawing algorithms are therefore designed to generate a more vertically compact drawing which places child nodes at a fixed distance from the parent nodes. Miyadera et al. present an $O(n^2)$ algorithm [23] for non-layered trees that horizontally positions parent nodes at a fixed offset from their first child, instead of centering above the children. Many other algorithms, such as Bloesch algorithm [24], Stein and Benteler algorithm [25] and Li and Huang algorithm [26], Marriot et al. algorithm [27] and Ploeg algorithm [28], employ the similar idea. After preprocessing such as discretizing, they run the extended Reingold and Tilford algorithm [19] to draw the non-layered trees.

2.3 Storyline Visualization Techniques

Many storyline visualization (or narrative visualization) techniques have been proposed [29, 30]. It's useful to classify them for clarity and comparison. Segel and Heer [29] proposed an excellent high level classification of storyline visualizations. But I am interested in how temporal and spatial dimensions are handled in these visualization, particularly how to integrate the temporal and spatial dimensions in a single visualization view. Storyline visualizations that do not deal with either time or location, such as Zhao, et al. [31], are not reviewed in this paper. Here I propose a lower level classification of storyline visualizations based on how temporal and spatial dimensions are handled.

The temporal dimension is very important for storyline visualization. Therefore most storyline visualizations include a temporal dimension. But some storyline visualizations do not need a spatial dimension [32–36].

For visualization techniques that depict both temporal and spatial dimensions, I further divide them into two categories: separated temporal and spatial views and integrated temporal and spatial views. For example, Zhu and Chen [37] show temporal dimension in a timeline and locations in a separate but synchronized table view. However, the majority of the papers I reviewed fall into the second category.

2.3.1 Integrated temporal and spatial views

Integrate temporal and spatial dimension in a single view is very desirable but difficult to achieve. The main reason is that spatial dimension is inherently 2D while temporal dimension is inherently 1D. The existing methods can be classified into four categories: map-based methods, timeline-based methods, hybrid methods, and unconventional methods.

Map-based methods can be further divided into two categories: 2D map-based methods, and 3D map-based methods. In 2D map based methods, the main challenge is how to show temporal information on a 2D map. Liu [38] draw timeline as a circle around the map. The temporal information is not directly overlaid on the map but instead linked through color coding. Sun, et al. [39] and Liu, et al. [40] experimented with different ways of overlaying temporal data on a 2D map: text label, visual symbol, color coding, and animation. In particular, they noted the difficulty of encoding timeline directions on a 2D map.

A major problem with 2D map-based methods is that the ordering of the temporal dimension is often broken. In their classification of storyline visualization techniques, Segel and Heer [29] identified three dimensions of the narrative structure tactics: ordering, interactivity, and messaging. Based on that, I further propose the concept of regular ordering and irregular ordering of temporal and spatial data. If the order and pace of temporal data is maintained, then I say the visualization has regular ordering for temporal data. If the geographical relationships of the spatial data is maintained, then I say the visualization has regular ordering for spatial data. In 2D map-based methods, the spatial data has regular ordering but the temporal data has irregular ordering. For example, events that happen at the same time are not grouped together and they are difficult to be identified quickly. For example, it's difficult to clearly differentiate multiple events that happen at the same location but at different times. It's also difficult to show the speed of movements on a 2D map. The event sequences are only maintained locally but not globally. It's difficult to compare the sequence of events on two locations that are not close. Bak, et al. [41] attempted to solve this problem by visualizing time as a circle around a location. This deals with event overlapping problem but the sequence of events are not maintained. It's also not well suited for dense data sets. This is why I say in 2D map-based methods, the temporal data has irregular ordering.

3D map-based methods attempt to solve these issues by drawing timelines above a map. For example, Gatalsky, et al. [42], Andrienko, et al. [43], and Kapler, et al. [44] use 3D maps to integrate spatial and temporal data in one view. With 3D map, both spatial and temporal data have regular ordering. However, 3D view has its own drawbacks [45]. One of the main problems is occlusion. It's also difficult to perceive distance or height precisely in 3D views. Users need to see objects from multiple angles simultaneously to have a clear sense of their distances.

Timeline-based methods center around a timeline. Location information needs to be encoded in this chart. In Munroe's hand drawn storyline visualization [46], locations are marked by the distance between lines at each timeframe. Crnovrsanin, et al. [47] adopted a similar approach: location is mapped to distances between entities or certain locations. Tanahashi and Ma [48] and Tanahashi, et al. [49] use a closed contour surrounding the events to depict location. Liu, et al. [50] improved on Tanahashi and Ma's [48]approach by handling location hierarchy with nested contours. The same location may be represented by multiple contours on the chart and they are linked by the same color coding. Both methods require substantial data preparation.

Timeline-based methods have regular ordering for temporal dimension but irregular ordering for spatial dimension. For example, the same location may appear multiple times in different places on a timeline-based visualization. The 2D location data is depicted in 2D shapes but locations of these 2D shapes have no relevance to the location information they depict. Instead, the location proximity is depicted in color. This is likely to cause some confusion. In some visualizations, only the distance between locations are depicted, not the geographical location itself. In other words, the 2D location information is reduced to 1D.

Another potential problem in timeline-based approach is excessive line crossovers [48]. Tanashida and Ma [48] proposed a number of design principles for storyline visualization and one of them is to minimize line crossovers.

The flaws in both map-based visualizations and timeline-based visualizations show the difficulty of integrating temporal and spatial information in one view.

Unconventional methods depict temporal and spatial data in more abstract and unfamiliar ways. For example, Cao, et al. [51] proposed a circular layout. Timeline is presented as nested rings. Locations are circles on the rings. The distance between circles reflect the distance between their locations.

Many visualizations may be classified as hybrid methods with synchronized timeline-based and map-based views. For example, Crnovrsanin, et al. [47], Chu, et al. [52], and Wang and Yuan [53] are hybrid methods.

My method is an attempt to address some of the issues in timeline-based approaches discussed above. My method is a hybrid method with a timeline-based visualization and a synchronized map view. The map view only show the location data without temporal information. My timeline-based visualization is different from previous timeline-based methods because my method maintains both regular ordering for temporal dimension and spatial dimension. In addition, I address the problem of excessive line crossovers by proposing a new heuristic method for reducing line crossings.

2.4 Tennis Match Visualization

We first give a general overview of sports data visualization [54] and then focus on tennis data visualization.

Sports data visualizations can be classified based on multiple parameters: data, visualization techniques, target audience, and domain.

Sports data can be roughly divided into three categories: on-court performance data, game statistics, and off-court statistics. On-court performance data are often collected by sensors or advanced imaging techniques. For example, the Hawk-Eye systems can provide tennis player's location, speed, ball speed, ball trajectory, etc. Special sensors put on racquets can record racquet speed. News media sometimes visualize performance data (e.g. player position and movement) and superimpose them on live images. In general, on-court performance data are not freely available.

Game statistics are compiled by human and often freely available [55]. They include scores, number of errors, number of aces, number of double faults, number of forehand or backhand winners, etc. Off-court statistics [56] may including player's ranking, age, height, weight, number of titles won, salary, prize money earned, etc.

2.4.1 Visualization techniques

The visualization techniques are generally chosen based on data types. On-court performance data are often displayed as markers, heatmaps [57–59], or trajectory lines superimposed on a court image [60]. Game statistics and off-court statics are displayed in a wide variety of visualizations [61–65].

2.4.2 Target audience

Sports data visualizations often target three types of audiences: experts, serious fans, and general public. For experts, the goal of data visualization is to help their data analysis. Therefore the visualizations tend to display more technical details gathered from on-court performance data, with high data density and interactions. For serious fans, the data visualizations often serve dual purposes. On one hand, they are used for casual analysis. On the other hand, they are treated as a type of visual art. For example, many data visualizations are designed to be posters or infographic [66]. This type of visualizations often feature colorful, complicated, and creative visual forms. For general public, the goal of sports data visualization design is to provide quick overview as well as easy interaction. The visual forms should be simple and clean, with details on demand.

The visualization techniques proposed in this paper is based on game statistics. The visualization features a series of mini-timelines arranged on horizontal bars. The target audiences are general public and serious fans.

Because different sports have different rules and different data properties, it is often difficult to compare visualizations across different sports. In this review, we focus on related work in tennis data visualization. Polk, et al. [67] developed a tennis match data visualization system called TenniVis. Both TenniVis and our visualization focus on presenting game statistics. But there are two main differences. First, the data in TenniVis needed to be collected by human, while the data in our visualization are retrieved automatically from a tennis match live blogging Web site. This means TenniVis is suitable for post-match analysis but not for live coverage. Second, TenniVis is designed for expert analysis. With glyph based graphs, it displays many technical details in a complex visualization design. Our visualization is designed for general public. It features a simpler design of basic match statistics. In addition, our visualization is web based program, while TenniVis is a standalone toolkit.

Saunder's tennis visualizations [60] mostly deal with on-court performance data. With access to Hawk-Eye data, he superimposes data visualizations on a tennis court image in both 2D and 3D. He has also designed a game tree visualization of Nadal's game statistics in 2013. Saunder's main target audiences are experts. But his work is also featured in a documentary film and news media. In the latter case, the data visualizations are used as infographics. Our work is quite different from Saunder's work because we focus on game statistics rather than on-court performance data.

In summary, existing tennis data visualizations focus on post-match analysis for expert users. The main contribution of our work is its focus on automatic data collection and live match data visualization.

PART 3

TEXT X-RAY: AN INTERACTIVE TEXT VISUALIZATION TOOL

3.1 Overview

Text X-Ray [68] is an online interactive text visualization system that can accept user-uploaded corpora, processing them, and display them in both textual and visual forms. It contains a server-side subsystem and a browser-side subsystem. The server-side subsystem is designed to handle natural language processing that requires intense computation. It contains seven components:

- Web Server
- Natural Language Processing (NLP) Engine
- Corpus Analysis Engine
- Corpus Cache
- Corpus Management System
- Corpus Database
- Corpus API

The browser-side subsystem consists of Visualization Engine and Visualization Interface, which provides user interactions and data visualizations. The server-side and browser-side subsystems communicate via HTTP request/respond messages.

Figure 3.1 shows the main components and the workflow in Text X-Ray. On the server side, there are three major pipelines. The first pipeline is for processing corpora. Once corpus processing requests are received, Web Server will check whether the requests have been processed before. If yes, then the requests will be sent to Corpus Cache, which will send

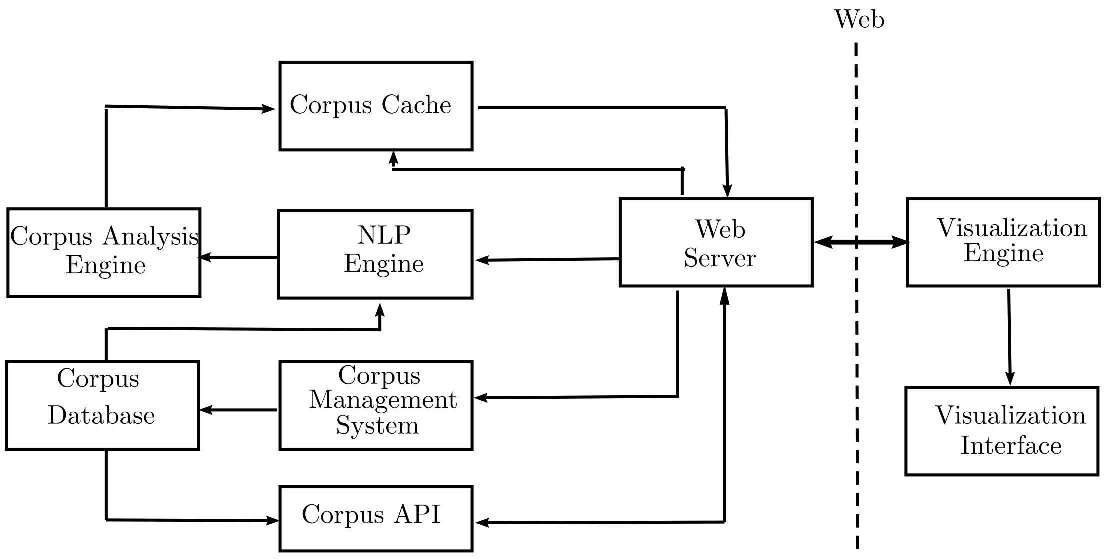


Figure (3.1) Workflow in Text X-Ray

the processed corpus data back to the browser. Otherwise, NLP Engine takes the requests and retrieves the requested corpus from the Corpus Database, and parses it. The results are delivered to Corpus Analysis Engine for further text analysis. Finally the processed corpus is cached in Corpus Cache and also sent back to the browser. The second pipeline is designed to allow corpus management. After receiving requests from Web Browser, Corpus Management System performs Add/Modify operations on Corpus Database. The third pipeline provides API for corpus based computation. Corpus API contains a set of functions that manage Corpus Database.

A requested corpus with plain text is processed in the server-side subsystem, transformed into a hierarchical object and sent to the browser-side subsystem. The hierarchical object not only retains the original full-text information, but also maintains identifiable information for paragraphs and sentences. Associated statistics for separate paragraphs and sentences, and sentence parsed trees are also contained in the hierarchical objects.

On the browser side, users can choose a corpus and explore the articles in that corpus. Visualization Engine will load the entire processed data for a specific corpus from the server side before users can interact with Text X-Ray. Such design is to guarantee that the user interaction is smooth and not affected by network transmission. Once corpus data is loaded, Visualization Engine constructs data visualization and is ready to respond to user inputs from Visualization Interface.

The server-side subsystem is written using Java servlet with Stanford Natural Language Parser [69] and Google Gson libraries. The browser-side subsystem is developed with jQuery UI [70] and D3 [71] library.

3.2 Natural Language Processing (NLP) Engine

The NLP Engine is built on top of Stanford Natural Language Parser [69]. The Stanford Parser is a probabilistic parser that uses the knowledge from human parsed sentences to analyze the structure of new sentences. The Stanford Parser provides Java APIs that can analyze and construct the grammatical structure of sentences. In Text X-Ray, I use the Stanford Parser to construct a grammar tree for each sentence. I also use Stanford Parser as a Part-Of-Speech (POS) tagger. The POS tags are stored in the grammar trees. The grammar trees are stored in an internal data structure. The text analysis and text visualization, particularly the sentence tree visualization, are based on these grammar trees.

For each article in a corpus, Text X-Ray extracts paragraphs and then divide it into sentences. It then uses Stanford Parser to process each sentence and construct a grammar tree. These grammar trees are forwarded to Corpus Analysis Engine.

The main concern in developing the NLP engine is the speed of parsing. Natural language processing is quite time consuming, especially for a large corpus consisting of thousands of sentences. To speed up the NLP engine, I have developed a multi-threading NLP engine using the Java executor framework. The NLP engine set up a thread pool with configurable number of threads in it. Then sentences are evenly distributed to the thread for processing. When done, the parsed sentences are collected and reorganized in the original order. On a 1.9 GHz quad-core processor, the engine can process an 8,000-word New Yorker article in about 5 seconds.

3.3 Corpus Analysis Engine

Corpus Analysis Engine has two primary functionalities. The first one is to compute linguistic statistics of the corpora. It contains a collection of functions that implement various linguistic analysis algorithms. Currently implemented functions include:

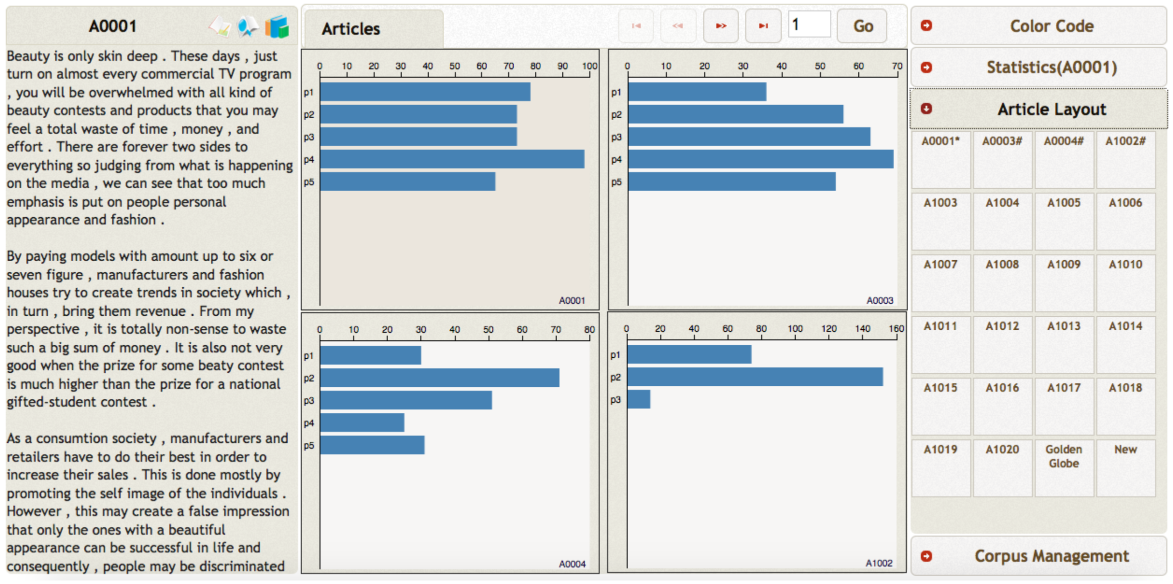
- Identify thirty-one Part-Of-Speech (POS) items such as nouns, verbs, adjectives, etc.
- Identify long words
- Identify long sentences
- Calculate readability indices
- Calculate lexical density
- Calculate word frequency

The second functionality of the corpus analysis engine is to prepare corpora for text visualization. Texts in corpora do not provide enough information for text visualizations. Visualization Engine needs to be able to identify paragraphs, sentences and words in corpora, and obtain information such as the length of a sentence or the POS tag of a word in order to visualize them. The solution in Corpus Analysis Engine is to transform a corpus into a hierarchical data structure of article objects, paragraph objects, sentence objects, and word objects, making the navigation of these objects efficient.

3.4 Visualization Engine and Visualization Interface

Visualization Engine and Visualization Interface constitute a standalone browser application. Visualization Interface is responsible for text visualization and user interaction while Visualization Engine controls the work flow and maintains visual system status.

Text X-Ray is divided into multiple panels: text panel, visualization panel, linguistic analysis panel, and control panels (Fig. 3.2). The text panel displays the currently selected



Beauty is only skin deep . These days , just turn on almost every commercial TV program , you will be overwhelmed with all kind of beauty contests and products that you may feel a total waste of time , money , and effort . There are forever two sides to everything so judging from what is happening on the media , we can see that too much emphasis is put on people personal appearance and fashion .

By paying models with amount up to six or seven figure , manufacturers and fashion houses try to create trends in society which , in turn , bring them revenue . From my perspective , it is totally non-sense to waste such a big sum of money . It is also not very good when the prize for some beaty contest is much higher than the prize for a national gifted-student contest .

As a consumtion society , manufacturers and retailers have to do their best in order to increase their sales . This is done mostly by promoting the self image of the individuals . However , this may create a false impression that only the ones with a beautiful appearance can be successful in life and consequently , people may be discriminated

Figure (3.2) Overview of Visualization Interface

article. The visualization panel, always parallel to the text panel, enable users to analyze the texts in five levels of detail: corpus, articles, paragraphs, sentences, and words. The linguistic analysis panel shows the output of linguistic analysis of the corpus or an article, such as readability indices, lexical density, etc. The control panel allows users to adjust the visualization settings and manage corpora or users.

As mentioned in the previous chapter, the text panel and visualization panel are displayed next to each other so that users can easily switch between reading visualization and reading the original text. The text and text visualization are also linked. For example, clicking on a sentence bar or a word block in the text visualization, the corresponding sentence or word is highlighted in the text window.

The corpus view allows users to have an overview of multiple articles in a corpus. Users can visually compare different articles for the length of the article and the length of the paragraphs. The paragraphs are visualized as horizontal bars.

The sentence bars, word blocks, and sentence trees (discussed below) give users a new way of analyzing the reading complexity of an article. Traditionally the complexity of an article (or reading difficulty) is measured and presented in various readability index numbers or lexical density number. (These index numbers are presented in the linguistic analysis panel.) Reading difficulty is usually calculated by counting long sentences, low frequency words, and complex sentence. But a single readability index does not convey enough information. For example, two articles may have similar readability indices but one article may have some very difficult paragraphs and some easy paragraphs, while the other article may have a more evenly distributed reading difficulty. A readability index may not distinguish these two articles, but my text visualization will. With my text visualizations a user (or author) can quickly see that certain parts of an article are more complex than other parts.

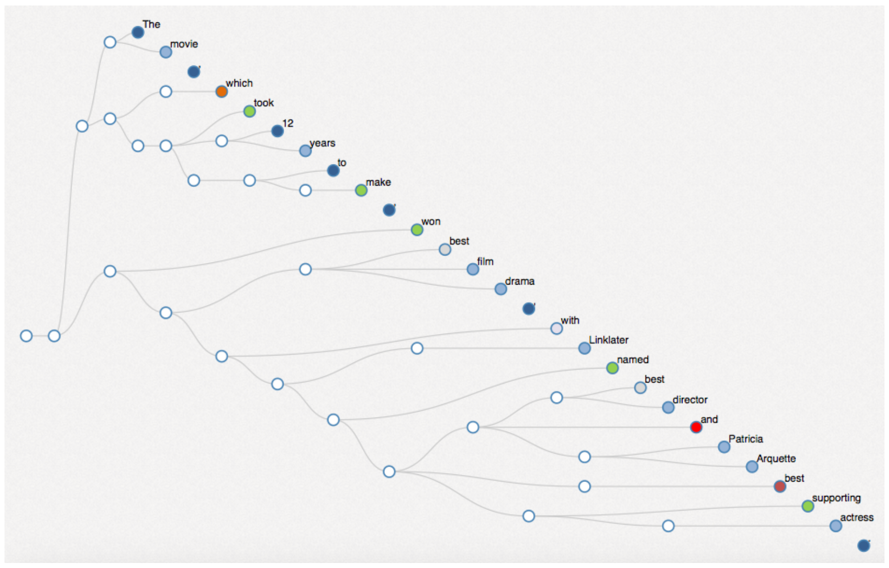


Figure (3.3) A sentence tree

My sentence tree visualizations (Fig. 3.3) show the grammatical structure of the sentences as parsed by the Stanford Parser. This visualization addresses a major weakness in the traditional reading difficulty analysis. In a plain English text, the grammatical complexity of the sentences cannot be quickly identified. Traditional readability indices only measure the complexity of a sentence by its length, not by its syntactic structure. The sentence tree visualizations makes it much easier for viewers to see and compare the structure of the sentence and its complexity. I also add color coding to the sentence tree visualization. By color coding words (i.e. leaf nodes) by their frequency or syllable count, the sentence tree visualization can show sentence length, sentence structure, and word difficulty in one view. To the best of my knowledge, none of the existing readability measures can achieve this.

The sentence tree visualization is implemented using my own Indented Level-base Tree drawing algorithm [72], which is based on a classic tree drawing algorithm [19] and implemented using d3.js and jQuery. The conventional way of drawing a grammar tree is to draw it vertically, with the root on top. In my Indented Level-base Tree drawing algorithm [72], the grammar tree is drawn horizontally from left to right. There are two benefits of drawing the tree horizontally. First, we read English texts horizontally from left to right. It's mentally easier for readers to switch between reading the text and reading the grammar tree that is displayed horizontally. Second, a horizontal sentence grammar tree fits the wide 16:9 aspect ratio of current computer displays. In my program, viewers can choose to use the traditional, vertical level-based tree drawing algorithm or my horizontal, indented level-based tree drawing algorithm.

The sentence tree visualization is highly interactive. Users can expand or fold any node on the tree and the tree will be automatically re-drawn. This allows users to simplify certain part of the grammar tree and focus on the other parts, thus working on multiple levels of detail. This kind of interactive grammar tree visualization is an innovation in text analysis and is not found in any existing corpus tool.

PART 4

AN INDENTED LEVEL-BASED TREE DRAWING ALGORITHM

4.1 Defining Indented Level-Based Tree Drawing Problem

The indented level-based tree drawing (see Fig. 4.1 for example) algorithm has the following properties:

1. A left child and a right child should be positioned on the right of their parent node.
2. The root node is the leftmost node of the tree. The tree is displayed horizontally.
3. The horizontal coordinates of the leaf nodes should be indented in such a way that, starting with the top leaf node, the leaf nodes are sorted horizontally from left to right. In other words, the leaf nodes can be read from left to right as a sentence.

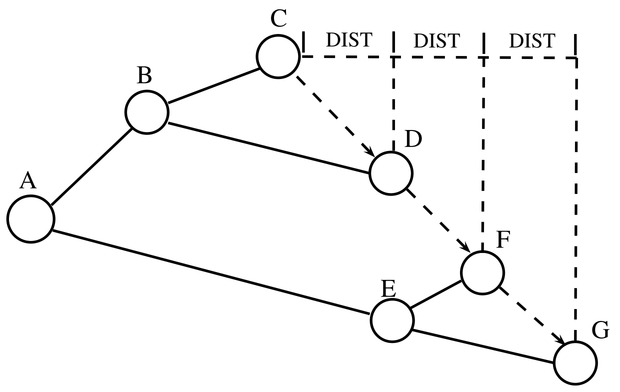


Figure (4.1) An example of the indented tree drawing. The leaf nodes are sorted from left to right horizontally.

The indented tree drawing, due to its third property, will first sort leaf nodes horizontally from left to right, and then vertically from top to bottom. Therefore the tree drawing problem is then reformulated as follows: given an input tree structure, calculate the horizontal and vertical coordinates for each node of the tree so that the drawing is compact and satisfies the properties as listed above.

4.2 Reingold and Tilford Algorithm

Reingold and Tilford Algorithm is one of the most important and influential algorithms in the area of tree drawings, and serves as the basis for my tree drawing algorithm. In this section, I discuss its basic ideas and the major challenges it has addressed.

Reingold and Tilford Algorithm is inspired by Wetherell and Shannon algorithm [17]. Wetherell and Shannon algorithm assigns equivalent vertical coordinates to nodes at the same level, and keeps track of the next leftmost available horizontal positions on each level when positioning nodes in an attempt to keep the width of the tree minimal. It works well in many cases, but can cause unpleasing result in some cases as the width between two sibling nodes is unnecessarily expanded. Reingold and Tilford proposed a new aesthetic (Aesthetic rule #4), stating that a subtree should be drawn the same way regardless of where it occurs in the tree. They designed a new algorithm to incorporate this new aesthetic. Instead of statically assigning the leftmost available position to nodes, the algorithm recursively computes the horizontal coordinates of the nodes and dynamically adjusts their positions. The brief description of the algorithm is as follows:

Two tree traversals are used to produce the final horizontal coordinates of nodes while their vertical coordinates can be pre-determined with their levels. The first post-order traversal assign the preliminary horizontal coordinates and modifier fields for each node. The second pre-order traversal compute the final horizontal coordinates for each node by summing its preliminary horizontal coordinates and modifier fields of all of its ancestors. In the post-order tree traversal, starting from leaf nodes (the smallest subtrees), smaller subtrees are positioned from left to right, and are combined with their parents to form greater subtrees. Parent nodes are placed in the center of their children. This process continues recursively until the root is reached.

The vital part of the algorithm is how to position sibling subtrees. For a given node, its

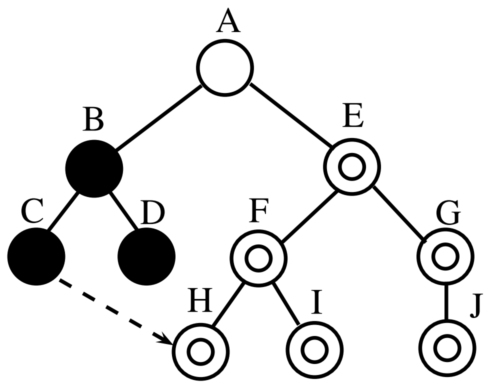


Figure (4.2) Position subtree E on the right of subtree B. A traversal of right contour of subtree B and left contour of subtree E is needed to compute the proper distance to separate subtree B and subtree E.

subtrees are positioned one by one, from left to right. When positioning a new child subtree, the subtree is shifted right until the pre-defined distance between the new subtree and its left sibling subtree (or sub-forest) is reached. The process starts at the level of subtree root. The subtree is pushed towards right so that the roots of the subtree and its sibling subtree are separated by the *sibling separation* value. At the next lower level, the subtree is pushed towards right again if the leftmost node of the subtree and the rightmost node of its sibling subtree at that level is not separated by the *subtree separation* value. The process continues until the bottom of the shorted subtree is reached. The shifting distance of the subtree is stored in the modifier field of the subtree root. In order to run the above steps in $O(n)$, Reingold and Tilford introduced the concept of "contours". The left (right) contour of a subtree is defined as the sequence of leftmost (rightmost) nodes at each level in the subtree. A linked list data structure is used to maintain the contour. I demonstrate these concepts with an example in Fig. 4.2. For simplicity, I refer subtree A as the subtree rooted at node A. Subtree A consists of subtree B and subtree E. The left contour of subtree B contains node B and node C and its right contour contain node B and node D. Likewise, Nodes E, F, H and nodes E, G, J compose the left and right contour of the subtree E, respectively. When positioning subtree E, node pairs (B, E) and (D, F) are examined at different level, and subtree E will be shifted accordingly to maintain the pre-defined distance between nodes in node pairs.

The construction of contours is recursively performed while positioning subtrees. The parent subtree can form its left/right contour by linking its root and the left/right contour of its child subtrees. For example, the left contour of subtree A contain node A, the left contour of subtree B, and node H which is on the left contour of subtree E.

The second tree traversal, a preorder traversal, determines the final horizontal coordinates for nodes. It starts at the root of the tree, summing each node's preliminary horizontal coordinate value with the combined sum of the modifier fields of its ancestors.

4.3 Indented Tree Drawing Algorithm

My tree drawing algorithm is based on Reingold and Tilford algorithm [19] but with significant change, mainly because of the third property listed in Section 4.1. Namely the leaf nodes need to be sorted horizontally. This property is in conflict with the Aesthetic rule #1 in the Reingold and Tilford algorithm. The underlying assumption in Reingold and Tilford algorithm is that the vertical coordinates of each node is determined by its level. In my case, this is no longer valid. Therefore, my algorithm needs to compute both horizontal and vertical coordinates recursively.

Here I still use a vertical tree for easier explanation. In the implementation, the tree is draw horizontally.

The algorithm requires three traversals of a tree. The first traversal is described in *firstWalk* procedure (See Algorithm 1). Starting from the leaves that I consider the minimal subtrees till the root of the whole tree, the algorithm recursively computes and stores the relative positions of the subtree root relative to their children (if they have any) and the relative positions of the subtrees relative to their sibling subtrees, respectively. The second tree traversal aggregates these relative positions, and compute the final position for each node. This process is described in *secondWalk* procedure (See Algorithm 2). The last tree traversal redistributes the vertical positions of nodes in order to make it look more pleasant while still maintaining the properties of the indented tree drawing (See Algorithm 3).

Algorithm 1 firstWalk procedure

```
 1: procedure FIRSTWALK(TreeNode $v$)
 2:     TreeNode $w$                                              ▷ $v$'s left sibling
 3:     List<TreeNode> $nodes$                                    ▷ child of $v$
 4:     int $mid$                                                 ▷ center of child

 5:     int $len \leftarrow nodes$.length
        ▷ Line 6-11 horizontal coordinates
 6:     if $v$ is a non-leaf node then
 7:         mid=($nodes$[0].prelimX+$nodes$[$len$-1].prelimX)/2
 8:         $v$.prelimX = mid;
 9:     else if $w$ exists then
10:         $v$.prelimX=$w$.prelimX+SS                                ▷ SS: const
11:     end if
        ▷ Line 12-19 vertical coordinates
12:     if $v$ is a non-leaf node then
13:         $v$.prelimY=$nodeY$.prelimY+$nodeY$.modY-DIST;
            ▷ DIST: const
            ▷ $nodeY$ is $v$'s child with the smallest vertical coordinates
14:         if $w$ exists then
15:             $v$.modY=$w$.modY+DIST*$w$.l;
                ▷ $w$.l denotes the number of its offspring leaves
16:         end if
17:     else if $w$ exists then
18:         v.prelimY=w.prelimY+DIST;
19:     end if
20:     if $w$ exists then
21:         Positioning $v$ on the right of $w$
22:     end if
23: end procedure
```

Like other positioning algorithms, my algorithm also uses two separate fields for the positioning of tree nodes. For a non-leaf leave, the *prelimX* or *prelimY* field of the node denotes its relative horizontal or vertical position to its children while the *modX* or *modY* field denotes the relative horizontal or vertical position that the subtree root node is from its sibling subtree. The position value in the *modX* or *modY* field of a node is assigned based on the entire subtree rooted at the node, and will be applied to all of its offspring nodes when calculating their final coordinates. For a leaf node, only the *prelimX* or *prelimY* is

Algorithm 2 secondWalk procedure

```
1: procedure secondWalk(TreeNode $v$)
2:     v.modX += v.parent.modX;
3:     v.x = v.prelimX + v.modX;
       ▷ Line 2-3 Compute final horizontal coordinates
4:     v.modY += v.parent.modY;
5:     v.y = v.prelimY + v.modY;
       ▷ Line 4-5 Compute final vertical coordinates
6: end procedure
```

needed to denote its relative horizontal or vertical position to its leftmost siblings. I assume in this paper that the coordinate system has its original point at the top-left corner. That is, if the height of a node is greater than that of another node, then the node has a smaller vertical coordinate.

In order to decrease the complexity of computation, the algorithm is designed to decouple the horizontal and vertical positioning of nodes. There are three major steps that will change the positions of the nodes. The first one is the initial assignment of both horizontal and vertical positions to nodes in accordance to the rules in the first tree traversal. The next step is the subtree positioning, which adjusts the horizontal positions of nodes. The third step is the node redistribution step in the third tree traversal, which tunes the vertical positions of nodes. I discuss these three steps in more details in the coming sub-sections.

4.3.1 Assigning initial positions to nodes

I have defined rules for the assignment of horizontal and vertical positions. For horizontal positioning, the rules state that (1) if a node is a non-leaf node, then place it in the center of its children; (2) if a node is a leaf with no sibling, assign 0 to *prelimX* field. (3) if a node is a leaf with a left sibling, then place it to the right of its left sibling at a pre-defined distance. The pseudo codes for these rules are listed in line 6-11 in Algorithm 1.

The rules for assigning vertical positions to nodes are more complicated:

Rule 1: If a node is a leaf with no left sibling, assigns 0 to its *prelimY* field.

Rule 2: If a node is a leaf with a left sibling, in order to satisfy the third property of indented tree drawings, the algorithm assigns w.prelimY+$DIST$ to the node's *prelimY* field where w is the node's left sibling and $DIST$ is the pre-defined distance.

Rule 3: If a node is a non-leaf node with no left sibling, in order to guarantee that a node is higher than any of its children, the algorithm assigns *nodeY*.prelimY+*nodeY*.modY-$DIST$ to the node's *prelimY* field where *nodeY* is the child which has the greatest height.

Rule 4: If a node is a non-leaf node with left siblings, not only should this node be higher than any of its children, but the entire subtree rooted on this node is also moved upwards by $DIST$*w.l where w.l represents the number of leaf node in its sibling subtrees. This is so because in order to satisfy the third property of indented tree drawings, the leftmost leaf of the current subtree should be $DIST$*w.l higher than the leftmost leaf of its sibling subtree.

The pseudo codes for the above rules are listed in line 12-19 in Algorithm 1.

4.3.2 Positioning subtrees

The horizontal positioning discussed above only calculates the relative position of a node to its siblings or children. The task of positioning subtree aims at computing the relative position of the current subtree to its sibling subtree (Line 21 in Algorithm 1). Positioning a subtree on the right of its sibling subtree in the level-based tree drawing is a complicated process since the algorithm has to travel the sequence of right-most nodes in the left subtree and the sequence of leftmost nodes in the right subtree in order to determine the minimal shifting distance that can separate two subtrees at a pre-defined distance. (I have discussed this process in Section 4.2.) However, positioning subtrees in my tree drawing algorithm is greatly simplified and can be achieved in $O(1)$ time.

For a given subtree, I argue that its leftmost node is its leftmost leaf. The proof is as follows: Assume the above statement is not correct. In other words, the leftmost node of a subtree is a non-leaf node. Since this is a non-leaf node, its offsprings contain at least one leaf node, and according to the second property of my tree drawing algorithm it is centered among its children. If the non-leaf node has only one leaf node, then it has the same horizontal coordinates as its leaf node. This can prove that the leftmost of a subtree is a leaf node. Otherwise, it must have a leaf node that is positioned on its left. This is a contradiction. So I prove that for any given subtree, its leftmost node is its leftmost leaf. With similar method, I can also prove that a subtree's rightmost node is its rightmost leaf.

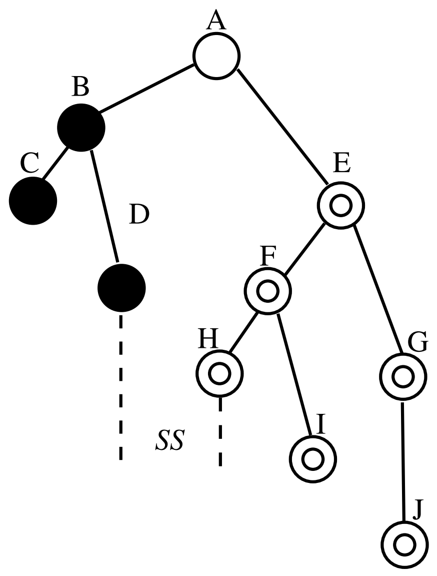


Figure (4.3) Node D is the rightmost node in subtree B. Node H is the leftmost node in subtree E. The task of positioning subtree E is equivalent to separating Node D and Node H horizontally at a distance of SS (a pre-defined constant).

This property makes the task of positioning subtrees quite simple. Traveling the left-/right contour of right/left subtrees is unnecessary because if the rightmost leaf in the left subtree and the leftmost leaf in the right subtree are properly separated, then other nodes in these two subtrees will be well separated. In Fig. 4.3, for example, if node D and node H are well separated, then nodes within subtree B and nodes within subtree E are also well separated. The distance that a subtree is moved is equivalent to the distance that the subtree's leftmost leaf is moved so that the leaf is properly separated from the rightmost leaf of its left sibling subtree.

4.3.3 Redistributing nodes

In my algorithm, left sibling subtrees are gradually lifted up in order to line up the leaf nodes from left to right. In most cases this would not cause problems. However, in some cases where the left subtree contains many more nodes than the right subtree, the resulting tree drawing may look less pleasant since nodes within the right subtree are not vertically distributed evenly. In Fig. 4.4, the tree drawing on the left does not look good as node A and node C is unnecessarily distant. The tree drawing on the right looks more pleasant after node C is repositioned between node A and node D.

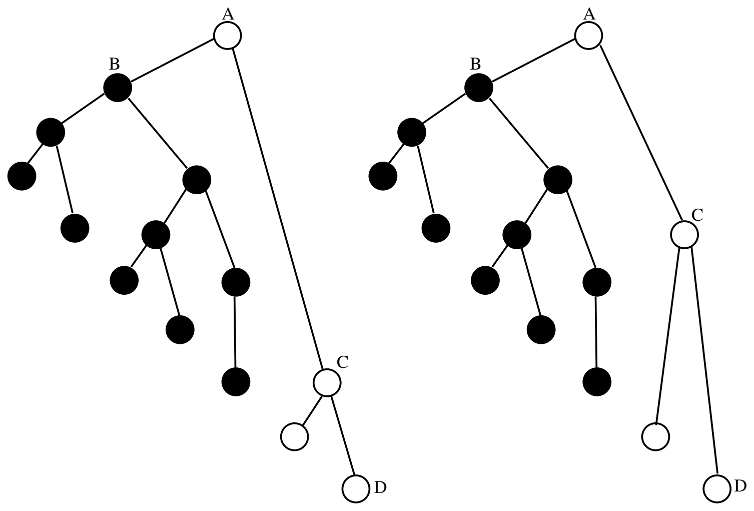


Figure (4.4) The left tree drawing is less pleasant. The right tree drawing is more pleasant after node C is repositioned.

The task of node redistribution is to carry out another tree traversal and redistribute the vertical positions of nodes recursively so that the nodes within subtrees are vertically more evenly distributed.

To implement the node redistribution, the algorithm requires another pre-order tree traversal. For any visited node, the algorithm recalculates the vertical coordinate for each of its children. The calculation is based on the idea that nodes at different levels within the visited node's subtree should be vertically and evenly positioned between the visited node and the leaf. As described in Algorithm 3, the *thirdWalk* procedure computes the allocatable space between the node and its child, evenly splitting the space among different levels of the subtree rooted on the child. It then determines the new vertical coordinate of the child node.

Algorithm 3 thirdWalk procedure

```
1: procedure THIRDWALK(TreeNode $v$)
2:     for each child $u$ of $v$ do
3:         $spots \leftarrow (u.\mathrm{y}-v.\mathrm{y})/\mathrm{DIST}-1$;
           ▷ $spots$: # of vertical positions between $v$ and $u$.
4:         $level \leftarrow u.\mathrm{level}$
           ▷ for a leaf node, its level is 1
           ▷ for a non-leaf node, its level is 1 plus the max of its child's level
5:         $ave \leftarrow spots/level$
           ▷ $ave$: # of vertical positions $u$ will be pulled toward its parent.
6:         $u.\mathrm{y} \leftarrow u.\mathrm{y}-ave*\mathrm{DIST}$
7:     end for
8: end procedure
```

4.4 Algorithm Analysis

My algorithm is composed of three tree traversals. In the first post-order tree traversal, for each node, the initial assignment of coordinates takes $O(1)$ time, and the associated subtree positioning takes $O(1)$ time. So the total time complexity for the first tree traversal is $O(n)$. The second pre-order tree traversal takes constant time for each node to sum up the coordinates of its parent node. Therefore, its total time complexity is $O(n)$. The time complexity of the last tree pre-order tree traversal is also $O(n)$. I conclude the time complexity for the algorithm is $O(n)$.

4.5 Implementation and Usage

I have implemented the indented level-based tree drawing algorithm in JavaScript and integrated it into D3.js [73], a popular JavaScript library for data visualization. The *Layouts* package in D3.js library provides efficient implementation of layout algorithms for various structures including the classic level-based tree. It also offers helper functions to facilitate the implementation of new layout algorithms.

Based on D3.js, I have added new APIs for quick construction of the indented tree drawing. I explain some of the APIs in Table 4.1:

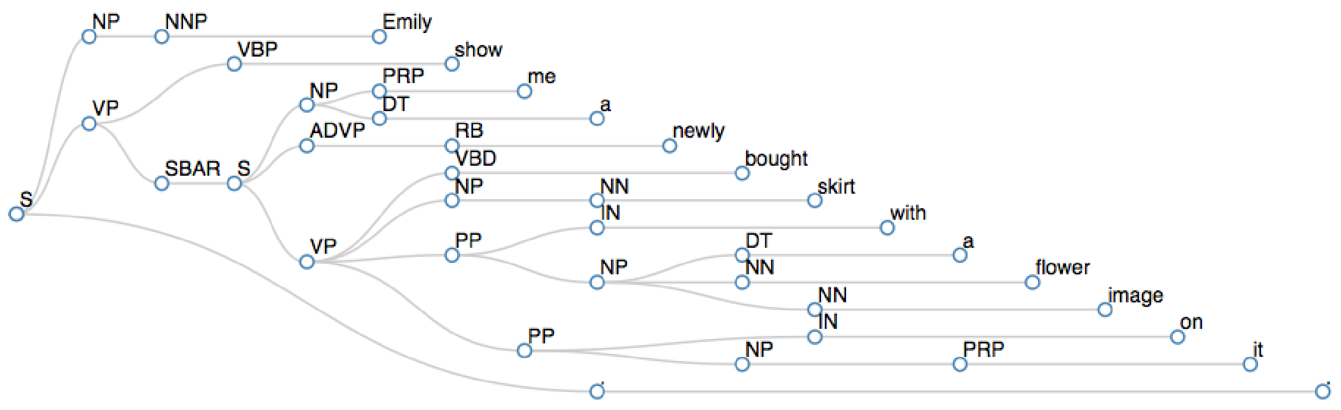


Figure (4.5) The indented level-based parsed tree of the sentence “Emily show me a newly bought skirt with a blue flower image on it.”

APIs	***Description***
d3.layout.indentedtree	Creates a new indented tree layout.
indentedtree.size	Sets the available layout size.
indentedtree.sort	Sets the sort order of sibling nodes.
indentedtree.separation	Sets separation between nodes.
indentedtree.nodes	Computes the tree layout.
indentedtree.links	Returns edge positions.

Table (4.1) APIs for the indented tree drawings

With these APIs, Web application developers only need to write a few lines to draw indented trees and embed them in their Web pages. The code snippet in Listing 1 provides a showcase for drawing a simple indented tree.

Line 2 creates an instance of indented tree. Then in Line 3 the size of the tree drawing is specified. The positioning algorithm is invoked in Line 6, and tree nodes’ horizontal and vertical coordinates are returned. The positions of tree edges are computed by pairing parent nodes and child nodes in Line 8. Once the position of tree nodes and edges are specified, then other D3 drawing routines can be used to draw the indented tree. Fig. 4.5 shows an example of the indented level-based tree drawing created with my new APIs. Comparing that with the traditional parsing tree in Fig. 1.5, we can see that my drawing has a wider aspect ratio that fits better with newer computer displays. It also places the words from left to right for easy reading, while clearly presenting the syntactic structure of the sentence.

```
// Create a tree layout
var tree = d3.layout.indentedtree();
tree.size([height, width]);
// Compute nodes' positions
// root: input data
var nodes = tree.nodes(root);
// Retrieve edges positions
var edges = tree.links(nodes);
Draw nodes and edges with D3 routines.
```

Listing 4.1: Sample code for drawing an indented tree

PART 5

INTEGRATED SPATIO-TEMPORAL STORYLINE VISUALIZATION

5.1 Overview

The four basic elements: time, location, characters and context, and their combinations are essential for storytelling. Although it is interesting that these story elements are presented in various forms in most of current storytelling visualizations, I believe that a large portion of their work is duplicated and can be reduced with the functionality provided by a visual storytelling platform. Also, the creation of storytelling visualization required solid techniques from computer science, and may not be acquired by people from journalism or other background. For example, the processing of large data set can be a great challenge for people with little knowledge of computer science. So one of the important design goals of my visual storytelling engine is to provide a platform for visualizing large spatial-temporal data sets and presenting these story elements to audiences in a uniform and narrative way.

Fig. 5.1 show the architecture of my visual storytelling engine. The data processing component can retrieve large raw data set from various data sources, clean them and transform them into the uniform format that the visual storytelling engine can readily visualize. Depending on the input data, the potential combinations of story elements include (1) time plus characters plus context (2) location plus characters plus context (3) time plus location plus characters plus context. The visual storytelling engine prepares three visualization paradigms for these different story element combinations. The traditional timeline can be used to visualize data sets with time and characters while the traditional interactive map is quite suitable for visualizing data sets with location and characters. As to data sets with time, location and characters, I combine an interactive map and a storygraph, and create a new visualization paradigm for such data sets. The motivation comes from the observation that traditional interactive maps can provide an excellent view for spatial data sets, but

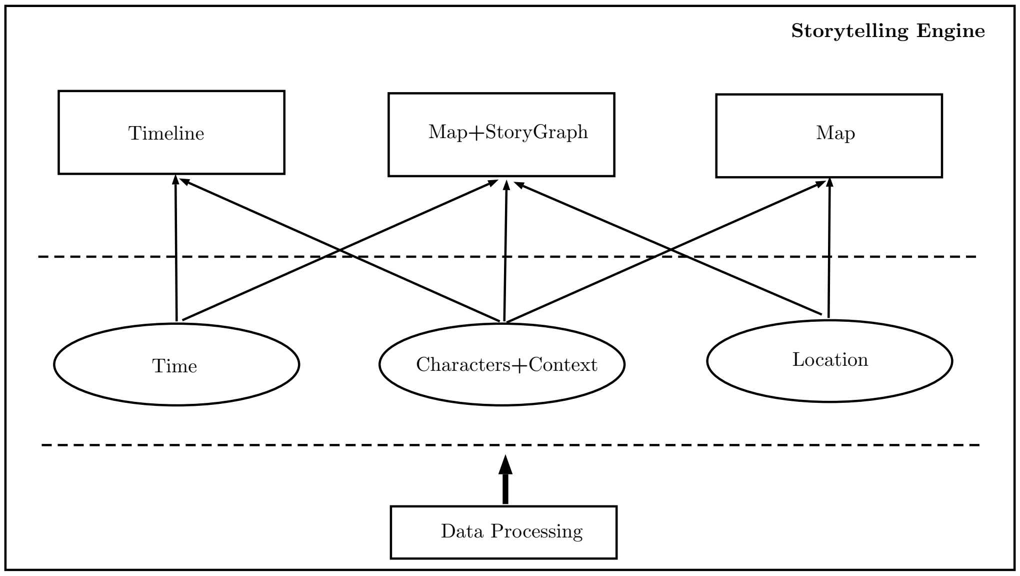


Figure (5.1) The Architecture of the Visual Storytelling Engine

they lack the ability to present the temporal information of events. On the other hand, storygraphs are designed to explore the events at specific locations, and can be considered a useful complement of the interactive map for visualizing spatial-temporal data sets. Context visualization is another feature I would like to add to the visual storytelling engine. The storytelling engine can identify the events that are geographically close to a specific event or the events that happen in a given time frame.

In the following section, I will focus on describing the interactive map and storygraphs, and how they are combined to visualize spatial-temporal data sets.

5.2 Interactive Map

An interactive map is a diagrammatic representation of an area of land or sea showing physical features, cities, roads, etc, and allows audience interaction. It is a natural choice for visualizing spatial data sets. I use Google Map as the interactive map implementation in the visual storytelling engine, and enhance it with new functionalities as follows:

- Integrate with Google SpreadSheet and allow audiences to input and visualize their data sets.

- Compute the center point of the input data sets, and load Google Map and make it center on the point.
- Allow zooming of selected region in Google Map.
- Allow the creation of a storygraph based on the selected region.

5.3 Storygraph

A storygraph is an innovative visualization paradigm I developed for spatio-temporal data, and is applied in the storytelling engine. It consists of two parallel vertical axes similar to the vertical axes in the parallel coordinates, along with a horizontal axis. The vertical axes represent the location pair e.g. latitude and longitude while horizontal axis represents time. Each location (x,y) or (latitude, longitude) is represented as a line segment referred to as location line. Location lines are created by joining the corresponding location values on the two vertical axes. Events occurring at the same location on different times are plotted as markers along the location line. Figure 5.2 is an example of a storygraph. Circles on the line represent the events that happen at the same place at different times.

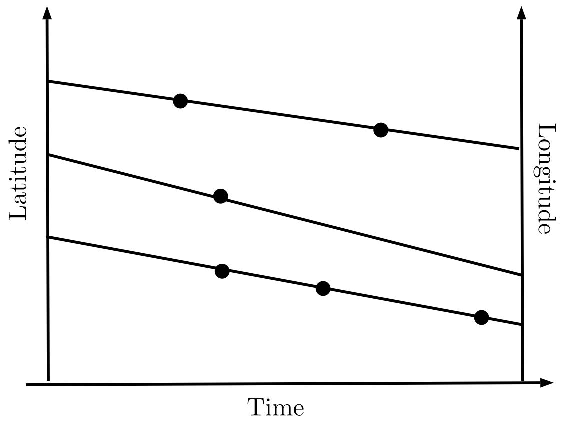


Figure (5.2) A storygraph example

The basic idea behind the storygraph is to visualize spatio-temporal or 3D data in a

2D coordinate system. One of the major challenges in storygraph construction and other data visualization practices is the visual clutter created due to large data set plotted on the display with limited area [74]. The visual clutter is problematic since it can impair the ability of data visualization to convert information. I divide the visual clutter problems into two subproblem: point clutter problem and line crossing problem, and elaborate them and corresponding solutions in the following sections.

5.3.1 Point Clutter

Point clutter in a storygraph results from placing much more points on vertical axes than the number of pixels the vertical axes contain in the display. For most of the displays on laptops or PCs, the number of pixels on vertical axes is about a thousand. Even a median size of data set when plotted can make the axes quite crowded. This can cause two issues. First, points on the axes become barely visible and distinguishable. Second, story lines connecting these points twist each other. The problems can become worser if points are permuted on axes in order to address line crossing problems. As a result, a preprocessing on the data set is necessary. The basic idea is to merge the neighboring points into one single point so that the issues I discussed are mitigated. Steps are as follows:

- Identify the max and min point value in the data set.
- Retrieve the number of pixels on the axis.
- According to range of point value and the number of pixels, evenly partition the range of point value into subranges.
- Points that fall in the same subrange of point value will be plotted to the same pixel.

5.3.2 Line Crossings

Location line crossing problem is another form of visual clutter in storygraphs. As I describe in the previous section, every location line in a storygraph stands for a unique geographic location and markers on the same location line represent events that happen at the same location but at different time. Yet line crossings in a storygraph do not provide any useful proximity information, and sometimes can be problematic since they make the storygraph less recognizable and create confusion for readers. Given that the latitude and longitude points are plotted on two parallel vertical axes in the ascending order, the number of line crossings in a storygraph is equal to the number of inversion pairs in a latitude/longitude coordinate (See Fig. 5.3). For a pair of 2D points a(x1, y1) and b(x2, y2), a and b are an inversion pair if and only if $(x1 > x2)$ XOR $(y1 > y2)$ is false.

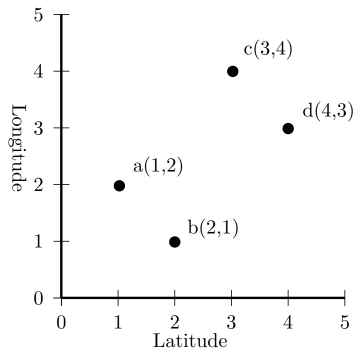


Figure (5.3) A latitude/longitude coordinate. (a, b) is an inversion pair. (c, d) is another inversion pair.

Another way to look at the line crossing problem is to model a storygraph into a bipartite graph. A bipartite graph can be defined as $G = (V, W, E)$, where V is the first disjoint node set on one vertical axis, W is the second disjoint node set on another vertical axis and E is the edge set connecting nodes in V to nodes in W. If we consider the latitude points to be V and longitude points to be W, then location lines constitute E and line crossing problem in a storygraph is equivalent to the edge crossing problem in a bipartite graph. The basic idea for minimizing line crossing is to permute points on one axis (level permutation problem or LPP) or both axes (bipartite drawing problem or BDP). Although both LPP and BDP are NP-hard in a general bipartite graph, these problems can be simplified in the storygraph context due to two properties of latitude/longitude points. First, the original orders of two point sets in a storygraph are numerically increasing order. Second, the identical latitude/longitude points are sparse. Based on these properties, my heuristics for minimizing line crossings is as follows:

The order of points on one axis is fixed. The points on the other axis are arranged in the same order as their adjacent points.

Suppose that every longitude point in location lines is unique, then with this heuristic I can eliminate line crossings in $O(e)$ time where e is the number of lines. The detailed steps are straightforward: for each location line, locate its latitude point in the numerically ordered axis and plot the location line horizontally. After all of the location lines are plotted, the points on the latitude axis are sorted and the points on the longitude axis are permuted. Since each location line is parallel to each other, the number of line crossings is zero. If some identical longitude points do exist in the data set, then there is no polynomial algorithm that can guarantee minimizing the number of line crossings. But the heuristics I discuss above is still useful. For each occurrence of identical longitude points, the number of line crossings it causes is the number of location lines between current location line and the location line in which that longitude points firstly occurs. In Fig. 5.4, location lines (x1, y1) and (x4, y1) share the same longitude point y1. The number of line crossings is equal to the number of location lines between (x1, y1) and (x4, y1).

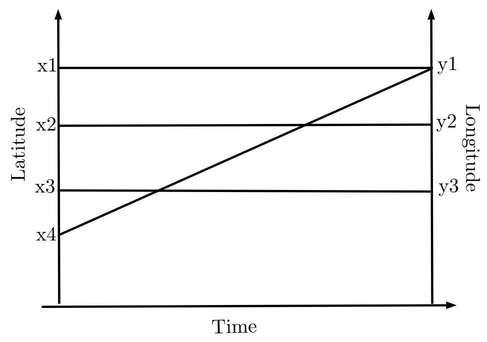


Figure (5.4) Location lines (x1, y1), (x2, y2), (x3, y3), (x4, y1). The number of line crossings are two.

5.3.3 Line Crossings after Point Merging

In the previous section, I mitigate the problem of point clutter by merging neighboring points. This preprocessing, however, induces extra issues to the line crossing problems. As neighboring points have merged into one point, a latitude point that originally only connect to a longitude point may become connected to multiple longitude point, making the assumption that identical latitude/longitude pairs are sparse in storygraphs and corresponding heuristics invalid.

I employ the modified median heuristics [75] as a solution. The heuristics is as follows:

The order of points on one axis is fixed. Horizontally align points on the other axis to the median points of their adjacent points.

I illustrate this heuristics with examples in Fig. 5.5 and Fig. 5.6. Fig. 5.5 is the storygraph that does not apply the heuristics. There are points x1∼x6 on the latitude axis, points y1∼y3 on longitude axis and seven location lines. All points are plotted in the numerically ascending order. The number of line crossings is ten. In Fig. 5.6, with the modified median heuristics, the position of x1∼x6 on the latitude axis remain unchanged, and y1∼y3 will be moved to the position on the longitude axis aligned to the median points of their adjacent points on the latitude axis. For example, y1 is adjacent to x3, x5 and x6, then y1 should be moved to position 5 on the longitude axis which is horizontally aligned to the position of x5 on the latitude axis. With the application of the modified median heuristics, the number of line crossings is reduced to two.

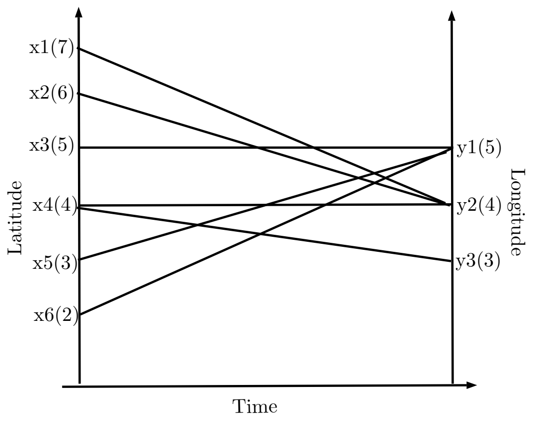


Figure (5.5) Layout of the storygraph without reduceing line crossings

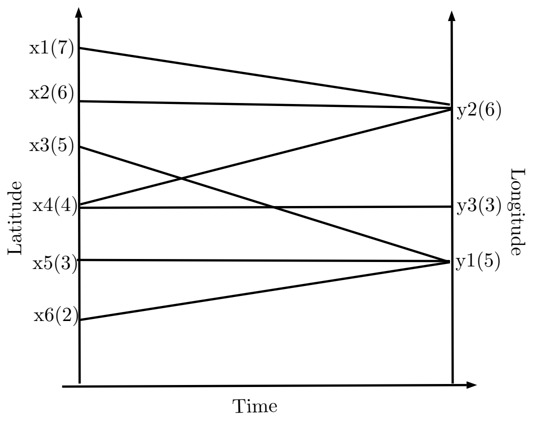


Figure (5.6) Layout of the storygraph after reduceing line crossings

PART 6

TENNIS MATCH VISUALIZATION

6.1 Overview

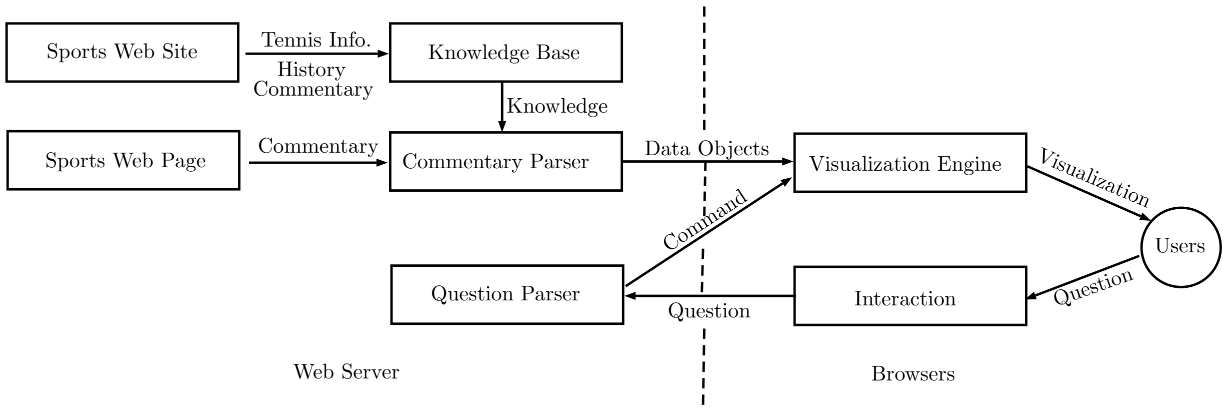


Figure (6.1) The Architecture of TennisMatchViz

TennisMatchViz is a tennis match visualization system that takes as the input a web page containing live commentary of a tennis match, parses it and presents the tennis match visually to audiences. As shown in Figure 6.1, TennisMatchViz is designed as a Web application and can be divided into two parts: server-side system and browser-side system. The server-side system establishes a tennis associated knowledge base by collecting information from sports websites. This knowledge base also includes history tennis commentaries which are crucial to the parsing of live tennis commentary. After this setup, the commentary parse is ready to accept a tennis web page, extract match associated information, parse comments, understand the match progress, and repackaging and transferring data objects describing the match to the visualization engine in the browser-side system. The visualization engine is responsible for displaying the tennis match to users. Its interaction with user is achieved with help of the interaction component. The interaction component accepts questions from users, have them parsed by the question parser in the server-side system which then issues commands to instruct the visualization engine to update the visualization accordingly.

The following sections will describe data processing and data visualization in more details.

6.2 Data Preparation

We obtain tennis match data primarily from two sources. On one hand, we crawl tennis match data from sports websites by analyzing the webpages that live blog the tennis matches, and reading match data directly from Web servers. On the other hand, we leverage point-by-point match data in the Match Charting Project [76]. These two data sources are inter-complemental since none of these data sources can provide complete set of tennis match data. The data from the first data source contains more subjective information, but is less formatted. The interpretation of such data depend on the accuracy of our algorithm. The data from the second data source comes from a group of tennis enthusiast charting on tennis match videos. These data is well formatted, and more accurate.

6.2.1 Crawl Tennis Match Data from Sports Websites

Live blogging of tennis matches on sports websites usually consists of a set of records. Each record contains three parts: a comment which expresses the opinion of the commentator towards a specific point, a current scoreboard (sets, games, points) and a timestamp. There can be multiple records for one single point. These records are easy for human beings to read and understand, and hard for computers to process. In order to effectively manage, understand and visualize this tennis match information, we need to transform the flat data into hierarchical data objects. Firstly, according to the tennis scoring system, a set of hierarchical data objects (match, set, game, point) are constructed. Then for each record, we parse the score board information it contains, determine which (set, game, point) this record is made for, and populate the data objects with the record. Below we will discuss some important detail for implementing this transformation.

- Not all the records contain useful information. Some of them contains commercial ads or other pictorial information. We need to preprocess these records, and remove unrelated records.
- The scoreboard changes only when a score is made. For the record that tracks the score, we can easily parse its scoreboard information and process it. But for other records made for the same point, their scoreboard information does not include the score, and is not correct. Our algorithm needs to identify these records, retrieve the comments in these records and store them in the proper data objects.
- Records may have different scoring rules and format for deuce points and points in tiebreak game.
- As records are input impromptu by commentators, typos are possible. The strategy we use to catch these typos and correct them if possible is based on the assumption that the order of records is in accordance with the progress of the match. If there is an abrupt change between the scoreboards of two successive records, it is likely that a typo is in the records. We can fix some of the typos, but lacks enough information for other typos. This is the limitation of our system.

Once we extract comments for each point, next step is to understand these commends and see what actions players have performed. The full understanding of arbitrary text is the ongoing research in the field of natural language understanding or computational semantics, and is traditionally considered to be a difficult problem. The current solution replies heavily on human-annotated text and sophisticated machine learning methods, but still can not achieve a satisfactory result. In our case, however, tennis comments text is a small subset of human language in terms of vocabulary and syntactic structure. With some tricks, we can grasp the key meaning out of them and provide data for tennis match visualization.

One important observation on the tennis comments is that many of sentences share the same structures. For example,

```
Murray fires a forehand over the baseline.
Federer fires a forehand over the baseline.
```

The above two sentences describe two different players making the same actions. These sentences are almost the same except that the subjects of the sentences are different. We have seen plenty of similar patterns in the tennis commentary and we believe that some sports websites use templates for their tennis live commentary. Based on this observation, we come up with an algorithm for training and understanding the tennis comments. The basic idea is to identify these patterns in the templates by computing common strings in the history comments and annotate them, and then use these annotated patterns to parse the new comments. The outlines of the algorithm is as follows:

1. Collect a certain number of tennis commentaries from the same sport website. Extract the comments from records in the commentaries.
2. Replace player names with the same string.
3. For each pair of comments, find the longest common string.
4. Count the frequency of longest common strings.
5. Sort these longest common strings according to their frequency.
6. Annotate these common strings with high frequency.
7. For a new comment, parse it by checking if it contains any annotated strings.

For completeness, we briefly discuss the algorithm for computing longest common string of two strings. Let us suppose we have two strings X and Y with a length of m and n. For any suffixes of X and Y, their longest common string has the following optimal substructure property:

$$LCS(i,j) = \begin{cases} 0 & X[i]! = Y[j] \\ LCS(i+1, j+1) + 1 & X[i] = Y[j] \end{cases}$$

Where i and j are the starting index of the suffixes in X and Y, respectively.

Therefore, we can create a $m * n$ table and use dynamic programming to compute the common string of any pair of suffixes of X and Y, and select the longest one.

6.2.2 Obtain tennis match data from a match charting project

Another way of obtaining tennis match data is to download the match data from the match charting project. The project is carried on by a group of tennis enthusiasts, and aims to provide more accurate tennis match data. These tennis match data are generated by tennis enthusiasts watching and charting on the video of such match. The project has predefined a set of signs to help describe shots in the match. Our parser program can understand these signs and thus understand how each shot is performed. Comparing to crawling data from sports websites, this method can provide more accurate data. However, The amount of tennis match data available in this project is limited.

6.3 Data Visualization

After the data processing step, processed tennis match data will be delivered to the browser and presented visually to audiences. The main goal of our visualization engine is to develop a general front-end framework for tennis match data visualization. We primarily classify the tennis match data into two categories: basic data and technical detail data. Basic data include the number of sets in the match, the number of games in each set, the scores in each games and the duration for each game and each set. Technical detail data contain information describing the tennis match from different perspectives. The following listing shows a subset of technical detail data that audiences are interested in.

- Serve (ace, fault, double fault)
- Ground strokes (forehand topspin, backhand topspin, forehand slice, backhand slice)
- Net game (volley, smash)

- Specialty shots (lob, dropshot, passing shot, half volley, swing volley)
- Errors (forced, unforced),
- Rally length.

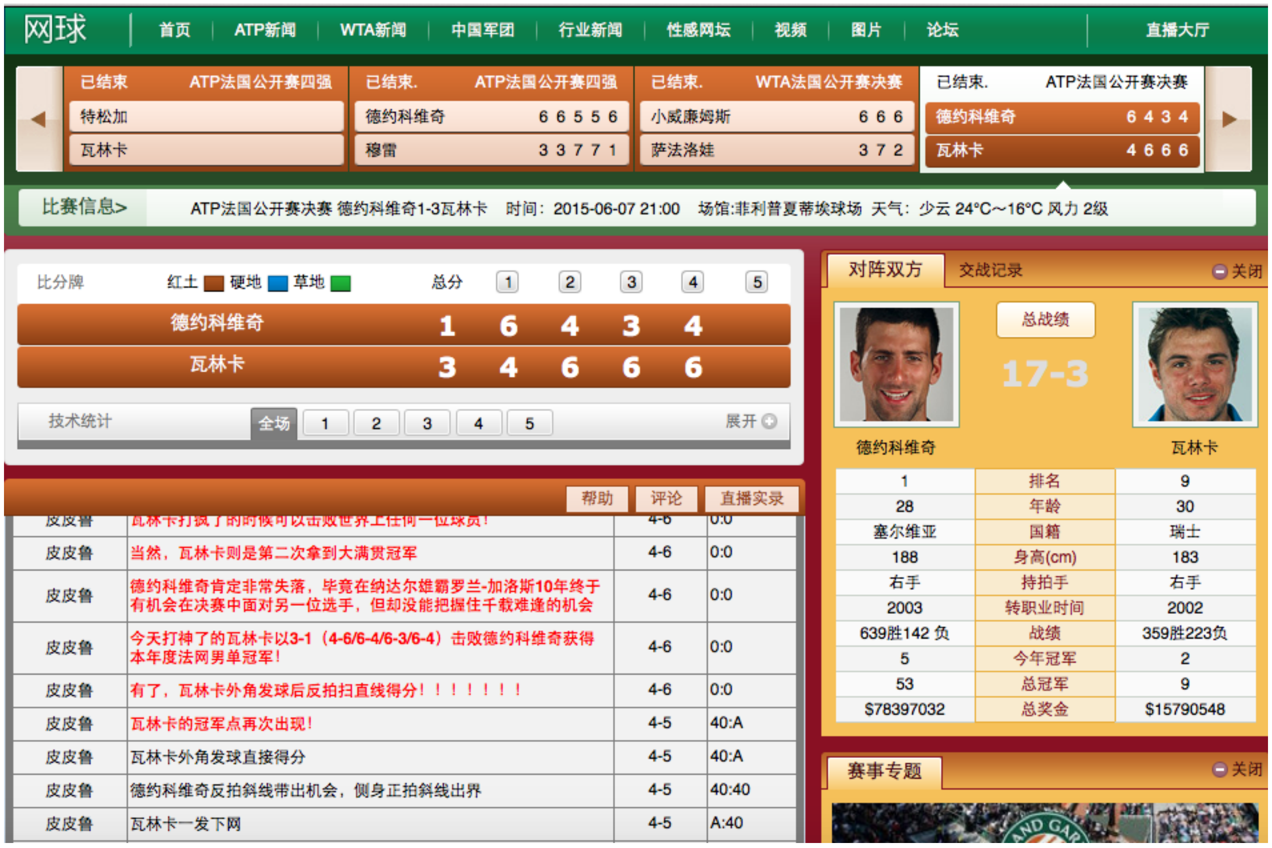
Basic data are visualized with a “non-traditional” table with one row denoting a set of tennis games. Each row is equipped with a Cartesian coordinate system, whose horizontal and vertical axis stand for the time of the games and the scores, respectively. A chart consisting of circles and lines is used to represent a game. Each player and his associated data are assigned a color. The colored circles in the chart represent the scores of players at every point. When the pointer hover over these circles, the associated comments will be popped up and displayed to audiences. Circles with same color are connected to form lines to show the trend of the game. These lines can provide a straightforward view to audiences on how the games are carried out. For example, in close games these lines will alternately grow. Charts are assigned background colors which are similar to that of players who win the games. The width of the chart is proportional to the duration of the game, and offers another aspect of game information to audiences.

6.4 Interaction

In comparison, technical detail data is harder to be visualized and shown to audiences. First of all, we can not show all the technical detail data to audiences in one visualization. They contain too much information and can easily confuse audiences. Secondly, in many cases, the audiences do not wish to understand the whole data set. They may want to filter these data, and retrieve what interest them most. For example, they may just want to see the double faults that one player performs in a certain set. Thirdly, it is more meaningful that the selected technical detail data can be visualized alone with the basic data.

We come up with the concept of “visualization on demand”. The idea is similar to that of Question-Answering in the field of Natural Language Processing. Nowadays, many famous products like Apple Siri, Google Now, Microsoft Cortana are based on Question-Answering technology, and are able to communicate with human being in a more user-friendly way. They take a user question as input, parse it, carry out the semantic search based on the parsed query and output results to users. Visualization on demand operates in the similar way, but instead of conducing semantic search, it produces a visualization that meets users' need.

The visualization on demand is implemented with the collaboration of the visualization engine, the interaction component and question parser. The interaction component can accept questions from users. These questions are parsed, and commands are produced and issued to the visualization engine. The visualization engine reads the commands and selects the proper technical detail data to update the visualization.



已结束	ATP法国公开赛四强
特松加	
瓦林卡	

已结束.	ATP法国公开赛四强
德约科维奇	6 6 5 5 6
穆雷	3 3 7 7 1

已结束.	WTA法国公开赛决赛
小威廉姆斯	6 6 6
萨法洛娃	3 7 2

已结束.	ATP法国公开赛决赛
德约科维奇	6 4 3 4
瓦林卡	4 6 6 6

比赛信息> ATP法国公开赛决赛 德约科维奇1-3瓦林卡 时间：2015-06-07 21:00 场馆:非利普夏蒂埃球场 天气：少云 24°C~16°C 风力 2级

	总分	1	2	3	4	5
德约科维奇	1	6	4	3	4	
瓦林卡	3	4	6	6	6	

皮皮鲁	瓦林卡打疯了的时候可以击败世界上任何一位球员！	4-6	0:0
皮皮鲁	当然，瓦林卡则是第二次拿到大满贯冠军	4-6	0:0
皮皮鲁	德约科维奇肯定非常失落，毕竟在纳达尔雄霸罗兰-加洛斯10年终于有机会在决赛中面对另一位选手，但却没能把握住千载难逢的机会	4-6	0:0
皮皮鲁	今天打神了的瓦林卡以3-1（4-6/6-4/6-3/6-4）击败德约科维奇获得本年度法网男单冠军！	4-6	0:0
皮皮鲁	有了，瓦林卡外角发球后反拍扫直线得分！！！！！！！	4-6	0:0
皮皮鲁	瓦林卡的冠军点再次出现！	4-5	40:A
皮皮鲁	瓦林卡外角发球直接得分	4-5	40:A
皮皮鲁	德约科维奇反拍斜线带出机会，侧身正拍斜线出界	4-5	40:40
皮皮鲁	瓦林卡一发下网	4-5	A:40

德约科维奇		瓦林卡
1	排名	9
28	年龄	30
塞尔维亚	国籍	瑞士
188	身高(cm)	183
右手	持拍手	右手
2003	转职业时间	2002
639胜142 负	战绩	359胜223负
5	今年冠军	2
53	总冠军	9
$78397032	总奖金	$15790548

Figure (6.2) The Original Web Page of the Tennis Match Between Novak Djokovic and Stan Wawrinka (in Chinese)

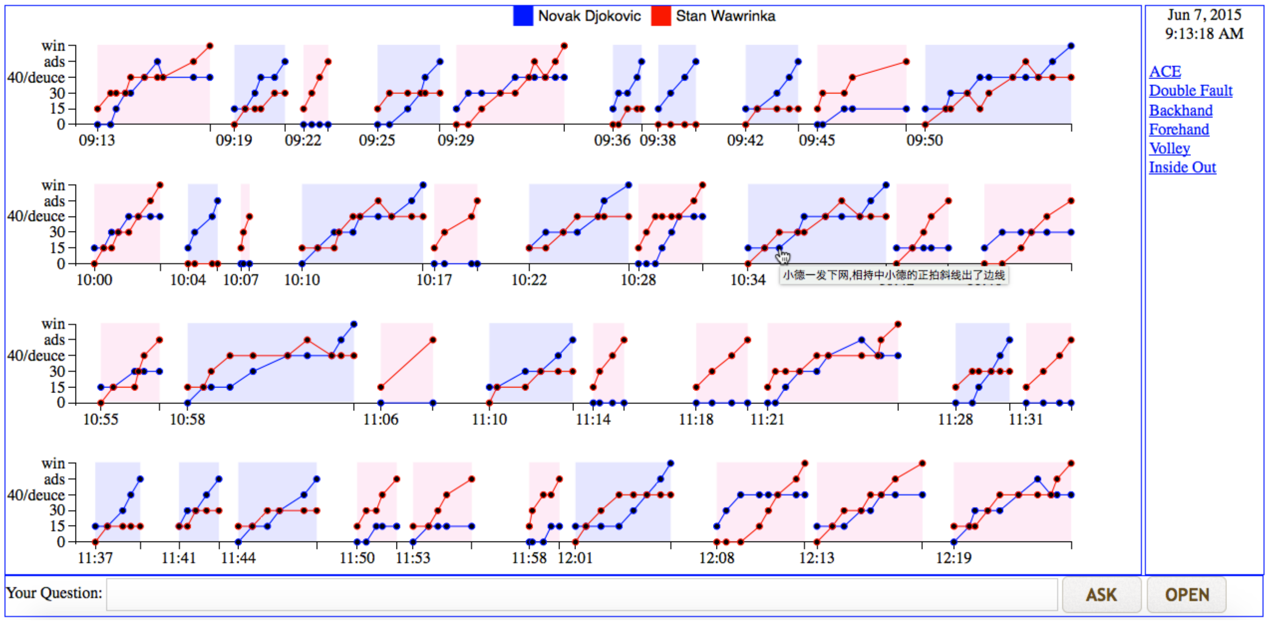


Figure (6.3) The TennisMatchViz Visualization for the Tennis Match Between Novak Djokovic and Stan Wawrinka

6.5 Implementation

TennisMatchViz is based on the browser-server architecture and can be divided into a sever-side system and a browser-side system. The server-side system is implemented with Java Servlet and Google Json library and is running in the Tomcat Web server. The browser-side system is implemented with Javascript and Javascript library such as D3.js, JQuery and JQuery UI.

6.6 Case Study

To demonstrate the use of our system, we will visualize the man single finals between Novak Djokovic and Stan Wawrinka in ATP France Open 2015. The live commentary is from a famous Chinese sports website sports.sina.com.cn. One reason for collecting data from Sina Sports is that it has been providing tennis match live commentary service for a long time, and have hundreds of tennis matches data available to the public. This is convenient and

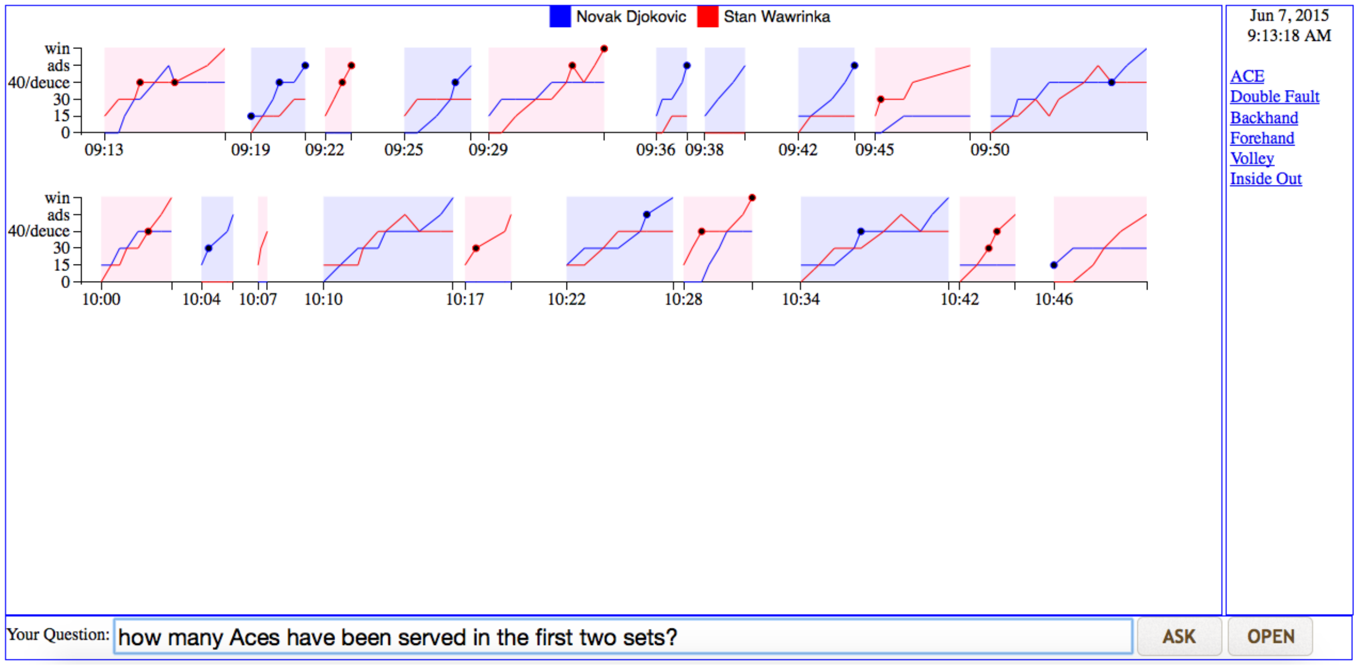
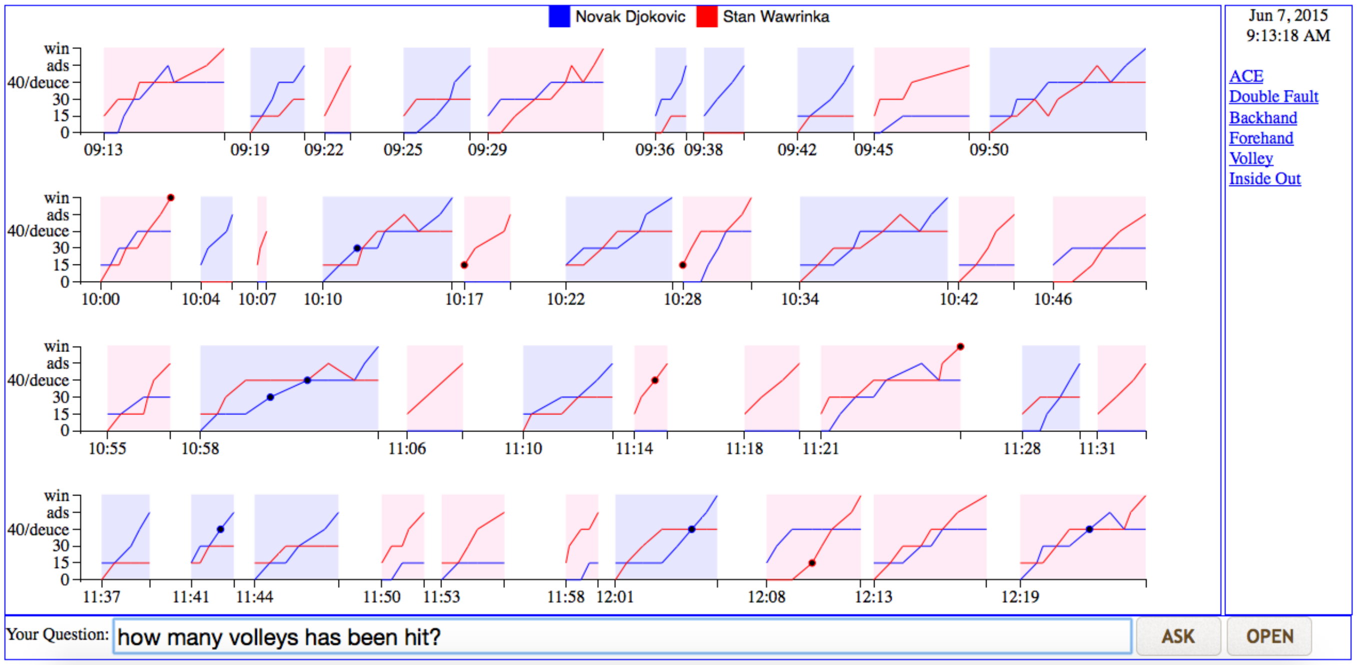


Figure (6.4) The TennisMatchViz Visualization in Response to A User Question about Aces

useful for building our knowledge base. Other reason is its comments follow a fixed pattern, making it possible for our program to retrieve information from these comments. Although these comments are written in Chinese, we can translate them into English in the human-annotation stage.

Figure 6.2 shows the original Web page for the match. The lower-left corner is the commentary area where the latest comments are popped up regularly. If users go to the interface of our system which is a web page, they can see a “OPEN” button at the lower-right corner of the page. Click it, type in the address of the tennis web site in the pop-up window, and submit it, a visualization page for the match will be presented to the users (See Figure 6.3). The lower panel of the page contains a search box which can enable users' interaction with TennisVis. For example, if a user submit a question like “How many Aces have been served in the first two sets?”, Figure 6.4 is the returning visualization which shows the user the ACEs in the first two set of the match. Figure 6.5 is another visualization in response to the question “How many Volleys have been hit?”.



Your Question: how many volleys has been hit?

Figure (6.5) The TennisMatchViz Visualization in Response to A User Question about volleys

PART 7

CONCLUSIONS

In this dissertation, I showcased a Web-based text visualization program for Corpus-based English language learning and teaching called Text X-Ray. Students can use Text X-Ray to explore a corpus for language usage patterns. They can also compare their own work with similar articles in a corpus. A teacher can use Text X-Ray for corpus-based language teaching. She can also use Text X-Ray to analyze a corpus of her students' writings. For example, she can also compare her students' works with another corpus of students' writings.The main difference between Text X-Ray and other corpus tools is the text visualization interface. The text visualizations allow users to quickly examine a corpus at multiple levels of detail. Users can quickly see the complexity of texts by comparing sentence length, sentence grammar tree, word length, word frequency, and part of speech (POS) distribution. These visualizations, combined with traditional linguistic analysis, give users a more complete picture of the texts. Text visualization is rarely used corpus tools and this work is an attempt to fill this gap. In the near future, we plan to enhance Text X-ray with more features, such as a concordancer and better filtering functions. We are also experimenting with new techniques for visualizing text complexity.

A new indented level-based tree drawing algorithm is also discussed in this dissertation. This is a modified Reingold and Tilford algorithm that satisfies a new aesthetic rule – make the tree fit the wider aspect ratio of the newer computer display while preserving the order of the leaf nodes in a sentence. This is motivated by the need of visualizing parse trees in a text visualization application. The new algorithm creates a parse tree drawing that makes optimal use of the space while maintaining the word order for easy reading. The new algorithm also solves the problems of line crossings for tidy presentation. This tree drawing algorithm is a useful complement to the traditional level-based tree drawing algorithms. In addition to drawing parse trees, it can be applied to any tree structure where the leaf nodes need to be horizontally sorted.

I also presented a new storyline visualization method that focuses on presenting the time, location, character, and context dimensions of a narrative. There are two main contributions. The first is a novel visualization technique called Storygraph that presents temporal and spatial information in one view. Storygraph allows users to look for patterns across the temporal and spatial dimensions, something very difficult to do in other types of visualizations. However, Storygraph often suffers from cluttering and excessive line crossings when dealing with big data sets. The second contribution is a new heuristic method to reduce cluttering and line crossings in Storygraph. This feature, combined with a synchronized map view, makes it much easier for users to conduct spatio-temporal data analysis. The current system works well with structured and semi-structured data. In the future, we plan to continue improving Storygraph with better layout algorithm for spatial data. We are also looking into methods to use unstructured texts as input.

Finally, I presented a new visualization technique for presenting tennis match statistics and live comments. Retrieving data automatically from tennis match live blogging Web sites, this visualization is designed as a supplement to live tennis match coverage. It provides a quick overview of a tennis match statistics but can also provide many details on demand. While existing tennis data visualizations focus on post-match analysis, my visualization is the first to feature automatic data retrieval and live visual broadcasting. All the basic match statistics are presented in a series of mini-timelines. The concise form is particularly suitable for mobile devices. Users can interact with the visualization by typing a question. In the future, I plan to enhance our program with better natural language processing capabilities as well as the ability to integrate data from multiple sources.

REFERENCES

[1] "1854 broad street cholera outbreak," https://en.wikipedia.org/wiki/1854_Broad_Street_cholera_outbreak, accessed: 2016-06.

[2] "Introduction to data visualization: Visualization types," http://guides.library.duke.edu/datavis/vis_types, accessed: 2016-06.

[3] E. R. Tufte and P. Graves-Morris, *The visual display of quantitative information.* Graphics press Cheshire, CT, 1983, vol. 2, no. 9.

[4] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations," in *Visual Languages, 1996. Proceedings., IEEE Symposium on.* IEEE, 1996, pp. 336–343.

[5] U. Römer, "Corpus research applications in second language teaching," *Annual Review of Applied Linguistics*, vol. 31, pp. 205–225, 2011.

[6] H. Nesi and S. Gardner, *Genres across the disciplines: Student writing in higher education.* Cambridge University Press, 2012.

[7] A. O'keeffe, M. McCarthy, and R. Carter, *From corpus to classroom: Language use and language teaching.* Cambridge University Press, 2007.

[8] "Us open men's final: Djokovic beats federer," http://www.bbc.com/sport/live/tennis/34067072, accessed: 2015-09.

[9] "BYU," http://corpus.byu.edu, accessed: 2015-01.

[10] L. Anthony, P. Crosthwaite, T. Kim, T. Marchand, S. Yoon, S.-Y. Cho, E. Oh, N.-Y. Ryu, S.-H. Hong, H.-K. Lee *et al.*, "A critical look at software tools in corpus linguistics," *Linguistic Research*, vol. 30, no. 2, pp. 141–161, 2013.

[11] “Compleat Lexical Tutor,” http://www.lextutor.ca/, accessed: 2015-01.

[12] “WORD AND PHRASE,” http://www.wordandphrase.info/, accessed: 2015-01.

[13] “Text visualization browser,” http://textvis.lnu.se/, accessed: 2015-01.

[14] H. Siirtola, T. Säily, T. Nevalainen, and K.-J. Räihä, “Text Variation Explorer: Towards interactive visualization tools for corpus linguistics,” *International Journal of Corpus Linguistics*, vol. 19, no. 3, pp. 417–429, 2014.

[15] J. Nualart-Vilaplana, M. Pérez-Montoro, and M. Whitelaw, “How we draw texts: A review of approaches to text visualization and exploration,” *El profesional de la información*, vol. 23, no. 3, pp. 221–235, 2014.

[16] D. E. Knuth, “Optimum binary search trees,” *Acta informatica*, vol. 1, no. 1, pp. 14–25, 1971.

[17] C. Wetherell and A. Shannon, “Tidy drawings of trees,” *IEEE Transactions on Software Engineering*, no. 5, pp. 514–520, 1979.

[18] R. E. Sweet, “Empirical estimates of program entropy,” Xerox Res. Cent., Tech. Rep., 1978.

[19] E. M. Reingold and J. S. Tilford, “Tidier drawings of trees,” *IEEE Transactions on Software Engineering*, no. 2, pp. 223–228, 1981.

[20] J. Q. Walker, “A node-positioning algorithm for general trees,” *Software: Practice and Experience*, vol. 20, no. 7, pp. 685–705, 1990.

[21] C. Buchheim, M. Jünger, and S. Leipert, “Improving walkers algorithm to run in linear time,” in *Graph Drawing.* Springer, 2002, pp. 344–353.

[22] R. Tamassia, *Handbook of graph drawing and visualization.* CRC press, 2013.

[23] Y. Miyadera, K. Anzai, H. Unno, and T. Yaku, “Depth-first layout algorithm for trees,” *Information processing letters*, vol. 66, no. 4, pp. 187–194, 1998.

[24] A. Bloesch, "Aesthetic layout of generalized trees," *Software: Practice and Experience*, vol. 23, no. 8, pp. 817–827, 1993.

[25] B. Stein and F. Benteler, "On the generalized box-drawing of treessurvey and new technology," in *International Conference on Knowledge Management*, 2007, pp. 416–423.

[26] X. Li and J. Huang, "An improved generalized tree layout algorithm," in *Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference on*, vol. 2. IEEE, 2010, pp. 163–166.

[27] K. Marriott, P. Sbarski, T. van Gelder, D. Prager, and A. Bulka, "Hi-trees and their layout," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 3, pp. 290–304, 2011.

[28] A. Ploeg, "Drawing non-layered tidy trees in linear time," *Software: Practice and Experience*, 2013.

[29] E. Segel and J. Heer, "Narrative visualization: Telling stories with data," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 16, no. 6, pp. 1139–1148, 2010.

[30] J. Hullman and N. Diakopoulos, "Visualization rhetoric: Framing effects in narrative visualization," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, no. 12, pp. 2231–2240, 2011.

[31] J. Zhao, F. Chevalier, C. Collins, and R. Balakrishnan, "Facilitating discourse analysis with interactive visualization," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 18, no. 12, pp. 2639–2648, 2012.

[32] S. Havre, B. Hetzler, and L. Nowell, "Themeriver: Visualizing theme changes over time," in *Information Visualization, 2000. InfoVis 2000. IEEE Symposium on.* IEEE, 2000, pp. 115–123.

[33] D. Fisher, A. Hoff, G. Robertson, and M. Hurst, "Narratives: A visualization to track narrative events as they develop," in *Visual Analytics Science and Technology, 2008. VAST'08. IEEE Symposium on.* IEEE, 2008, pp. 115–122.

[34] M. Ogawa and K.-L. Ma, "Software evolution storylines," in *Proceedings of the 5th international symposium on Software visualization.* ACM, 2010, pp. 35–42.

[35] N. W. Kim, S. K. Card, and J. Heer, "Tracing genealogical data with timenets," in *Proceedings of the International Conference on Advanced Visual Interfaces.* ACM, 2010, pp. 241–248.

[36] P. H. Nguyen, K. Xu, R. Walker, and B. Wong, "Schemaline: Timeline visualization for sensemaking," in *Information Visualisation (IV), 2014 18th International Conference on.* IEEE, 2014, pp. 225–233.

[37] W. Zhu and C. Chen, "Storylines: Visual exploration and analysis in latent semantic spaces," *Computers & Graphics*, vol. 31, no. 3, pp. 338–349, 2007.

[38] S. Liu, J. Pu, Q. Luo, H. Qu, L. M. Ni, and R. Krishnan, "Vait: A visual analytics system for metropolitan transportation," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 14, no. 4, pp. 1586–1596, 2013.

[39] G. Sun, Y. Liu, W. Wu, R. Liang, and H. Qu, "Embedding temporal display into maps for occlusion-free visualization of spatio-temporal data," in *Pacific Visualization Symposium (PacificVis), 2014 IEEE.* IEEE, 2014, pp. 185–192.

[40] H. Liu, Y. Gao, L. Lu, S. Liu, H. Qu, and L. M. Ni, "Visual analysis of route diversity," in *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on.* IEEE, 2011, pp. 171–180.

[41] P. Bak, F. Mansmann, H. Janetzko, and D. A. Keim, "Spatiotemporal analysis of sensor logs using growth ring maps," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 15, no. 6, pp. 913–920, 2009.

[42] P. Gatalsky, N. Andrienko, and G. Andrienko, "Interactive analysis of event data using space-time cube," in *Information Visualisation, 2004. IV 2004. Proceedings. Eighth International Conference on.* IEEE, 2004, pp. 145–152.

[43] G. Andrienko, N. Andrienko, C. Hurter, S. Rinzivillo, and S. Wrobel, "Scalable analysis of movement data for extracting and exploring significant places," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 19, no. 7, pp. 1078–1094, 2013.

[44] T. Kapler and W. Wright, "Geotime information visualization," *Information Visualization*, vol. 4, no. 2, pp. 136–146, 2005.

[45] P. O. Kristensson, N. Dahlback, D. Anundi, M. Bjornstad, H. Gillberg, J. Haraldsson, I. Martensson, M. Nordvall, and J. Stahl, "An evaluation of space time cube representation of spatiotemporal patterns," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 15, no. 4, pp. 696–702, 2009.

[46] "Movie narrative charts," http://xkcd.com/657, accessed: 2015-03.

[47] T. Crnovrsanin, C. Muelder, C. Correa, and K.-L. Ma, "Proximity-based visualization of movement trace data," in *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on.* IEEE, 2009, pp. 11–18.

[48] Y. Tanahashi and K.-L. Ma, "Design considerations for optimizing storyline visualizations," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 18, no. 12, pp. 2679–2688, 2012.

[49] H. Tanahashi and K.-L. Ma, "An efficient framework for generating storyline visualizations from streaming data," *Visualization and Computer Graphics, IEEE Transactions on*, 2015.

[50] S. Liu, Y. Wu, E. Wei, M. Liu, and Y. Liu, "Storyflow: Tracking the evolution of stories," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 19, no. 12, pp. 2436–2445, 2013.

[51] N. Cao, Y.-R. Lin, X. Sun, D. Lazer, S. Liu, and H. Qu, "Whisper: Tracing the spatiotemporal process of information diffusion in real time," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 18, no. 12, pp. 2649–2658, 2012.

[52] D. Chu, D. A. Sheets, Y. Zhao, Y. Wu, J. Yang, M. Zheng, and G. Chen, "Visualizing hidden themes of taxi movement with semantic transformation," in *Pacific Visualization Symposium (PacificVis), 2014 IEEE*. IEEE, 2014, pp. 137–144.

[53] Z. Wang and X. Yuan, "Urban trajectory timeline visualization," in *2014 International Conference on Big Data and Smart Computing (BIGCOMP)*. IEEE, 2014, pp. 13–18.

[54] "The 1st workshop on sports data visualization," http://workshop.sportvis.com/, accessed: 2013-10.

[55] "Atp scores & stats," http://www.atpworldtour.com/en/scores/current/us-open/560/results?, accessed: 2015-08.

[56] "Roger federer," http://www.atpworldtour.com/en/scores/current/us-open/560/results?, accessed: 2015-09.

[57] H. Pileggi, C. D. Stolper, J. M. Boyle, and J. T. Stasko, "Snapshot: Visualization to propel ice hockey analytics," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 18, no. 12, pp. 2819–2828, 2012.

[58] F. Beck, M. Burch, and D. Weiskopf, "Visual comparison of time-varying athletes' performance," in *The 1st Workshop on Sports Data Visualization*. IEEE, 2013.

[59] B. Moon and R. Brath, "Bloomberg sports visualization for pitch analysis," in *The 1st Workshop on Sports Data Visualization*. IEEE, 2013.

[60] D. Demaj, "Geovisualizing spatio-temporal patterns in tennis: An alternative approach to post-match analysis," 2013. [Online]. Available: http://gamesetmap.com/?page_id=2

[61] "Daily data visualization: Analyzing federer's quarterfinal comeback," http://www.si.com/tennis/2014/09/06/daily-data-visualization-analyzing-federers-quarterfinal-comeback, accessed: 2014-09.

[62] "Atp tennis world tour venue map," http://gamesetmap.com/atp2015/, accessed: 2015-09.

[63] R. Cava and C. Dal Sasso Freitas, "Glyphs in matrix representation of graphs for displaying soccer games results," in *The 1st Workshop on Sports Data Visualization*. IEEE, 2013.

[64] S. G. Owens and T. Jankun-Kelly, "Visualizations for exploration of american football season and play data," in *The 1st Workshop on Sports Data Visualization*. IEEE, 2013.

[65] R. Sisneros and M. Van Moer, "Expanding plus-minus for visual and statistical analysis of nba box-score data," in *The 1st Workshop on Sports Data Visualization*. IEEE, 2013.

[66] "Infographic design," http://visual.ly/creative-services/infographics, accessed: 2015-09.

[67] T. Polk, J. Yang, Y. Hu, and Y. Zhao, "Tennivis: Visualization for tennis match analysis," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 20, no. 12, pp. 2339–2348, 2014.

[68] "Text X-Ray," http://textvis.gsu.edu:8080/NLPVis/html/index.html, accessed: 2015-01.

[69] R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng, "Parsing with compositional vector grammars," in *In Proceedings of the ACL conference*, 2013.

[70] "jQuery UI," http://jqueryui.com/, accessed: 2015-01.

[71] "D3.js," http://d3js.org/, accessed: 2015-01.

[72] X. He and Y. Zhu, "An indented level-based tree drawing algorithm for text visualization," in *submitted to Graphics Interface Conference*, 2015.

[73] M. Bostock, V. Ogievetsky, and J. Heer, "D^3 data-driven documents," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2301–2309, 2011.

[74] G. Ellis and A. Dix, "A taxonomy of clutter reduction for information visualisation," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 13, no. 6, pp. 1216–1223, 2007.

[75] P. Eades and N. C. Wormald, *The median heuristic for drawing 2-layered networks.* University of Queensland, Department of Computer Science, 1986.

[76] J. Sackmann, "The match charting project," https://github.com/JeffSackmann/tennis_MatchChartingProject, accessed: 2016-06.