FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Boundary Labeling for annotated documents

TestSubtitle

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

EnglischerCurrName

by

Jakob Klinger
Registration Number 1125755

to the Faculty of Informatics
at the TU Wien

Advisor: Pretitle Forename Surname, Posttitle

Vienna, 25th September, 2017

Jakob Klinger	Forename Surname

Erklärung zur Verfassung der Arbeit

Jakob Klinger
TODO: ADDRESS!

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 25. September 2017

Jakob Klinger

Contents

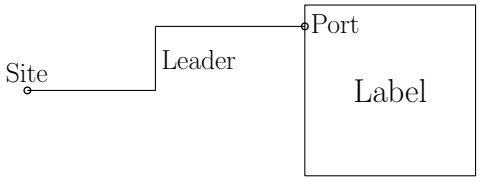

Figure 0.1: Illustrated guide to the labeling terminology

0.1 Introduction

Annotating a document is usually solved by adding footnotes or figures in an appropriate position and adding a simple reference in the text, leaving the reader to find the referenced content by themselves. If a more obvious connection between the text and the referenced content is required, the reference is visibly connected to the text by drawing a straight line or a more complex path between them.

In this paper, we will look at ways to use Boundary Labeling, which means that all annotations will be placed outside of the text they are referencing and will be visually connected to the feature they are referencing. (See also [2])

The guidelines on how to create an optimal labeling are as follows: the connections should be as direct as possible, no important information should be obscured, and it should be easily discernable which Label belongs to which site. These three easily come into conflict with one another, as the text usually is very dense and leaves little space for lines in between, yet one shouldn't allow them to pass through the text, as this makes the text harder to read.

While there are many papers discussing Boundary Labeling in general, only very few exist that apply this concept to written text. Generally, this approach isn't used very often, and tends to use simplistic algorithms which produce mediocre results. However, the papers that do discuss boundary labeling in text offer interesting contributions.

In the paper about the Luatodonotes-Package[3] illustrates some of the different styles of drawing these connecting paths, and came to the conclusion that paths without bends are easier to follow. However, most solutions proposed in that paper do not consider whether a path overlaps with text or not, which results in a decrease in readability.

The paper by Loose[4] on the other hand is based around only using the free space between lines and words, which produces longer paths, and forces curves, but doesn't obscure any part of the text.

0.2 Terminology and Fundamentals

While Boundary Labeling (or an equivalent concept) can be applied to a space with different geometry or more dimensions, this paper will only concern itself with two-dimensonal, euclidean space. To easily reference important concepts, some additional terminology will be introduced as well. (See Fig. 0.1 for a visual explanation)

A *Graph* $G = \langle V, E \rangle$ is a tuple of *Vertices* ($V = \{v_1, v_2, ..., v_n\}$) and *Edges* ($E = \{e_1, e_2, ..., e_m\}$). A vertex $v_i (i \in \mathbb{N})$ is a featureless object. Each edge $e_i \, (i \in \mathbb{N})$ is a relation between two vertices ($E : V \to V \times V$). The vertices forming the edge are considered *adjacent* to each other. Edges can also be directional, or have a weight, which affects how they are treated by algorithms. A *Path* ($P = v_1, ..., v_h$) is an ordered series of vertices, where each vertex must have an edge connecting it to the subesquent one.

Polylines are a connected series of *Line segments*. Line segments are straight lines that contain each point between their starting and end point. *Labels* hold additional information and are represented as boxes containing this information. They are usually placed in the *Label area* which is an area designated to hold labels and is located off to a side so the labels don't obscure anything. The point or object that this information refers to is called a *Site*. The site and the label are connected via a *Leader*, a polyline that can be further classified by looking at the orientation of its segments: *O-Segments* run orthogonally to the border of the label area. *P-Segments* run parallel to the border of the label area, and as such must be combined with other segments for the leader to reach its destination. *S-Segments* are not required to have any particular orientation, and simply connect their start and einding points in a straight line. The leader's name is created by combining the name of the segments - for example, the leader from Fig 0.1 would be classified as an OPO-Leader. The location where a leader connects to the label is called the *Port*. It can either be located freely along the label's edge, or restricted to pre-defined positions.

- **Monotonous/Monotonicity:** Describes a steady progression towards a direction or goal. Will be used in this paper to indicate that there's no loss of progress made from the site to the label, effectively restricting the direction a leader can go at each bend.

- **Depth-first search:** A searching algorithm that continues along a set of rules as far as possible until either the goal is found, or backtracking is required. A common example is navigating a maze by turning the same way whenever possible.

0.3 The program

In our program, the focus was put on keeping the text as readable as possible. This means that leaders are only allowed in the space between words or lines, which can make certain locations for sites unavailable, as there might already too many leaders routed through that area to insert yet another for the new site. We also wanted the leaders to be monotonous, which means that they will be one of the shortest possible connection between port and site locations. Our final priority was to efficiently use the space for the labels, which meant that each label should initially be placed as close to the available area's border as possible. While this allows for more space in which labels can be placed, it also means that the distance between Label and site often increases, which leads to longer leaders.

Algorithm

To easily create leaders that exclusively use the space between words and lines, we decided to use a graph similar to the one Loose [4] used, placing vertices above and below each word's beginning and end, and connecting them with their horizontally and vertically adjacent neighbours.For the sites, we inserted an additional vertex above the center of the word, which will serve as the leader's starting point. We also decided to give the graph's edges a capacity of 1, which means that each leader will always use edges which haven't already be taken by other leaders. While this causes some sites to become un-routable due to another leader passing right through their starting vertex, it makes visualizing the leaders much easier.

Our first approach was going through the labels in the order they appear in the text and placing each as far up as possible, to maximize the remaining space other labels could use. We also opted to use fixed ports located in the top left corner of the annotation, as this allowed us to unambiguously place any annotation by knowing either their port's location, wich is also the leader's ending point. However, since putting the labels right next to the text would result in the labels' positions being restricted by the text's line spacing, we then decided to leave a buffer zone in between the two areas, which allows the leaders to adjust between the position forced by the line spacing and the label's final position. As this allowed us to place the labels more freely, we adjusted the label's positions after all labels were placed to reduce the distance that needed to be covered in the buffer area.

Implementation

The program was written in Java, using only JGraphT[1] as additional library. Since we only want to create leaders that don't intersect with the text, the graph was created alongside the placement of the words on the canvas, as it was easy to extract measurements at this point. Due to some problems with Java's various methods of calculating linebreaks, this process was done completely by hand.

As the Graph's vertices represent fixed locations on the canvas, they each have coordinates associated with them, with the vertices representing the sites containing extra information, such as references to the corresponding annotation and to the leader connecting the two (if existent). This additional information is used for both routing and drawing the leaders: The routing was done via a depth-first search algorithm that prioritized vertices located further above if possible, and vertices to the right otherwise. After the algorithm terminates, it returns a vertex-based graph walk describing the route from the site to the border of the text area alongside information about where to draw the OPO-segment in the buffer zone if successful, or a walk containing only the starting vertex, if no valid path was found. Using this information alongside the positional data stored in the vertices, the polyline representing the leader can be easily drawn.

Bibliography

[1] Jgrapht - a free java graph library.

[2] Michael A. Bekos, Michael Kaufmann, Antonios Symvonis, and Alexander Wolff. Boundary labeling: Models and efficient algorithms for rectangular maps. *Computational Geometry*, 36(3):215 – 236, 2007.

[3] Philipp Kindermann, Fabian Lipp, and Alexander Wolff. *Luatodonotes: Boundary Labeling for Annotations in Texts*, pages 76–88. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[4] Amrei Loose. Annotation von texten unter berücksichtigung von textzwischenräumen. Master's thesis, Karlsruhe Institute of Technology, 2015.