On the beauty and usability of tag clouds

Christin Seifert
Know-Center Graz
cseifert@know-center.at

Barbara Kump
Institute for Knowledge Management
Graz University of Technology
bkump@tugraz.at

Wolfgang Kienreich
Know-Center Graz
wkien@know-center.at

Gisela Granitzer
Know-Center Graz
ggrani@know-center.at

Michael Granitzer
Know-Center Graz
mgrani@know-center.at

Abstract

Tag clouds are text-based visual representations of a set of tags usually depicting tag importance by font size. Recent trends in social and collaborative software have greatly increased the popularity of this type of visualization. This paper proposes a family of novel algorithms for tag cloud layout and presents evaluation results obtained from an extensive user study and a technical evaluation. The algorithms address issues found in many common approaches, for example large whitespaces, overlapping tags and restriction to specific boundaries. The layouts computed by these algorithms are compact and clear, have small whitespaces and may feature arbitrary convex polygons as boundaries. The results of the user study and the technical evaluation enable designers to devise a combination of algorithm and parameters which produces satisfying tag cloud layouts for many application scenarios.

1. Introduction

A tag cloud is a text-based visual representation of a set of tags which usually depicts tag importance by font size. Recent trends in social and collaborative software have greatly increased the popularity of this representation form. Sites like flickr [1] and del.icio.us [2] employ tag clouds to provide users with a comprehensible overview on the content of large, tagged repositories. When individual tags link to relevant subsets of repository content, tag clouds also become powerful instruments of topical browsing.

Tag clouds occupy a peculiar niche in the domain of visualization. Designers have to balance aesthetic considerations, for example with regard to font size and tag arrangement, against the capabilities of algorithms to produce optimized layouts. In this task, they are limited by shortcomings of existing algorithms and by lack of knowledge about user perception and acceptance of resulting layouts.

In this publication, we present a novel family of layout algorithms for tag clouds. These algorithms are able to produce layouts that inscribe tags into arbitrary convex polygons, thus including the standard rectangular layout as a special case. Furthermore, these algorithms are capable of dynamically adapting tag font size within predefined constraints. This enables placement of a maximum number of tags while retaining the representation of tag importance by size. The inscription of tags into arbitrary convex polygons paves the way for many new design ideas. For example, it enables the extension of the tag cloud concept to hierarchically structured sets of tags. Common visualization approaches for hierarchically structured repositories employ Voronoi diagrams to subdivide given areas into regions corresponding to branches in the hierarchy. Evaluations of such systems have shown that the usability of the resulting visualizations is severely limited at the level of individual repository items [2]. The use of tag clouds inscribed into Voronoi cells could be a solution for this problem.

We have evaluated this family of algorithms in a user study and discuss the study results in considerable detail. It is our hope that a family of new algorithms, together with evaluation results clearly documenting their benefits and shortcomings, will significantly ease the task of tag cloud designers.

2. Related Work

The standard tag cloud **visualization** used by Web 2.0 websites such as del.icio.us and flickr uses a rectangular line-by-line layout with different fontsizes to indicate the

[1] http://www.flickr.com
[2] http://del.icio.us

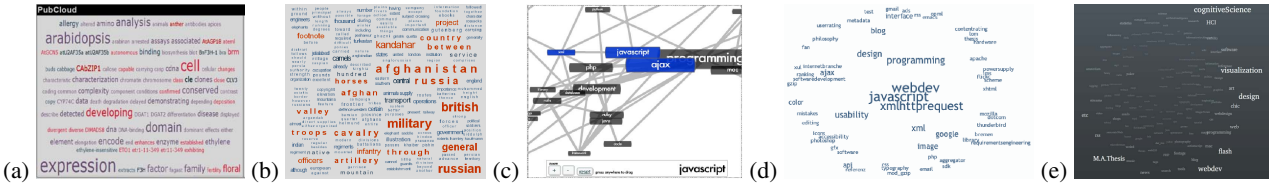


Figure 1. An overview of existing tag layouts: (a) PubCloud layout [6], (b) in nested HTML tables [5], (c) by a graph structure [8], (d) elastic tag maps [9] and (e) circular layout with focus [1].

relevance of the tags. The tags may be sorted by their relevance or alphabetically. Kuo et. al [6] further include color for the tags to indicate the actuality of the tag, which is calculated from the average publication data of the underlying documents (see figure 1 (a)). These visualizations naturally produce a lot of whitespaces, an issue which was tackled by Kaser & Lemire [5] who used an EDA packing algorithm to layout tags in nested tables for HTML based websites (see figure 1 (b)). Shaw [8] was the first who broke-up with the rectangular layout and further introduced the visualization of relations between tags. His tag map is a graph-like 2D representation of tags, where tags correspond to nodes and related tags are connected by an edge (see figure 1 (c)). However the visualization contains a lot of whitespaces and tags are overlapping each other, problems also not solved by Stefaner [9] who introduced elastic tag maps. The tags are placed in an nearly circular 2D space using PCA and CCA, with the most relevant tags defining the extremes of the spanned space (see figure 1(d)). Bielenberg and Zacher [1] also proposed a circular layout, where the font-size and position from center to orbits show the relevance of the tags. The most relevant tags are placed in the center of the circle and referred to as focus. Also in this visualizations huge whitespaces and overlaid tags remain (see figure 1 (e)). Another interesting approach was recently published by Stein & Benteler [10]. The authors proposed a generalized algorithm for box drawing extending the tree layout algorithms for fixed node sizes. However, the problem of how to construct the bounding boxes for the tags, still remains.

We propose a family of algorithms which solve the above mentioned open issues, i.e. (unused) whitespace, restriction to certain bounding areas (rectangle, circle) and the overlapping of tags. All of the existing algorithms take the font size of the tags as a fixed parameter, since it is calculated prior to the layout step. In our approach the variation of the tags' font sizes within certain constraints leads to more flexibility for the layout.

There are only a very few **user evaluation studies** existing in the literature. Kuo et al. [6] evaluated their PubCloud interface by comparing it to the PubMed list layout. The 20 participants of the study had to answer questions of a unknown biomedical field. The response time and the correctness of the answers in both, the list and the cloud interfaces were measured. The authors found out that the response time was less when using lists, but the quality of the answer was improved by the cloud layout. Furthermore clouds support users in summarizing descriptive information, but do not help in identifying relational concepts.

Halvey & Keane [3] evaluated the influence of different tag cloud properties on search time. They compared vertical and horizontal lists to tag clouds, all of them either sorted by tag's relevance or alphabetically. The 62 participants had to find a given item within ten items presented. The authors found that task completion took the longest for the tag cloud presentation. In all presentations alphabetization helped the participants in their search. Tags with larger font sizes were identified quicker than tags with smaller font sizes. The tag's position was also found to influence the search time. Tags at the beginning of the list or in the upper-left corner of the tag cloud were found much quicker.

Another study [7] evaluated the effectiveness of different tag cloud layouts and further proposes an evaluation paradigm for tag clouds. The first experiment tested the influence of the font-size and the word location on recall for 13 participants. The finding were similar to Halvey and Keane [3]: words with larger fonts and those located in the upper-left corner were significantly better recalled. The second experiment evaluated the influence of the font-size and word-location on gist and recognition for 11 participants. The authors found a significant effect of layout on gisting. The bin-packed tag cloud layout was found to support second best the identification of categories present after the frequency ordered list layout. There was no effect found of the layout for the recognition task, where the font-size showed to have a great influence - words laid out with large font-sizes were recognized much quicker.

In the user study described in this paper we address both, the helpfulness of a certain tag cloud layout and the perceived beauty of this layout in order to determine the user acceptance of our proposed algorithms. Furthermore, we conducted a technical evaluation and incorporated the results with the results of the user study to clarify the usability and applicability of our proposed algorithms.

3. Algorithm

In this section we describe our layout algorithm. First, we show possible ways to influence the tags' visual representation. Second we explain the calculation of the bounding boxes, and, third, we describe the the core of the algorithm - the layout of the bounding boxes. Finally, we show possible combinations of the previous steps and conduct four different algorithms.

T is the set of all tags: $T = \{t_i | t_i \text{ is a Tag}, 1 <= i <= n\}$. A tag t_i is given by a tag string t_i^s and a tag relevance value t_i^r (also called the weight of the tag). The algorithm assigns a certain position and font size to a subset of the tags, or ideally, all tags. We have no a-priory knowledge about the tags, however, there are some constraints on the layout: (i) the fontsize is not fixed, but has to be in a sensible interval (too small font sizes are not readable, too large font size range does not give a visually attractive layout) [3, 4], (ii) strings can be truncated, but have to consist of at least three letters, and (iii) the font family should be unique for all tags since we do not want it to influence the perceived relevance of a tag. To account for these constraints we introduced three parameters for our algorithm: thresholds for the maximum and minimum font size allowed θ_{max} and θ_{min} and an initial value for the font size assigned to the tag with the minimal weight s^0_{min}. The initial value for the font size assigned to the most relevant tag, s^0_{max} is set to θ_{max}. s^0_{min} and s^0_{max} define the current font range interval s^0_r.

There are two possible ways to change the size of the visual representation of a tag: to shrink/enlarge the font size, i.e., the height and width of the tag or to truncate the tag string, i.e. to reduce the width of the tag. Given our input parameters θ_{max}, θ_{min} and s^0_{min} we identified three processes to influence the tags' visual representation and therewith the bounding boxes of the tags: (i) the shift of the font size interval, (ii) the scaling of the font size interval and (iii) the truncation of tag strings. If the initial layout trial was not successful (i.e. not all tags could be laid out with initial parameter settings), one of these processes changes the tags' visual representation leading to new bounding boxes. Another layout trial based on the new bounding boxes is then started. An overview of the algorithm is depicted in figure 2.

We do not claim to achieve optimal layouts, we heuristically find a good layout that meets the following criteria:

1. The most relevant tags are laid out, i.e.: if a tag t_i is laid out, all tags t_j which could not be laid out have a lower or equal relevance value ($t_i^r \geq t_j^r$ for $t_i, t_j \in T$).
2. The tags are laid out comparable to an orbital layout, starting with the most relevant tag in the center of mass of the polygon. (Center of mass was chosen, because it is defined for arbitrary polygons and yields consistent layouts over a wide range of shapes.)

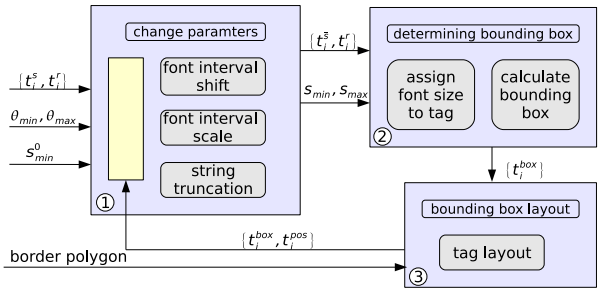


Figure 2. ① Algorithm Overview: Depending on the input values and the success rate of the previous layout attempt new parameter values are determined. ② The font-size and then the bounding boxes for all tags are determined. ③ The bounding boxes are laid out.

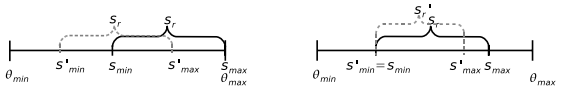
3. The bounding boxes of two distinct tags do not overlap.
4. Two tags with the same relevance value are drawn using the same font size.
5. A tag with a higher relevance value as another tag is drawn with an at least equal font size.

3.1. Influencing the Tags' Visual Representation

This section describes possibilities to adapt parameters after an unsuccessful layout trial. Section 3.4 then discusses possible combinations of these steps.

As mentioned above, the tags' visual representation can be changed by (i) shifting the font size interval, (ii) scaling the font size interval and (iii) truncating the tags' strings. The goal of the first method is the reduction of all font sizes in a linear way in order to get smaller bounding boxes. The interval shift is defined by: $s'_{min} = s_{min} - 1$ and $s'_{max} = s_{max} - 1$, which means the font size range keeps constant, $s'_r = s_r$. The minimum font size must not be less than the minimum font threshold: $s'_{min} \leq \theta_{min}$. Figure 3(a) depicts the procedure. The scaling of the font size interval leads to a smaller range of font sizes by reducing the maximum value ($s'_{max} = s_{max} - 1$) while keeping the minimum value constant ($s'_{min} = s_{min}$). Therefore the font range decreases: $s'_r = s_r - 1$, with $s'_{max} \geq s'_{min}$ (see Figure 3(b)). The third method, the string truncation, substitutes the last letter of the tag string t_s with three dots. If this is done iteratively, the string becomes shorter while retaining as much information as possible (the beginning of the word and the indication, that the word continues). The minimum string length is set to 3. The truncation of strings

leads to shorter string and therefore bounding boxes with smaller width.



(a) Font Size Shift (b) Font Size Scale

Figure 3. The font size interval can be shifted and scaled to decrease the size of the bounding boxes.

3.2. Determining the Tags' Bounding Boxes

This section describes how the bounding boxes are determined depending on the current font size settings and the tags' relevance value. We are given the relevance values $t_i^r \in \Re$ of the tags t_i and the maximum and minimum font size values s_{min} and s_{max}. We define a function m that maps tag relevance values to font sizes: $m : t_i^r \rightarrow t_i^s$, such that $s_{min} \leq t_i^s \leq s_{max}$, $1 \leq i \leq n$. Hoffmann [4] identified three different font size distribution algorithms, the linear method, the logarithmic method and a clustering of font sizes. We implemented the linear and the logarithmic distribution. For the linear distribution, the font size t_k^s of the tag t_k is calculated as

$$t_k^s = (t_k^r - \min_i t_i^r) \cdot \frac{s_{\max} - s_{\min}}{\max_i t_i^r - \min_i t_i^r} + s_{min}$$

Accordingly, for the logarithmic distribution, the following equation is used

$$t_k^s = (\log t_k^r - \min_i \log t_i^r) \cdot \frac{s_{\max} - s_{\min}}{\max_i \log t_i^r - \min_i \log t_i^r} + s_{min}$$

Independently on which of the above methods was used, each tag $t_i \in T$ now has a valid font size t_i^s assigned. The width and the height of the bounding box t_i^{box} for the current graphics display is now determined by using library functions calculating a string's bounding box given a font size and a string.

3.3. Rectangle Placement

In this section we describe the core algorithm - the layout of rectangles in a convex polygon B. A rectangle $r = (w, h)$ is given by its width w and height h. R is the set of all rectangles $R = \{r_i | 1 \leq i \leq n\}$. The goal is to determine a subset $R' \subseteq R, |R'| = k \leq n$, and a set P of align points $P = \{p_i = (x_i, y_i) \mid 1 \leq i \leq k, p_i$ is the bottom left corner of $r_i\}$ such that

- all r_i are fully inside the polygon B
- no two r_i and r_j do overlap for $i \neq j$
- R' contains all rectangles with the largest height, i.e. $h_i \geq h_j, \forall r_i = (w_i, h_i) \in R$ and $r_j = (w_j, h_j) \in R \backslash R'$

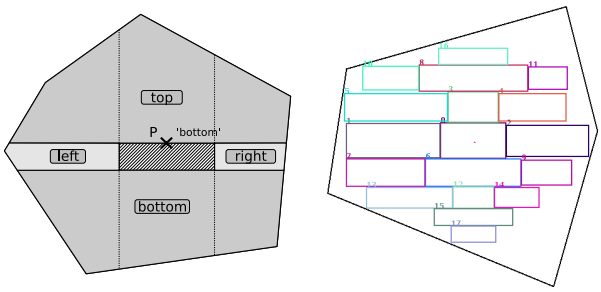
The last condition arises from the overall goal to layout tags. In case, that not all rectangles would fit in the polygon, we want to drop the rectangles that represent the least relevant tags. The rectangles represent tags, their height is proportional to the tags' font size and therefore proportional to the relevance of the tag (see Section 3.2). There are three more requirements to the layout algorithm arising from the tag layout application: (i) horizontal layout should be preferred to vertical layout to take the human westernized reading direction into account, (ii) compact layout is preferred, such that there is as little whitespace between boxes as possible, (iii) the most important tags should be in the center of the polygon as suggested by the notion of focus in [1].

The algorithm is as follows: The rectangles are sorted descendingly by their height. The first rectangle is placed centered in the center of mass of the polygon. The edges of this rectangle split the polygon into four new polygons as depicted in figure 4(a). The resulting polygons are stored in a list L together with the align point P and an align hint. The align point and the align hint are used for placing the next rectangle in the polygon. For the clipped *top*-polygon, the align hint will be *bottom* and the align point is the center point of the top-edge of the rectangle, see figure 4(a) for an example. Additionally, for each new polygon a priority value is stored. The priority v of a region is calculated from the distance d of its center of mass to the center of mass of the original polygon B and a weight value θ: $v = \frac{1}{d} \cdot \theta$, with

$$\theta = \begin{cases} 1, & \text{if region is top or bottom region} \\ 1.5, & \text{if region is left or right region} \end{cases}$$

This choice enforces the algorithm to create a compact and preferable line-by-line layout of the rectangles.

By placing the first rectangle four new polygons were created. For each of the remaining rectangles (remember that the rectangles were sorted by height), all polygons with a larger area than this rectangle are processed in order of their priority value. When a rectangle could be placed in a polygon at the align point's location according to the align hint, zero to three new polygons are created and added to the list and the old polygon is removed from the list. In case the rectangle did not fit in the polygon the next polygon (next less priority value) gets tested. The algorithm stops, if for a rectangle all polygons from the list which at least the area as the rectangle were tested and the rectangle did not fit in one of these.



(a) clipping regions (b) alignment example

Figure 4. Rectangle Layout: 4(a) clipping regions for the first rectangle, 4(b) 18 of 20 aligned random rectangles in a polygon.

3.4. Combining the Steps

This section describes how the individual steps are combined into a system for tag layout. In figure 2 we gave a coarse overview of the whole procedure without describing the sequence of the individual steps. We identified four sensible possibilities to combine the steps (in a static way):

Shift-Trunc (ST): Try font interval shift first, if no success start to truncate tag strings. After one truncation reset the font interval. Try to lay out again. Go on with the interval shift. If no more shift possible, truncate again, and so on. Stop, if all tags are laid out or no more truncation is possible.

Shift-Trunc-Scale (STC): An extension of Shift-Trunc, which does not stop if no more truncation is possible, but tries to scale the font size interval instead. It stops if the interval can not be scaled anymore.

Shift-Scale-Trunc (SCT): Try font interval shift first, if no success try font interval scaling. If no success, try further with string truncation.

Trunc-Shift-Scale (TSC): Try string truncation first, if no success try font interval shifting. If still no success try font interval scaling.

Algorithm 1 Trunc-Shift-Scale

```
while not all boxes laid out and truncation possible do
  while not all boxes laid out and scaling possible do
    while not all boxes laid out and shifting possible do
      createBoundingBoxes()
      layOutBoxes()
      shiftFontInterval()
    end while
    scaleFontInterval()
  end while
  truncateTagStrings()
end while
```

As can be seen, the four proposed combination differ in the presence and sequence of their elements. As an example, algorithm 1 shows the pseudocode for the combination Trunc-Shift-Scale.

The sequence of the sub-processes is fixed for each iteration of the outer loop. There are many approaches applicable to make the sequence more dynamic, some will be discussed in Future Work.

4. Evaluation

We investigated whether one of the proposed algorithms (ST, STC, SCT, TSC) outperforms the others in terms of runtime, number of laid out tags, and the perceived 'beauty' and 'helpfulness' of the produced layout. We therefore performed a technical evaluation and a user study.

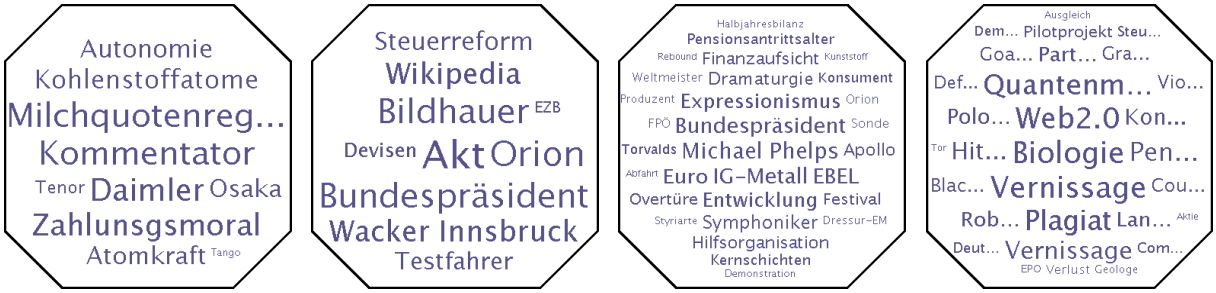
4.1. Data

In the following we describe the data we used for both, the technical evaluation and the user study. We defined five categories: politics, economy, sports, art and culture, and science and technology and selected 100 key words for each category from daily Austrian newspapers and German and Austrian news-sites. We used German words instead of English ones to avoid the influence of uncertainty in the pronunciation of English words in the user study. The length of the words ranged from 2 to 24 letters. The border polygon was a fixed octagon as shown in figure 5. In order to have only two variable parameters (algorithm and number of tags) we set $\theta_{max} = 50pt$, $\theta_{min} = 12pt$ and $s_{min} = 20pt$ and used the linear font size distribution function. These parameters were tested to give a potentially nice layout for chosen polygon.

4.2. Technical Evaluation

For the technical evaluation we created 600 tag clouds with random tags and random relevance values taken from a uniform distribution for each of the four algorithms, 300 of them consisting of 10 tags and 300 of them consisting of 30 tags. For each tag cloud we measured the following values:

Tags:	The number of actual laid out tags.
Time:	The time in milliseconds needed for the layout.
Area%:	The area used by the tags' bounding boxes relative to the area of the polygon.
Letters:	The total number of cropped letters.
Min-Font:	The minimal font size used for the layout.
Max-Font:	The maximal font size used for the layout.
Font-Shift:	The number of steps, the font interval was shifted by one point.



(a) ST (b) STC (c) SCT (d) TSC

Figure 5. Four example tag clouds presented in the user test with 10, 10, 30 and 30 tags (from left to right). Due to the design of the algorithm, there is no difference between ST and STC when laying out only a few tags.

Font-Scale: The ratio the font interval was scaled (a value of 0.5 means the font interval was scaled by the algorithm to half of its originally given size).

Results: The results of the technical evaluation are summarized in Table 1 for 10 tags and in Table 2 for 30 tags.

10 Tags	**ST**	**STC**	**SCT**	**TSC**
Tags	10	10	10	10
Time in ms	**1.38**	1.56	**1.34**	2.37
Area in%	54.39	55.27	53.51	**61.82**
Letters	0.72	0.69	**0.0**	8.62
Min-Font	16.72	16.74	16.80	19.85
Max-Font	46.72	46.74	46.16	49.85
Font-Shift	3.28	3.26	3.84	0.15
Font-Scale	1.0	1.0	0.98	1.0

Table 1. Results for 300 random trials for each of the four algorithm with 10 tags as input (mean values are displayed).

As expected, the runtime of the algorithms depended on the number of tags, it was in the range of 1-3 ms for 10 tags and 50-400 ms for 30 tags, see figures 6(c) and 6(d) for a comparison. All but one algorithm could successfully lay out all given tags; in the 30 tags case, the algorithm ST could only lay out 12.6 tags on average. At maximum 76.6% (STC, 30 tags) and at least 53.5% (STC, 10 tags) of the polygon area could be filled. For 10 tags, no algorithm needed to shift the font size to the minimum allowed font size θ_{min}, in contrary to the 30 tags case. Further, the maximum font size was larger in the 10 tags case than in the 30 tags case for all algorithms, see figure 6(b) for a visualization.

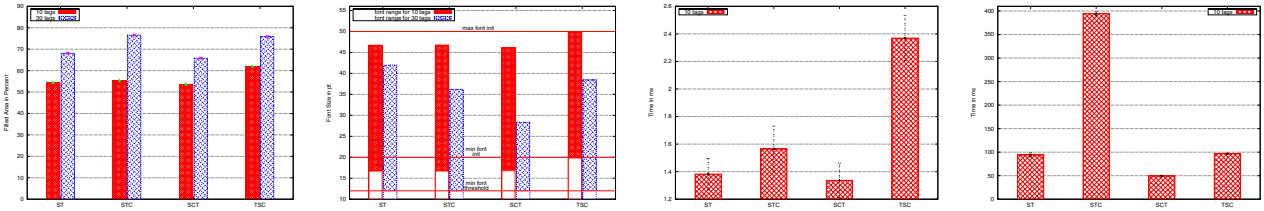
30 Tags	**ST**	**STC**	**SCT**	**TSC**
Tags	12.55	30	30	30
Time in ms	94.24	394.61	**49.61**	96.69
Area in%	68.13	**76.58**	65.81	**75.91**
Letters	33.54	111.85	**0.0**	139.57
Min-Font	12	12	12.01	12.05
Max-Font	42.0	36.21	28.36	38.43
Font-Shift	8.0	13.79	21.64	11.57
Font-Scale	1	0.81	0.55	0.88

Table 2. Results for 300 random trials for each of the four algorithm with 30 tags as input (mean values are displayed).

Discussion: From the technical evaluation we could not conclude a clear winning algorithm. The performance of the algorithms heavily depended on the number of tags.

For 10 tags, algorithm TSC outperformed the others in terms of the filled area and the font-range (see figures 6(b) and 6(a)), i.e. tags are nicely distributed all over the area with large and largely differing font sizes. However, TSC is clearly outperformed by the others in terms of processing time and the number of cropped letters, processing takes about 1 ms longer and information is lost by truncating the tag strings. SCT was the fastest of the four algorithms and did not crop any letters from the tag strings.

For 30 tags, algorithm SCT showed good results regarding processing time, number of tags and number of cropped letters, but poorly filled the polygon area. Algorithm TSC performed well with regard to filled area, number of tags and the font distribution, but need to crop much more letters than the other three algorithms. Comparing SCT and TSC,



(a) Area (b) Fonts (c) Time, 10 tags (d) Time, 30 tags

Figure 6. Diagrams for the technical evaluation showing the mean values for the 300 trials. In 6(c) and 6(d) also the standard deviation is shown.

both pay their good performance with loss of information. SCT heavily scales the font interval which leads to less well distinguishable tag relevances in the layout (between-tag information loss), whereas TSC crops a lot of letters which leads to an inner-tag information loss.

4.3. User Evaluation

In the user evaluation we wanted to find out whether the algorithm and the number of tags have an influence on the reaction time and the correctness for identifying the most important word in a tag cloud. Moreover, we wanted to examine the perceived beauty, and the perceived helpfulness of the different tag clouds.

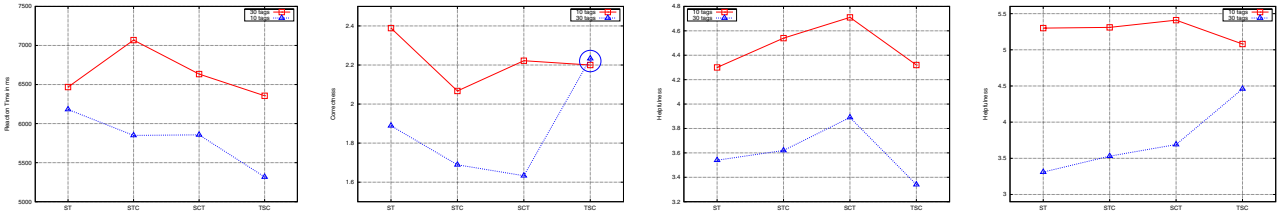
Design: We used a 2x4 within-subjects design with the two independent variables algorithm and the number of tags. We have two different levels for the factor number of tags (10 tags and 30 tags) and four different levels for the factor algorithm (ST, STC, SCT and TSC) which results in 8 different combinations (conditions). Each person participated in each condition, i.e. we did not group the participants. This means that we have eight different conditions. Dependent variables were: (i) reaction time (in milliseconds), (ii) correctness (0-3), (iii) beauty, (iv) helpfulness. The reaction time [ms] is defined as the time the participants needed to name the three most important words within the tag cloud. Correctness was measured using the precomputed ground truth. Since the relevance value for each tag is known, we could find out which of the three most important tags were identified by the user. Correctness therefore has a value between 0 (none identified) and 3 (all identified). Beauty and helpfulness, both were rated by the user on a scale between 1 (not beautiful/not helpful at all) and 7 (very helpful/very beautiful).

Participants: 30 German-speaking volunteers (30% technical professionals) participated in the 15 minutes evaluation, 11 males and 19 females. The age of the persons ranged between 14 and 57 years with 17 persons being between 25 and 35 years. 20 of them stated to have no prior-knowledge about tag clouds, only one person stated to have high prior-knowledge.

Test Material: For the user evaluation we created six tag clouds for each of the four algorithms with random tags and random relevance values taken from a uniform distribution. Three of the tag clouds for each algorithm consisted of 10 tags and three of 30 tags. Alltogether we therefore had 24 tag clouds, three of them are shown in figure 5. All other parameters were set to the same values as for the technical evaluation.

Procedure: The evaluation comprised of three trials depicted in figure 7. In each trial all 24 tag clouds were shown to each participant in the same sequence. Before the first trial the participants obtained the instruction to name as fast as possible the three tags they regarded as most relevant on the first glance. We did not give any hint in which way a relevant tag was laid out. After the introduction the 24 tag clouds were presented to the participants one after another. The participants had to speak out loud the answer to the question “Which are the three most important words in the picture?”. The experimenter noted the answers. Then the participants clicked to see the next image. We measured the reaction time (the time between the display of the image and the user’s click to see the next image) and counted the number of correct answers. In a second and third trial the same 24 tag clouds were shown to the participants again. In the second trial they had to answer the question “How beautiful is the picture?” and in the third trial the question was “How helpful was the picture regarding to your task in trial one?”. The answers were noted by the experimenter. For each of the eight conditions (combinations of algorithm and number of tags) we computed the mean value over all three items to get one single value per condition and person.

Results: Before computing the ANOVAs we tested the dependend variables against normal distribution using the Kolmogorov-Smirnov Z-Test. None of the variables did sig-



(a) reaction time (b) correctness (c) perceived beauty (d) perceived helpfulness

Figure 8. Influence of the algorithm and the number of tags on the dependent variables

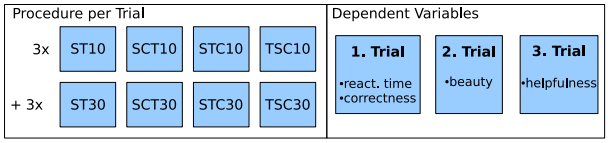


Figure 7. Procedure of the user evaluation.

nificantly differ from normal distribution and therefore the prerequisites for computing an ANOVA were fulfilled.

In order to answer the question whether the algorithm and the number of tags influence reaction time and correctness for identifying the most important words, we used a 2-factorial multivariate ANOVA for repeated measures on both factors. We found a significant main effect for algorithm ($F_{6,24} = 10.09; p < .01$), a significant main effect for number of tags ($F_{2,28} = 17.02; p < .01$) and a significant interaction of algorithm x number of tags ($F_{6,24} = 8.17; p < .01$).

Tag clouds with 10 tags led to a significant lower reaction time ($\bar{x} = 5800.88, s = 326.29$) than tag clouds with 30 tags ($\bar{x} = 6631.82, s = 506.55$). Algorithm TSC led to the lowest reaction time ($\bar{x} = 5836.51, s = 370.97$), see Table 3 for the according values. The effect of the interaction of algorithm and number of tags on reaction time is shown in Figure 8(a).

Tag clouds with 10 tags led to a significant higher correctness ($\bar{x} = 2.22, s = 0.12$) than tag clouds with 30 tags ($\bar{x} = 1.86, s = 0.10$). The main effect of algorithm can not be fully interpreted because of the significant interaction of algorithm x number of tags, see Figure 8(b). The condition leading to the highest correctness was algorithm ST with 10 tags ($\bar{x} = 2.39, s = 0.13$). Viewing the correctness, algorithm TSC seems to compensate the effect of the number of tags.

For answering the question whether the algorithm and the number of tags influence the perceived beauty of a tag cloud, we used a 2-factorial univariate ANOVA for repeated measures on both factors. We found a significant main effect of the number of tags ($F_{1,29} = 8.71; p < .5$), a tendency for the main effect algorithm ($F_{1.9,55.14} = 2.79; p < .10$) and a non-significant interaction ($F_{2.43,70.59} = 0.19; n.s.$). Tag clouds with 10 tags were perceived more beautiful ($\bar{x} = 4.47, s = 0.19$) than tags clouds with 30 tags ($\bar{x} = 3.60, s = 0.22$). Algorithm SCT tends to produce a more beautiful layout ($\bar{x} = 4.30, s = 0.21$). The highest rated beauty was obtained using algorithm SCT with 10 tags ($\bar{x} = 4.71, s = 1.17$), see Figure 8(c).

For answering the question whether the algorithm and the number of tags influence the perceived helpfulness of a tag cloud, we used a 2-factorial univariate ANOVA for repeated measures on both factors. We found a significant main effect of the number of tags ($F_{1,29} = 38.88; p < .01$), a tendency for the main effect algorithm ($F_{2.39,69.26} = 2.29; p < .10$) and a significant interaction ($F_{2.16,62.52} = 9.41; p < .01$). Tag clouds with 10 tags were considered more helpful ($\bar{x} = 5.28, s = 0.19$) than tag clouds with 30 tags ($\bar{x} = 3.75, s = 0.21$). The main effect algorithm could not be fully interpreted due to the interaction, see Figure 8(d). The condition leading to the highest perceived helpfulness was algorithm SCT with 10 tags ($\bar{x} = 5.31, s = 1.18$), algorithm TSC seems to compensate the effect of the number of tags.

Reaction Time [ms]		
Algorithm	$\bar{x}$	**s**
ST	6324.48	425.12
STC	6459.26	441.13
SCT	6245.13	421.86
TSC	5836.51	370.97

Table 3. Influence of the algorithm on reaction time [ms] (main effect).

Discussion: As for the technical evaluation we could not conclude a clear wining algorithm in the user evaluation. The influence of the different algorithms heavily depended on the number of tags. Across all four dependent variables (reaction time, correctness, beauty, helpfulness) we found better results for tag clouds with 10 tags compared to tag clouds with 30 tags. Regarding the objective measures (re-

action time and correctness), algorithm TSC outperforms the others in both, the 10 tags and the 30 tags condition. Algorithm TSC was considered the most helpful in the 30 tags conditions, however there was no clear difference between the four algorithms in the 10 tags case. From an aesthetic perspective algorithm TSC was the least satisfying one within both conditions (10 tags, 30 tags). Our assumption is that this is because of the huge proportion of cropped letters which might have disturbed the perceived symmetry of the picture, as can be seen in figure 5(d). The algorithm considered the most beautiful one in both conditions was algorithm SCT.

5. Conclusion/Discussion

With this investigation we tried to examine various technical and usability issues for the four proposed algorithms ST, STC, SCT and TSC. More specifically with the user evaluation the question should be answered which of the algorithms is the most convenient for identifying the three most relevant words in a tag cloud. The technical evaluation showed that algorithm SCT and TSC both are satisfying for almost all parameters and do not come along with major drawbacks. TSC also outperformed the others in three of four usability parameters, when the participants were asked to name the three most important words. Its major drawback was the not so high rated beauty. If correctness and reaction time play an important role for the application of a tag cloud algorithm TSC is the one to choose. For applications with a strong focus on aesthetics one should think of using algorithm SCT.

6. Future Work

A second usability study is scheduled for the near future. In the presented study we compared our four algorithms among each other. A further usability study will then compare our algorithms with other state-of-the art algorithms.

With regard to the algorithms, hyphenation will be implemented instead of the character-wise clipping approach currently used. This should increase legibility of clipped tags. Appropriate experiments will be included in the second usability study.

The reduction of algorithmic complexity will also be addressed. A dynamic combination of the shift, truncate and scale steps described in the algorithmic section is foreseen. Such a combination could be based on heuristics or on a learning approach. This measures will hopefully enable application of the algorithms in large-scale scenarios.

The algorithms described in this publication are currently being applied to several real-world scenarios. This process will hopefully yield valuable insights aiding us in prioritization of our research agenda.

7. Acknowledgment

The Know-Center is funded within the Austrian COMET Program - Competence Centers for Excellent Technologies - under the auspices of the Austrian Ministry of Transport, Innovation and Technology, the Austrian Ministry of Economics and Labor and by the State of Styria. COMET is managed by the Austrian Research Promotion Agency FFG.

References

[1] K. Bielenberg and M. Zacher. Groups in social software: Utilizing tagging to integrate individual contexts for social navigation. Master's thesis, Program of Digital Media, University of Bremen, 2005.

[2] M. Granitzer, W. Kienreich, V. Sabol, K. Andrews, and W. Klieber. Evaluating a system for interactive exploration of large, hierarchically structured document repositories. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'04)*, pages 127–134, Washington, DC, USA, 2004. IEEE Computer Society.

[3] M. Halvey and M. T. Keane. An assessment of tag presentation techniques. In *Proceedings of the 16th International World Wide Web Conference (WWW2007)*, Banff, Alberta, Canada, May 2007.

[4] K. Hoffman. In search of the perfect tag cloud. Whitepaper, August 2006.

[5] O. Kaser and D. Lemire. Tag-cloud drawing: Algorithms for cloud visualization. In *Proceedings of the Workshop on Tagging and Metadata for Social Information Organization (WWW2007)*, Banff, Alberta, Canada, May 2007.

[6] B. Y. L. Kuo, T. Hentrich, B. M. Good, and M. D. Wilkinson. Tag clouds for summarizing web search results. In *Proceedings of the 16th International World Wide Web Conference (WWW2007)*, pages 1203–1204, New York, NY, USA, 2007. ACM Press.

[7] A. W. Rivadeneira, D. M. Gruen, M. J. Muller, and D. R. Millen. Getting our head in the clouds: toward evaluation studies of tagclouds. In *Proceedings of the Computer/Human Interaction Conference (CHI2007)*, pages 995–998, New York, NY, USA, 2007. ACM Press.

[8] B. Shaw. Utilizing folksonomy: Similarity metadata from the del.icio.us system. Project Proposal, December 2005.

[9] M. Stefaner. Visual tools for the socio-semantic web. Master's thesis, University of Applied Sciences Potsdam, June 2007.

[10] B. Stein and F. Benteler. On the generalized box-drawing of trees, survey and new technology. In *Proceedings of International Conference on Knowledge Management (I-KNOW)*, pages 416–423, September 2007.