Tag-Cloud Drawing: Algorithms for Cloud Visualization

Authors: Owen Kaser and Daniel Lemire

Jason Ye
University of Virginia
What is a tag-cloud?
Characteristics of a tag-cloud

Flickr tag-cloud

- Visual representation
- Navigator
- Histogram

https://www.flickr.com/photos/tags/
Two types of tag-clouds

Inline Text

Popular Tags
AdSense Affiliate marketing Author rank Authorrank Authorship cCTLD CMS Content Management System Content writing service DC DNS Domain Name System Domains Drupal Duplicate content Extensions Facebook Google Google+ communities Google AdSense Google Analytics Google Pagerank Google Plus Google PR gTLD Joomla Joomla 3 Joomla extensions Link building Linkbuilding Magic quotes Make money on website Name server Outsourced content writing Pagerank Plugins Search Engine Optimization SEO Social networks Spam TLD Twitter Webhosting Webhosts Wordpress

Arbitrary Placement


Kaser and Lemire
## Two types of tag-clouds

<table>
<thead>
<tr>
<th>Inline Text</th>
<th>Arbitrary Placement</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Order of text has no semantic meaning</td>
<td>● Tags can be reordered, placement depends on relationships</td>
</tr>
<tr>
<td>● Paragraph made exclusively from inline elements (span, em, i)</td>
<td>● HTML nested tables</td>
</tr>
<tr>
<td>● Excessive clumps of white space</td>
<td>● Wasted space</td>
</tr>
</tbody>
</table>
Algorithm 1: Break up an ordered list of tags

- **Greedy Algorithm**: O(n)
- **Knuth-Plass Algorithm**: O(n^2)
  - Compute *badness* of fit
  - Minimize sum of squares of each line’s badness
  - Reconstruct optimal badness recursively
Algorithm 2: Reorders tags to decrease badness

- NP-hard Strip Packing Problem (SPP)
  - Use dynamic programming to place tags optimally while keeping the best solution
- First Fit Decreasing Height, Weight (FFDHW)
SPP Approximation Algorithms

http://cgi.csc.liv.ac.uk/~epa/ffdh.GIF

http://cgi.csc.liv.ac.uk/~epa/nfdh.GIF
Results: Inline text

\( l_1 \) norm: the sum of all the “badness”
- FFDH and FFDHW is much better than dynamic programming

\( l_2 \) norm: the sum of all the squares of “badness”
- FFDH and FFDHW only slightly better, dynamic programming is a competitive solution
Arbitrary Placement

Algorithm: Electronic Design Automation (EDA)

1) Min-cut Placement: NP-hard
   ○ Bipartitioning into “right” and “left”
Arbitrary Placement

2) Slicing floorplans
   ○ Recursive bipartitioning represented by slicing tree

(a) Slicing tree. Numbers next to nodes relate to areas in the slicing floorplan.

(b) Slicing floorplan

Kaser and Lemire
Arbitrary Placement

3) Nested Tables

- Every internal node in tree is a 2-element table

```
<table>
  <tr>
    <td>father</td>
    <td>vivian</td>
  </tr>
</table>
```

Kaser and Lemire
Results: Arbitrary Placement

- Greedy method used 2-17% less area than min-cut
- However, min-cut approach much better for semantic proximity