FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Dynamic Word Clouds

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Martin Seyfert
Matrikelnummer 0726731

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ivan Viola, Associate Prof. Dipl.-Ing. Dr.techn.

Wien, 1. Jänner 2001

Martin Seyfert	Ivan Viola

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Dynamic Word Clouds

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Martin Seyfert
Registration Number 0726731

to the Faculty of Informatics

at the TU Wien

Advisor: Ivan Viola, Associate Prof. Dipl.-Ing. Dr.techn.

Vienna, 1st January, 2001

Martin Seyfert	Ivan Viola

Erklärung zur Verfassung der Arbeit

Martin Seyfert
St. Bartholomäusplatz 1, 1170 Wien

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, 1. Jänner 2001

Martin Seyfert

Acknowledgements

I would like to express my gratitude to my advisor Ivan Viola, not only for his friendly support throughout the work on this thesis but also for his encouragement to submit a paper based on it to SCCG 2017. Experiencing an international scientific conference firsthand was an incredibly valuable opportunity.

Further, I would like to thank the Visualization Group at TUWien, in particular Dr. Manuela Waldner and Meister Eduard Gröller, for their feedback and support.

Finally, I want to thank my family and friends, especially my parents and my girlfriend, Marisa, for standing by me throughout the more stressful parts of this work.

This project has been funded by the Vienna Science and Technology Fund (WWTF) through project VRG11-010 and supported by EC Marie Curie Career Integration Grant through project PCIG13-GA-2013-618680.

Conference Paper

A paper based on this thesis was submitted and accepted for the Spring Conference on Computer Graphics (SCCG) 2017. The accompanying talk at the conference won the audience award for "best presentation".

Kurzfassung

Die Verwendung von dynamischen Word Clouds für zeitlich veränderliche Daten ist ein noch wenig erforschtes Feld. Das Ziel unseres Ansatzes ist, eine neuartige Art der Generierung von gleichmäßig animierten Word Clouds zu zeigen, um Änderungen in der Wordfrequenz mittels Schriftgröße darzustellen. Im Gegensatz zu existierenden Methoden wird ein kompaktes Layout, inspiriert von dem populären Word Cloud Generierungs-Werkzeug Wordle, während der Animation beibehalten und mittels Web-Technologien implementiert. Veränderungen in der Wortgröße werden außerdem mittels Farbe und Rotation illustriert.

Abstract

Using word clouds to visualize dynamic time-varying data is a field still under-explored. The goal of our approach is to provide a novel way of generating smoothly animated word clouds to show changes in word frequency via font size. Unlike existing methods, a compact layout, inspired by the popular word cloud generation tool Wordle, is preserved during animation and implemented using web technologies. Word size changes in time are also illustrated via color and word rotation.

Contents

CHAPTER 1

Introduction

The method of visualizing word frequency via font size goes back many years. Stanley Milgram already did early experiments using font size to represent information in the 1970s, illustrating the popularity of various landmarks in a map of the city of Paris via text [Mil76]. It was the rise of so-called "tag clouds" in Web 2.0 design, however, which brought the visualization method into the mainstream [VW08]. In 2002, photo-sharing site Flickr started visualizing tags people used on their photographs by sorting them by popularity and showing more frequent ones in bigger font sizes. This was done in a simple paragraph of words being sorted alphabetically. The goal was to provide a quick overview.

The look most frequently associated with the term "word cloud" today goes back to a layout algorithm Jonathan Feinberg had created for the social bookmarking application "dogear" for IBM. He later implemented this method in a freely available website called Wordle[1], which popularized a new way of displaying words in a cloud layout [SI10]. Wordle offers automatic text analysis for word frequency (which led to a shift from the term "tag cloud" to "word cloud"), places words freely on a canvas instead of within a paragraph and takes the white space between individual glyph shapes into account to create a more compact layout. Unlike simpler tag cloud implementations before, the goal was also for the result to be aesthetically pleasing. Wordle allows a variety of merely decorative options, such as color, word rotation or font choice, which do not represent any actual information.

Since word clouds do not allow exact measurement or comparison of the underlying data, their main purpose is to provide a quick overview over a more in-depth subject. While other visualization methods (as an obvious example, a simple, vertical list), can provide as good or better results in forming an overall impression [RGMM07], word clouds still have been found to be a useful supplementary research tool [ML10]. With

[1] http://www.wordle.net/

time as an additional dimension, the appearance of the result also changes to a point where it significantly differs from any static representation, which suggests that findings based on static word clouds are no longer directly comparable. It is not obvious whether dynamic word clouds are more or less successful at visualizing the underlying data than static word clouds.

It is also worth noting that the original motivation for Wordle had a strong aesthetic component to it, which was powerful enough for it to quickly spread in popularity among users who do not work with text analysis on a professional or scientific level [SI10]. It can be argued to act as an "ice breaker" of sorts, getting people to notice interesting patterns in word frequency even where they had no intention to actively look for them. The animated nature of a dynamic word cloud can serve as an additional source of attention, getting users to form an interest in the subject via a quick overview and potentially inspire later, more in-depth insights. A possible use could be a widget that accompanies an article on a website.

While the usefulness of static word clouds [RGMM07, ML10] and further experiments in user interaction [KLKS10, JLS15] have been explored in the past, literature on visualizing data with changing word frequency over time—via dynamic, animated word clouds—is surprisingly sparse. Further, the focus of existing methods lies with simpler word collision detection that does not take into account the more compact layout made possible in Wordle-inspired methods.

This thesis proposes a novel way of creating dynamic word clouds for visualizing time-varying data. Our approach takes into account the shift in size changes at all keyframes simultaneously and uses them to arrange words more efficiently for a smooth animation of transitions. Also, a Wordle-like placement algorithm assures a compact layout.

Additional typographic options besides font size are explored as a source of visual information rather than simply aesthetic choices. A color gradient as well as word rotation are used to emphasize changes in word size. A custom font style can also be specified per word to categorize words in the dataset.

A goal was also to test the feasibility of implementing dynamic word clouds using web technologies like HTML5, SVG and JavaScript, especially regarding generation time.

CHAPTER 2

Related Work

2.1 Static Word Clouds

The modern, space-efficient layout of word clouds is primarily based on Wordle, by Jonathan Feinberg. He describes his approach in detail in Chapter 3 of the book "Beautiful Visualization: Looking at Data through the Eyes of Experts" [SI10]. After simply determining the font sizes based on relative word count, each word is placed in a random position. Collisions are done for individual glyph shapes using hierarchical bounding boxes for optimization. When a collision is found, the word is gradually moved outward along a spiral path to find the closest free placement position.

There have been several attempts to improve static word cloud layouts and to account for additional requirements such as position and overall shape. "Rolled-out Wordles" [SSS+12] offer an improved word placement strategy via an additional sorting step before overlap removal. The result is a more even layout of the overall Wordle shape.

Another possible feature is the preservation of spatial information, like the location of cities tagged in a map in "Geo Word Clouds" [BCL+16]. It is notable that finding a satisfying layout using such complex requirements can have a significant impact on performance (in the case of Geo Word Clouds, the algorithm can take a full hour to place 126 location-constrained tags in the shape of Great Britain). Of course, simpler, bounding box based collisions such as the placement strategy used in WordBridge [KKEE11], which simply moves words along a vertical and horizontal axis, are also an option and lead to larger amounts of white space with the added benefit of faster computation times.

2.2 Dynamic Word Clouds

There have been several attempts at providing more interactive and flexible manipulation of word clouds based on Wordle. ManiWordle [KLKS10] allows moving and rotating

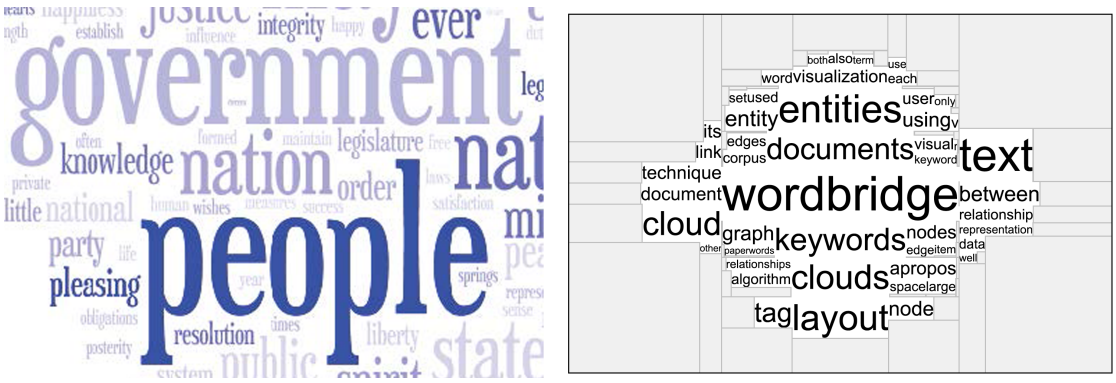

Figure 2.1: A comparison between a layout generated via bitmap-based collision (Wordle [SI10], left) and bounding-box collision (Wordbridge [KKEE11], right). For words with extreme size changes and fonts with long ascenders and descenders (such as the letters "p" and "l" in the word "people"), the available space can be filled more efficiently using the more exact, bitmap-based method used by Wordle.

individual words in a word cloud via drag-and-drop, which allows the manual refinement of the generated layout. WordlePlus [JLS15] provides a similar set of tools but adds resizing, adding and grouping of words as well as an option to animate the result by making words pop in one after the other. The end result, however, is still a static word cloud.

One way of using word clouds for visualizing trends in time-varying (or otherwise dynamic) data is to combine multiple visualization techniques. Parallel Tag Clouds [CVW09] combine parallel coordinates and a gradient line to illustrate changes in word frequency while also using this data for the font size of words arranged in columns. SparkClouds [LRKC10] are simple tag clouds with a sparkline underneath each word. Tag frequency, of course, can also be visualized without using such complex layouts as illustrated by "Cloudalicious" [Rus06], which simply displayed tag frequency changes over time as a graph.

Cui et al. [CWL+10] have proposed a method for illustrating time-varying word cloud data while preserving the overall layout of a graph connecting nearby words. Simple bounding boxes are used for detecting collisions and repulsive, spring and attractive forces push words until collisions are resolved. Colored tags are used to illustrate appearing, disappearing and unique words per keyframe.

A more complex approach for "morphable word clouds" by Chi et al. [CLC+15] uses interpolated boundary shapes and constrained rigid body dynamics to deal with collisions, which also allow words to be rotated to better fit within the layout. Word shapes are approximated via a convex polyhedral. This method can require manual intervention and tweaking to prevent words blocking each other in collision.

WordSwarm [Kan14] uses the real-time 2D physics engine Box2D to apply a gravitational force to each word's bounding box and gradually move them to the center of the

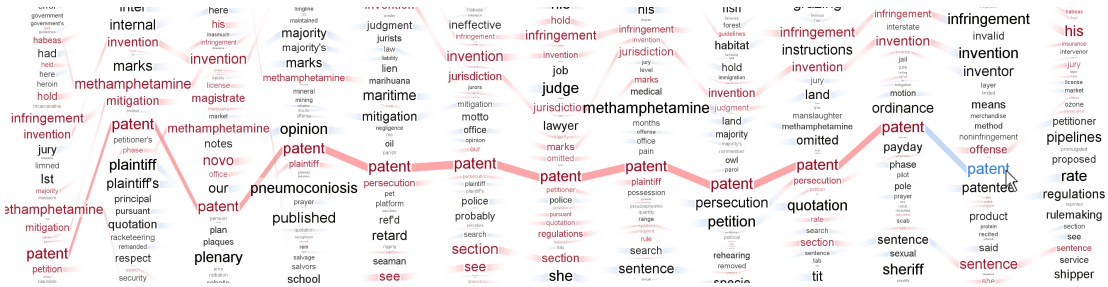

Figure 2.2: Parallel Tag Clouds, as proposed by Collins et al. [CVW09], are an example of combining multiple visualization methods to illustrate changes in word frequency. Tag clouds are used in conjunction with parallel coordinates and a gradient line.

screen. The layout takes time to reach a stable form and overlaps can occur because of compromises in the real-time optimized physics engine.

Existing approaches to dynamic word cloud generation either introduce additional visualization methods or use rather simple collisions (bounding boxes) and sparse layouts. This can result in wasted space, jittery animation or other collision artifacts such as overlapping words which can, in some cases, even require manual tweaking. It is our goal to use some techniques previously only attempted in static word clouds as well as further optimizations in how to handle time-varying data to overcome these compromises.

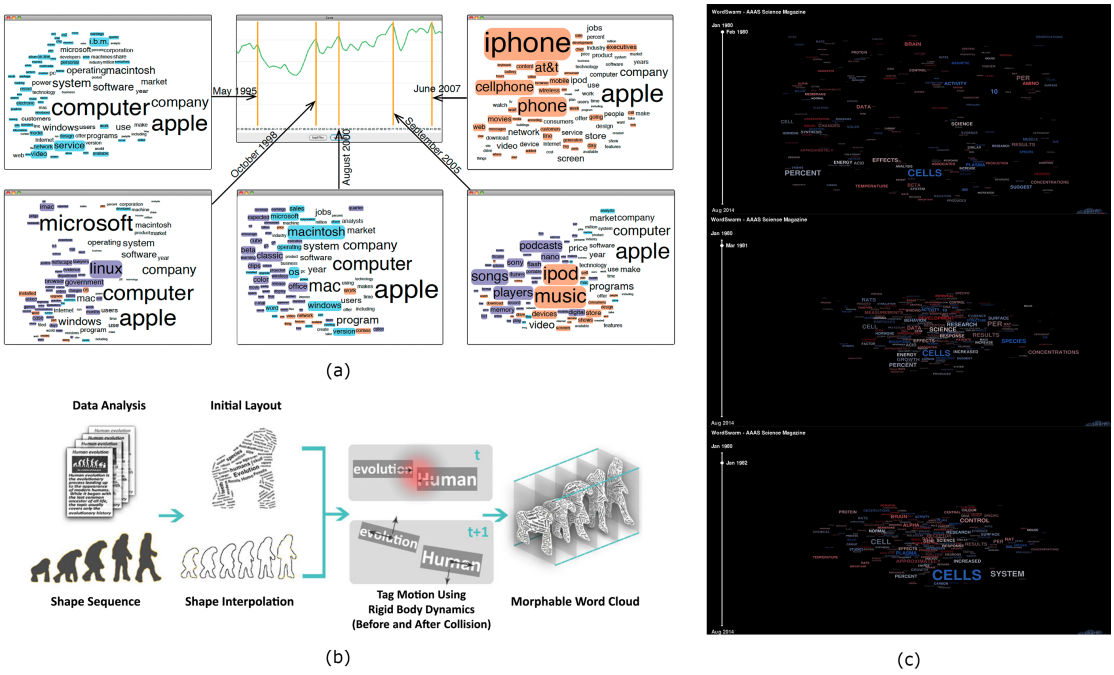

Figure 2.3: Three examples of existing approaches to dynamic word cloud visualization. (a) Cui et al. [CWL+10] demonstrate a method using a force-directed model. (b) Chi et al. [CLC+15] propose "morphable word clouds" using complex rigid body dynamics and shape constraints but can require manual tweaking to resolve words blocking each other in transitions. (c) WordSwarm [Kan14] uses a 2D physics engine that gradually moves words towards a center via simulated gravity.

CHAPTER 3

Methodology

The basic idea of this work is to use an exact, Wordle-like placement strategy where all collisions over multiple key-frames are considered simultaneously. Time can be thought of as an additional dimension along a time axis.

Bitmap-level collisions are used to create a compact layout that avoids the distracting amounts of white space that can be the result of great size differences between words as well as glyphs with significant ascenders or descenders. The goal is to create a concise and aesthetically pleasing overall shape. To avoid unnecessary checks for overlaps and to keep size changes balanced over the whole time axis, a special algorithm is used to pick the order at which new words are added. Instead of Wordle's initial random placement, words are always placed using a spiral placement strategy starting from the center to achieve an optimal layout. The size change between different keyframes is also used for coloring and word rotation to further illustrate changes.

3.1 Choosing the word placement order

The first step in word placement is ranking words for their placement order. A simple approach of picking the words with the highest total size (over all keyframes) can create a satisfying layout in which more prominent words are closer to the center. A more advanced placement strategy (Fig. 3.1) can help handling word clouds with drastic size changes. For that, words are sorted by their average size difference (as measured by the diagonal of their bounding boxes) over all keyframes. This allows to pick the most temporarily stable in size word as a starting point. The word is placed in the middle of what Wordle considers the "playing field" [SI10], an area of sufficient size to hold the combined area of all words.

After an initial word has been placed, the next word has to be chosen. For this, the average absolute difference in bounding box diagonals between the already placed and

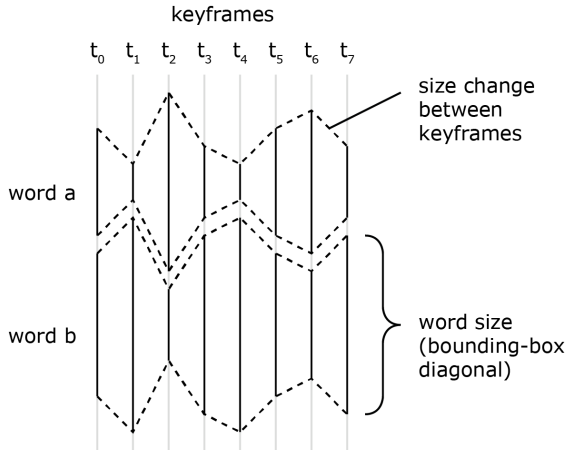

Figure 3.1: An example of an ideal match between two bounding boxes. The change in size (as measured by its bounding-box diagonal) between word a and word b at each keyframe adds up to zero while the combined size stays constant.

the new word at all keyframes is considered and the new word with the minimal change in size difference is chosen. When the already placed word is growing, the new word should be shrinking and vice versa. This is continued until all words are placed, always comparing the bounding box of the next word to the total bounding box enclosing already placed words.

The benefit of this approach is that during collision detection, size changes from one keyframe to the next likely compensate each other. This means that if a non-overlapping position is found in the first keyframe, it likely also fits in all other keyframes, despite the size changes.

3.2 Resolving collisions

For resolving collisions, we choose a spiral path placement strategy as it is used in Wordle. Collisions also take into account the exact glyph shape of letters in each word rather than simple bounding boxes. This allows more efficient and compact layouts, especially when there is a big contrast in word sizes and fonts with long ascenders or descenders are used.

For each considered position, collisions are checked in all keyframes. Once a collision is detected in any keyframe, the position is rejected for all keyframes as illustrated in Fig. 3.2. The word is moved along a spiral path going outward from its initial placement position until there is no collision found in any keyframe (see Algorithm 3.1). A simple rectangular spiral pattern (Fig. 3.3) is used. While it can cause the overall cloud layout to look slightly rectangular, it is good enough to generate a centered layout. An alternative would be a rounder path such as one along an Archimedean spiral.

Figure 3.2: Collisions have to be tested at all keyframes. In example (a), the new word "Gamma" doesn't overlap in keyframes t1 and t3 but does so in t2. As a result, the position is rejected and the word is moved to a next position. In example (b), the word has been moved slightly and no longer overlaps in any keyframe. It is notable that this has caused slightly more white space between the words, especially in t1 and t3. This is solved by moving the word towards the center of the already placed words which is done in a final step (c).

Algorithm 3.1: Resolve collisions

```
for each word do
    place word at center;
    while collision found do
        move word along outward spiral;
    end
    while no collision found and distance < threshold do
        move word towards center;
    end
end
```

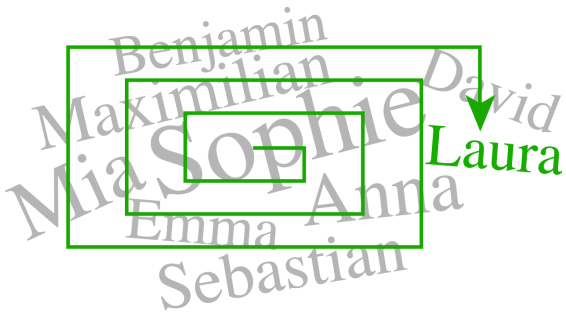

Figure 3.3: In order to find the closest available position to the center of the word cloud, the word is moved outward along a rectangular spiral path until no collisions are found.

The disadvantage of this method is that longer calculation times are necessary than in randomized placement. The best way to solve this problem depends on the implementation platform but checking spline-based glyph shapes for collision is certainly too expensive. A simple approach is rendering font shapes into bitmaps and using those for collisions. The bitmap resolution has to be chosen based on the desired exactness of the collision. A minimum resolution (or, respectively, a minimum word size) to handle the smallest words in the cloud should be considered. Further, the distance a word is moved along the spiral path in each iteration can be increased to get to potentially valid placement positions more quickly (for a bitmap representation, the word could for example be moved 5 pixels upon finding a collision, rather than to the directly neighboring pixel).

In a last step, the newly placed word is moved linearly towards the center of the combined bounding box of all previously placed words, until it collides. This is done separately in all keyframes. The goal is making the layout even slightly more compact. A threshold value is used for a maximal movement distance to avoid too rapid position changes between keyframes. The complete result is then centered in the playing field before the next word is going to be placed, to keep the word cloud from wandering towards the edge.

A problem that can occur with the time data, is missing or zero values for font size, for example when a word only starts appearing at a later keyframe or disappears from the word cloud completely. A simple solution is to convert words that are zero sized or otherwise so small that they might not be visible to a minimal size that can be used for collision but set them to not be rendered. That way some space is considered for animations without significantly hurting the overall layout. The bitmap resolution also has to be considered to avoid words shrinking below a reasonable size (for example, a one pixel high word might not produce a bitmap that is suitable to represent collisions or, in extreme cases, even be empty).

3.3 Color, rotation and other typography

Wordle uses randomized word colors as an aesthetic choice. The only real concern is contrast to the background color to ensure readability. Given the added complexity of time data, however, color can be used as an additional source of information. For example, Cui et al. [CWL+10] use colored labels to tag appearing, disappearing and unique words. Besides making changes more visible during animation, using color in this way also allows the user to see trends in a static frame.

Our approach uses the derivative in size change for color intensity with a certain threshold for maximum change. The color and threshold can be chosen by the user. Possible choices could include green for growing and red for shrinking words and words growing to twice or half their previous size as a threshold for maximum color intensity of a gradient from a black base color.

Word rotation is another option in Wordle to add visual interest and is widely used in word cloud generation. Rotation can either be chosen freely (with certain constraints like keeping words from being rotated to an upside-down orientation) or from a fixed set of angles (for example, 0°, 90°and -90°).

In our case, word rotation can also be used to further illustrate changes in size since the last keyframe. Shrinking words are rotated clockwise to make them point downwards in reading direction, words growing in size are rotated counter-clockwise. A constraint is set to avoid too extreme rotations (for example, capped at 30°and -30°).

It is tempting to use additional typography to further illustrate aspects of the word cloud, such as bold or italic font faces for the aforementioned changes, but the additional fonts should also not overload the visuals, which could lead to reduced readability. One reasonable option would be to allow for font changes that require editing the input data. While using color or rotation might require some general tweaking of threshold values, it is otherwise automated. Additional meta data could help illustrate input values but would have to be done manually by the user for the entire dataset. For example, in a word cloud illustrating the popularity of male and female given names, male names can be set to display a different font than female ones. Such options could be set by the user in the source database files.

3.4 Frame Interpolation

While methods described so far guarantee words not intersecting at the provided keyframes as defined in the input data, the interpolation of position, size and rotation used to animate in-between states can cause unwanted overlaps. This effect can range from subtle to rather noticeable (Figure 3.4). These overlaps are especially unpredictable when word rotation is used.

A solution to this problem is calculating collisions for a set amount of in-between frames. The interpolated frames are treated like keyframes for collision and are thus fully

Figure 3.4: A case of subtle collisions that can occur during animation because of interpolated positions. Although both keyframes of the word "Delta" do not overlap, in-between frames do.

considered in the steps described in Fig 3.2. This further increases calculation time and therefore should be considered a luxury refinement. A single frame of interpolation can already serve as a compromise.

CHAPTER 4

Results

4.1 Target Platform

The goal of the described methodology is allowing an implementation in a modern, real-world environment: a web-based, in-browser solution not depending on plugins. The code is available on GitHub[1].

4.1.1 HTML5 and SVG

The resulting word cloud is displayed as a group of SVG (Scalable Vector Graphics) text objects. This has several advantages. SVG is an XML-based graphics format which allows simple manipulation (for example, for specifying font size, position, color and rotation) within a website. It also is a vector-based representation which means it is resolution-independent. SVG objects are supported by all modern browsers (the most significant exception being Internet Explorer 8 and below) and thus should work for a reasonable subset of potential users.

Vector-based collision detection, however, is too expensive to calculate, therefore collisions are done on a bitmap level. To make this possible, the SVG text has to be rasterized and written to an HTML5 canvas, the result of which can be read as a binary bitmap. Only a bitmask has to be considered so all pixel color values above a certain threshold are considered "filled", all below as "empty". A text outline can also be rendered during rasterization to act as a simple barrier for collision, which avoids word shapes sticking too close to each other.

[1] https://github.com/martinsft/wdc

Figure 4.1: Algorithm 3.1 applied to an example data set, placing the word "Felix" in a word cloud. After a collision is detected, the word is gradually moved along an outside rectangular spiral path. After the first free position is found, the word is moved towards the center.

4.1.2 JavaScript

To avoid the need for external plugins (which are associated with additional install-hassles and potential security risks), all code is done in-browser using JavaScript. The advances in browser JavaScript engines in recent years, fueled by the rise of Google's Chrome browser, have made the language a reasonable choice even for performance-intensive tasks [Hoe12]. However, compared to native implementations, JavaScript implementations can still be slower by a factor 2 or more. The advantage of not requiring additional downloads should still make it a reasonable compromise.

Compatibility is also a concern for a rapidly changing web-environment. It is questionable whether excessive backwards-compatibility or future-proofing forms is a reasonable requirement. The SVG objects used to render the text are not even compatible with versions of Internet Explore before IE9. As a compromise, only features as defined in ECMAScript 5 are used.

4.1.3 D3

D3 [Bos11] is a data visualization library implemented in JavaScript. It is based on SVG, HTML5 and CSS. Version 4 is used for the implementation.

D3 provides a wide variety of visualization tools while giving developers control over individual elements. It allows the binding of arbitrary data to the DOM (Document Object Model) of a website and supports a variety of animation and interaction methods. In our work, D3 is mostly used to interpolate and animate transitions between keyframes. It also provides interactive elements such as sliders.

4.2 Optimization

Performance is a big concern considering the comparably slow nature of JavaScript. While many browsers' JavaScript engines are now reasonably optimized, delays and

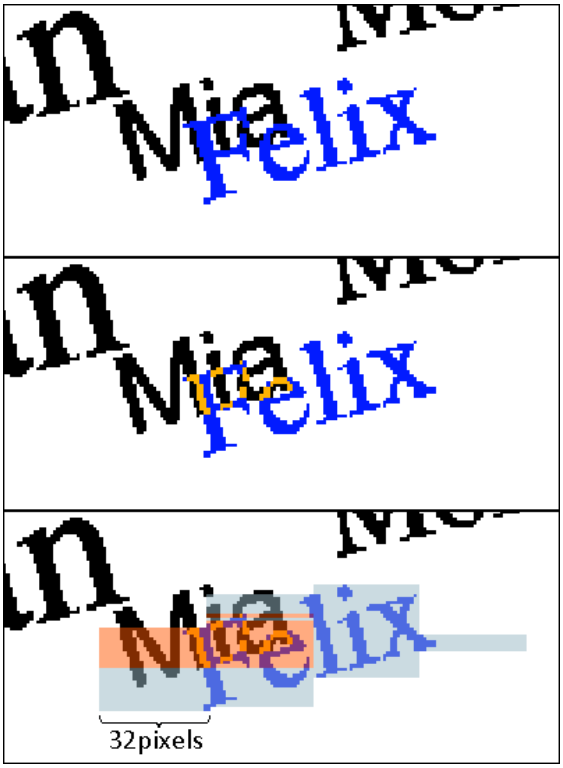

Figure 4.2: For collision, the SVG text shape is rendered to an HTML5 canvas. The resulting bitmaps are then stored as a raster of 32 pixel wide blocks, represented by one 32-bit integer per block. A newly placed bitmap ("Felix") is compared to the bitmap of already placed words by performing a simple binary operation on overlapping pixel blocks. The overlapping parts are shown in orange.

inefficiencies still can be a problem. One possible optimization used in Wordle is using hierarchical bounding boxes. However, an existing implementation for static word clouds using D3 by Jason Davis [Dav12] suggests a faster option using 32-bit integers.

To make bitwise operations possible, the canvas area has to be treated as a grid of one pixel high and 32 pixels wide blocks. If the desired area isn't a multiple of 32, additional columns of pixels are added.

After a new word is rendered onto the HTML5 canvas, the canvas image data is read and converted to 1-bit pixel values and the SVG text object's bounding box is used to define the bitmap's borders. Instead of individual pixel values, the bitmap data is converted to 1x32 pixel blocks aligned with those of the overall canvas grid.

Moving bitmaps is now a matter of left- and right-shifting the 32-bit integers representing 1x32 pixel blocks and changing the bitmap's coordinates. Simple AND and OR operations can then be used to efficiently compare the contents of two bitmaps (Fig. 4.2).

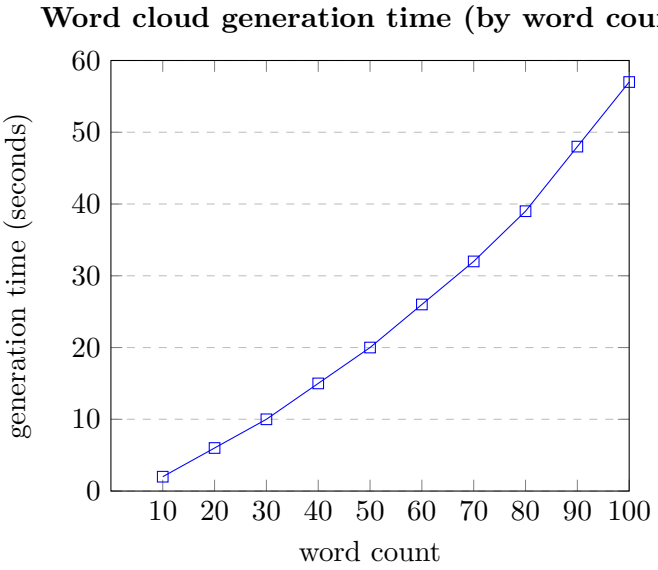

Figure 4.3: Word cloud generation time for a dataset of the most popular given names in Vienna between 2006 and 2014 (9 keyframes, no interpolation). While generation time grows as more words are used, the new words added are also smaller in size and thus easier to place, which keeps generation time almost linear within realistic word counts.

Because of certain inefficiencies in how JavaScript handles 32-bit integers (a conversion from floating point values is necessary), the performance gain isn't as drastic as handling 32 pixel values at once might suggest. It is possible, however, to speed up the process by roughly a factor of 5 compared to checking all pixel values individually. The final performance also depends on the browser used.

4.3 Data

While not a focus of this work, retrieving data is an important and often rather straightforward part of word cloud generation. Wordle [SI10] uses a simple method that starts with a large amount of text as an input. Words are separated by spaces and punctuation. Stop words such as "the", "it" and "and" are removed since they are of little interest to the user. Of course, different languages require their own lists of "stop words". The resulting words are simply weighted by their frequency. Other sources for word cloud generation can of course be data collected in a database or Excel file. The original use for tag clouds relied exclusively on data provided in such an easily retrievable fashion.

Handling a time-series of word frequencies is a little more complex as it requires separate data from multiple points in time. Our input must be pre-formatted as a comma-separated values list (CSV) which already has word size entries for each desired keyframe. Keyframes also have a label (for example, the year) which will be used to label the ticks

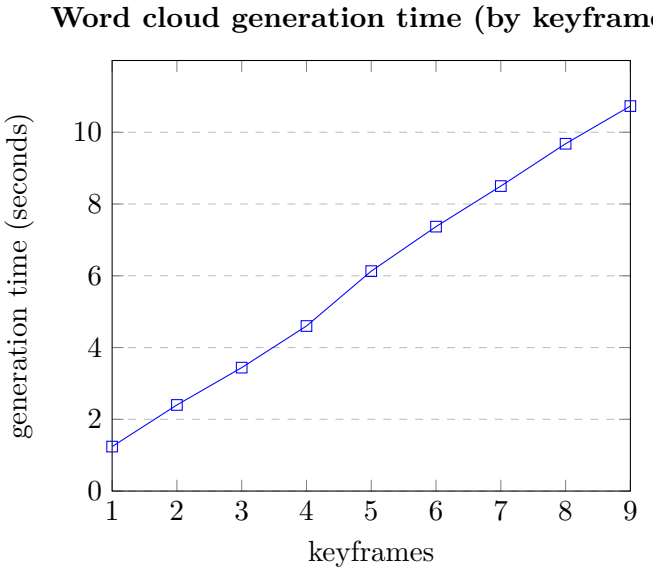

Figure 4.4: Word cloud generation time for a dataset of the 30 most popular given names in Vienna between 2006 and 2014 with different amounts of keyframes (1 to 9).

of a time axis in the interface. Additionally, a font style can be specified per word.

For testing, a simple data set containing the 30 most popular given names in Vienna between the years 2006 and 2014 was created, based on data from the Austrian government[2]. The preformating of the data was simply performed in Excel. The input was saved in the form of a CSV file. Different font families (Arial for female, Times for male) are used to indicate gender. The first five entries of the dataset look as follows in their CSV representation:

```
word,       2006,2007,2008,2009,2010,2011,2012,2013,2014, font
David,       122, 125, 119, 102, 169, 162, 151, 200, 167,times
Maximilian, 118, 120, 120, 120, 146, 183, 126, 128, 148,times
Alexander,  113,  99,  98, 114, 140, 130, 126, 130, 136,times
Sophie,     100, 102, 100,  92, 109, 122, 107, 130, 132,arial
Anna,        99, 108,  88,  86, 130, 110, 133, 102, 133,arial
```

A continuous transition from red (shrinking) to green (growing) was used for color. Word angle (upward in reading direction for growing, downward for shrinking) was applied with a maximum of 30°. A slider at the bottom can be manually dragged to a wanted year. Pressing the space bar plays an automated animation. In either case, transitions are smoothly animated.

[2]https://www.data.gv.at

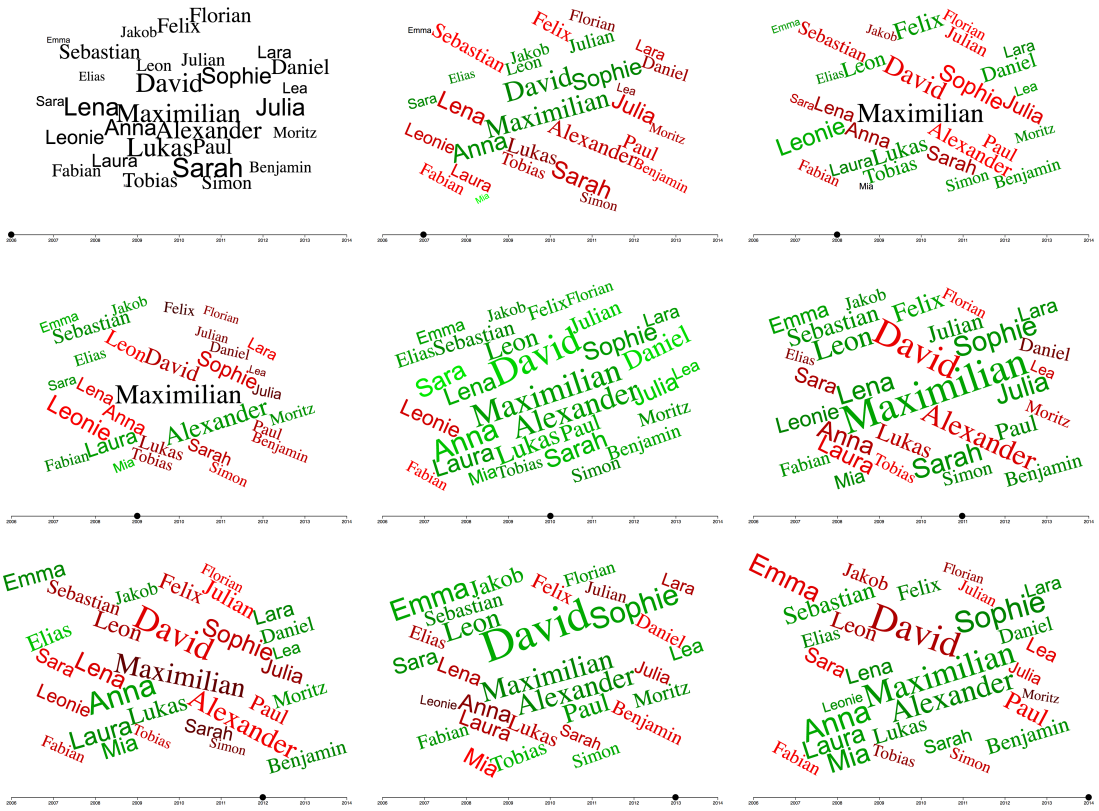

Figure 4.5: An example of a dynamic word cloud illustrating the most popular given names in Vienna between 2006 and 2014. Shades of red and clockwise rotation indicate shrinking, shades of green and counter-clockwise rotation growth. Font family (Arial for female, Times for male) indicates gender. The transitions are smoothly animated. Word overlaps can occasionally occur in interpolated frames.

In the resulting visualization (Fig. 4.5), popularity trends can be recognized. Since word size changes are calculated from the previous keyframe, the first keyframe does not contain such information. In 2010, for example, most names can be seen growing, with the exception of "Leonie" and "Fabian".

For an alternative use of dynamic word clouds, keywords from PacificVis 2016 were compared (Fig. 4.6). Only two key-frames are used: One with the 25 most submitted keywords and another with how often they were present in accepted papers. Keywords with acceptance ratios below 0.5 are shown in red and downward facing. The result demonstrates that even static frames can encode information about the changes between two data points. The length of the keyword-phrases poses a problem and shows a general limitation of word clouds in terms of overall layout.

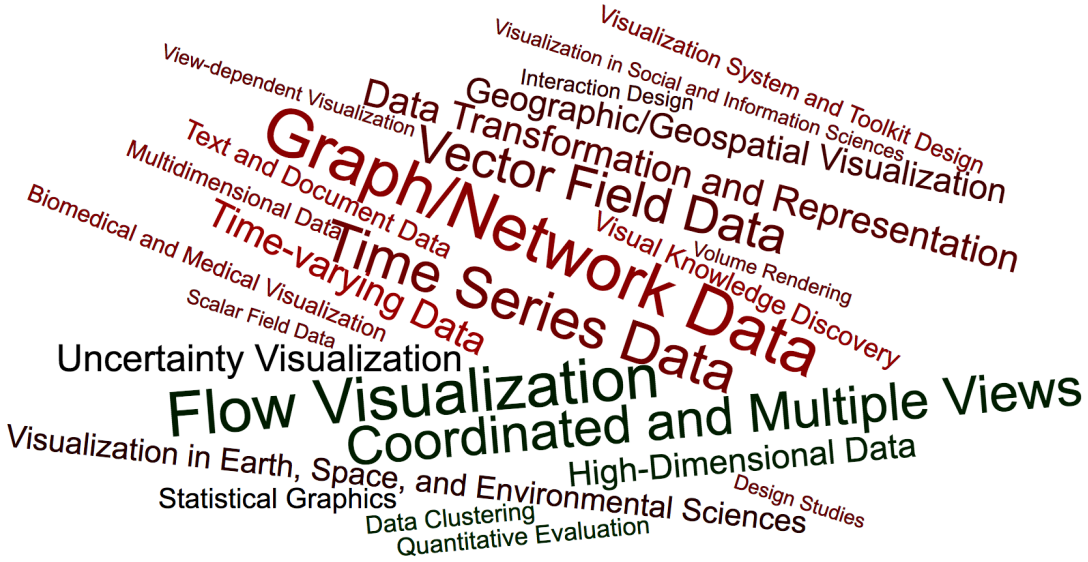

Figure 4.6: A visualization of submitted and accepted keywords from PacificVis 2016, weighted around a 0.5 ratio. For example, exactly half of the submitted papers about "Uncertainty Visualization" were accepted while significantly less than half (about 24%) of papers using the popular keyword "Graph/Network Data" were accepted.

CHAPTER 5

Evaluation

A small user study was conducted. 12 participants were given a link to a web implementation[1] of a dynamic word cloud using the dataset of given names. The UI consisted of a slider to move through different points in time, the space bar to start an automated animation and radio buttons to switch color and rotation on or off (Fig. 5.1). For convenience, the word positions were pre-calculated to avoid the significant generation time. There was no time limit given but most users spent between 2 and 5 minutes interacting with the dynamic word cloud before answering the questions. Users were asked to describe their general impression (positive and negative), what they learned about the dataset and which combination of visualization elements (color, rotation, size) they preferred.

Participants had a mostly positive impression, describing the nature of the visualization as inviting and fun, but noted that the rotation element appeared confusing and chaotic in movement. The distinction between male and female names through font type was criticized as being too subtle. On being asked for their preference, most participants (50%) mentioned "size and color only" as their favorite combination of visualization elements, followed by "size, color and rotation" with 33%. The reason given was that the addition of color makes the individual words easier to distinguish.

Similar to static word clouds, the word size put the focus on overall larger words such as David, Maximilian and Leon while exact measurements and comparisons between words of similar size were considered difficult. Users noticed trends such as the name Mia showing strong growth over the whole time period, the growth of all names except Leonie and Fabian in 2010 as well as certain popular names such as Maximilian, David and Sophie staying rather constant. One user mentioned that he found it striking that certain names gain popularity for only one or two years before going down again. Users could come up with examples for other data sets for which they could imagine the visualization

[1] https://martinsft.github.io/dwc_eval/

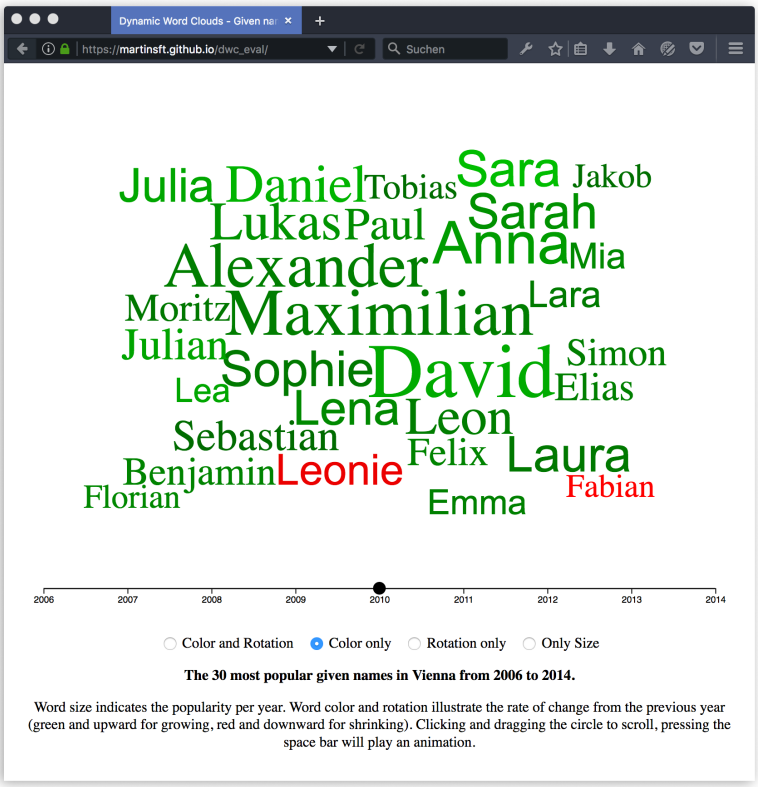

Figure 5.1: A screenshot of the web-implementation used for evaluation. The slider is set to display values from 2010, word rotation is switched off.

to be useful, ranging from marketing surveys, changes in bird population to software downloads.

In addition to the user study, the authors of reference papers were contacted for expert feedback. Ming-Te Chi, author of "Morphable Word Clouds for Time-Varying Text Data Visualization" [CLC+15] recommends avoiding using rotation and color at the same time or at least using rotation carefully because the amount of attributes changing might lead to "change blindness" which weakens the effectiveness of the visualization. He also suggests only using a single hue with different saturation instead of using different colors for growing and shrinking which could help users identify words and trace their trend.

CHAPTER 6

Discussion and Future Work

6.1 Discussion

The result preserves a compact layout usually only found in static word clouds and allows for smooth transitions along multiple keyframes. Overlaps are minimal but can occur in interpolated frames, especially during rotation.

One concern is performance, as even with several optimizations, large word clouds (100+ words) can result in generation times of over a minute, which might turn out to be a barrier in certain use cases. In our example dataset, adding more words did not increase performance as drastically as feared, however. This is probably due to the dataset being ordered by decreasing overall word frequency, as common for data used in word clouds. Smaller words create less bitmap data and are thus faster to collide. Similarly, while having significant impact on performance, having more keyframes does not increase calculation time too quickly (Fig. 4.4).

It is questionable whether word clouds with large amounts of words (100+) are even useful for analyzing time series data. The used library, D3, also struggles animating so many words in real-time, which is another technical barrier for realistic use.

Certain limitations found in static word clouds remain in dynamic word clouds. For example, making exact comparisons based on word size is considered difficult for users. Larger word sizes or using longer keyword phrases with spaces in between can make finding a suitable layout arrangement difficult.

The settings most useful to illustrate changes in different datasets for word rotation, color and categorization via font style can vary between uses. For example, rotation and color used in conjunction might be visually overwhelming in animation.

6.2 Conclusion

Expanding a Wordle-like layout strategy along a time axis is feasible, even though the bitmap-based collision detection causes significant word cloud generation times. For implementing such a method on the web, the generally short attention span of users has to be considered. Our JavaScript implementation takes several seconds, even for a reasonably sized data set. For presenting a dataset to a wider audience, it is currently necessary to pre-generate the word positions in the layout and re-use the result to avoid the performance bottleneck during collision detection. Further optimization seems necessary.

There are several ways of illustrating word change per keyframe typographically, without adding separate visualization methods that would go beyond what can be considered a "word cloud". Methods we considered include word color, orientation and font. By using these options (which in other word cloud generators, such as Wordle, are only used for aesthetic reasons) for size change information, trends can be highlighted and are noticeable even in a static frame of the result.

Evaluation shows users responding generally positive to using the visualization methods for finding trends in a dataset but certain settings (rotation, font style) are described as problematic and need further consideration and tweaking.

6.3 Future Work

Currently, this method of generating dynamic word clouds requires all time data to be available at the moment of generation. This makes it unsuitable for streaming data. Adding new words or unpredictable changes in word size would undo the benefits of the existing placement strategy and require an entirely different approach for resolving collisions. It would be interesting to explore whether the placement strategy could be expanded to handle streaming data efficiently. Chi et al. [CLC+15] describe a similar limitation of their approach and briefly mention a possible solution involving splitting up the streaming data into smaller sub-data.

Performance is another concern as collision detection over multiple keyframes for many words (50+) can become a major bottleneck, taking up several seconds up to a minute on a modern browser. More efficient collision methods could improve the work flow and allow users to see results more quickly. One possible way of achieving this would be to give options for using simpler collision methods such as bounding boxes only, however this would undo the work done on improving the layout and balance of white space. A more ideal solution would lie in implementing more efficient, bitmap-based collision methods, for example using the GPU. The implementation might also benefit from parallelization, especially for checking collisions in multiple keyframes at once. Further, the rendering of individual SVG elements in animation for very large amounts of words can cause slowdowns. While the benefits of interaction methods provided by D3 and SVG are

significant, an alternative rendering mode might be useful. For example, WebGL might be a way to animate larger amounts of words at more frames per second.

Finally, the effect of using the different visualization methods described, especially word rotation as compared to color changes, is not fully explored. A perception test with users might give insights into which methods or combinations thereof and which settings (such as rotation angle or color hue) are best suited to communicate the information. Further studies with different datasets and more precisely measured responses could also clarify where the use of dynamic word clouds is a desirable option or where alternative visualization methods are preferable.

Bibliography

[BCL+16] Kevin Buchin, Daan Creemers, Andrea Lazzarotto, Bettina Speckmann, and Jules Wulms. Geo word clouds. In *PacificVis*, pages 144–151. IEEE Computer Society, 2016.

[Bos11] Mike Bostock. D3 – data-driven documents. `https://d3js.org/`, 2011. Accessed: 2017-02-15.

[CLC+15] Ming-Te Chi, Shih-Syun Lin, Shiang-Yi Chen, Chao-Hung Lin, and Tong-Yee Lee. Morphable word clouds for time-varying text data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 21(12):1415–1426, 2015.

[CVW09] Christopher Collins, Fernanda B. Viégas, and Martin Wattenberg. Parallel tag clouds to explore and analyze faceted text corpora. In *IEEE Visual Analytics Science and Technology*, pages 91–98. IEEE Computer Society, 2009.

[CWL+10] Weiwei Cui, Yingcai Wu, Shixia Liu, Furu Wei, Michelle X. Zhou, and Huamin Qu. Context preserving dynamic word cloud visualization. In *PacificVis*, pages 121–128. IEEE Computer Society, 2010.

[Dav12] Jason Davies. Word cloud generator. `https://www.jasondavies.com/wordcloud/`, 2012. Accessed: 2017-02-15.

[Hoe12] Rama C Hoetzlein. Graphics performance in rich internet applications. *IEEE Computer Graphics and Applications*, 32(5):98–104, 2012.

[JLS15] Jaemin Jo, Bongshin Lee, and Jinwook Seo. Wordleplus: Expanding wordle's use through natural interaction and animation. *IEEE Computer Graphics and Applications*, 35(6):20–28, 2015.

[Kan14] Michael Kane. Word swarm. `http://www.kdnuggets.com/2014/07/wordswarm-visualizing-word-trends-periodicals.html`, 2014. Accessed: 2017-02-15.

[KKEE11] KyungTae Kim, Sungahn Ko, Niklas Elmqvist, and David S. Ebert. Wordbridge: Using composite tag clouds in node-link diagrams for visualizing content and relations in text corpora. In *HICSS*, pages 1–8. IEEE Computer Society, 2011.

[KLKS10] Kyle Koh, Bongshin Lee, Bo Hyoung Kim, and Jinwook Seo. Maniwordle: Providing flexible control over wordle. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1190–1197, 2010.

[LRKC10] Bongshin Lee, Nathalie Henry Riche, Amy K. Karlson, and M. Sheelagh T. Carpendale. Sparkclouds: Visualizing trends in tag clouds. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1182–1189, 2010.

[Mil76] Stanley Milgram. Psychological maps of Paris. *Environmental psychology: People and their physical settings*, pages 104–124, 1976.

[ML10] Carmel McNaught and Paul Lam. Using wordle as a supplementary research tool. *The Qualitative Report*, 15(3):630, 2010.

[RGMM07] A. W. Rivadeneira, Daniel M. Gruen, Michael J. Muller, and David R. Millen. Getting our head in the clouds: toward evaluation studies of tagclouds. In *Conference on Human Factors in Computing Systems*, pages 995–998. ACM, 2007.

[Rus06] Terrell Russell. Cloudalicious: Folksonomy over time. In *JCDL*, page 364. ACM, 2006.

[SI10] Julie Steele and Noah Iliinsky. *Beautiful Visualization: Looking at Data through the Eyes of Experts*. O'Reilly Media, Inc., 2010.

[SSS+12] Hendrik Strobelt, Marc Spicker, Andreas Stoffel, Daniel A. Keim, and Oliver Deussen. Rolled-out wordles: A heuristic method for overlap removal of 2d data representatives. *Computer Graphics Forum*, 31(3):1135–1144, 2012.

[VW08] Fernanda B. Viégas and Martin Wattenberg. Timelines - tag clouds and the case for vernacular visualization. *Interactions*, 15(4):49–52, 2008.