Dynamic Word Clouds

Martin Seyfert
TU Wien
Vienna, Austria

Ivan Viola
TU Wien
Vienna, Austria

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.

CCS CONCEPTS

• **Human-centered computing → Information visualization**;

KEYWORDS

word cloud, tag cloud, dynamic, animated, time-varying

ACM Reference Format:
Martin Seyfert and Ivan Viola. 2017. Dynamic Word Clouds. In *SCCG '17: SCCG '17: Spring Conference on Computer Graphics 2017, May 15–17, 2017, Mikulov, Czech Republic.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3154353.3154358

1 INTRODUCTION

The method of visualizing word frequency via font size goes back many years, from early experiments in the 1970s by Stanley Milgram [Milgram 1976] to the rise of so-called "tag clouds" in Web 2.0 design [Viégas and Wattenberg 2008]. 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.

Jonathan Feinberg's work on the social bookmarking application "dogear" for IBM and, eventually, Wordle[1] (his free, web-based implementation of the algorithm) popularized a new way of displaying words in a cloud layout [Steele and Iliinsky 2010]. Wordle offers automatic text analysis for word frequency (which led to a shift from the term "tag cloud" to "word cloud"), places words freely instead of within a paragraph and considers the white space between individual glyph shapes to create a more compact and aesthetically pleasing layout.

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 [Rivadeneira et al. 2007], word clouds still have been found to be a useful supplementary research tool [McNaught and Lam 2010]. With 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 in 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 [Steele and Iliinsky 2010]. 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 [McNaught and Lam 2010; Rivadeneira et al. 2007] and further experiments in user interaction [Jo et al. 2015; Koh et al. 2010] 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.

With these considerations in mind, we propose 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 visualization methods add visual information besides font size. A color gradient as well as word rotation is used to emphasize changes in word size. 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.

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

SCCG '17, May 15–17, 2017, Mikulov, Czech Republic

ACM ISBN 978-1-4503-5107-2/17/05…$15.00
https://doi.org/10.1145/3154353.3154358

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" [Steele and Iliinsky 2010]. 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. "Rolled-out Wordles" [Strobelt et al. 2012] offer an improved word placement strategy to resolve overlaps, which results in 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" [Buchin et al. 2016]. 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 [Kim et al. 2011] 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. ManiWordle [Koh et al. 2010] allows moving and rotating individual words in a word cloud via drag-and-drop to refine the layout manually. WordlePlus [Jo et al. 2015] 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.

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 [Collins et al. 2009] 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 [Lee et al. 2010] 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" [Russell 2006], which simply displayed tag frequency changes over time as a graph.

Cui et al. [Cui et al. 2010] 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. A more complex approach for "morphable word clouds" by Chi et al. [Chi et al. 2015] 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 [Kane 2014] 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 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.

3 METHODOLOGY

The basic idea of our approach 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. 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" [Steele and Iliinsky 2010], 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 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

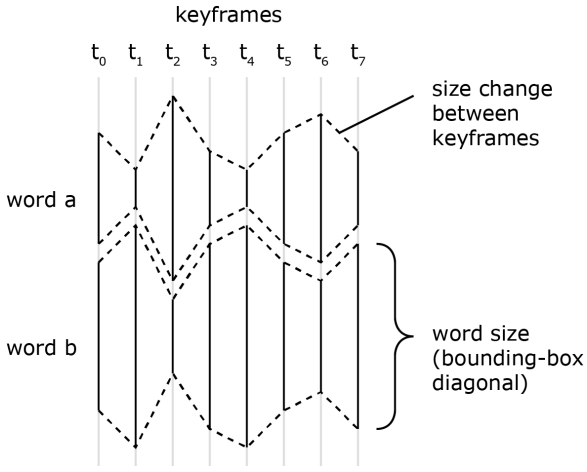

Figure 1: An example of an ideal match between two bounding boxes. The change in size (as measures by its bounding-box diagonal) between word a and word b at each keyframe adds up to zero while the combined size stays constant.

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 points over time. Once a collision is detected in any keyframe, the position is rejected for all keyframes as illustrated in Fig. 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. A simple rectangular spiral pattern is used. While it can cause the overall cloud layout to look slightly square, it is good enough to generate a centered layout (see Fig. 3).

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.

In a last step, the newly placed word is moved linearly towards the center of the combined bounding box of the previously placed words, until it collides. This is done separately in all keyframes. The goal is making the layout even slightly more compact. The complete result is then centered in the playing field before the next

Figure 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. Note 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).

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

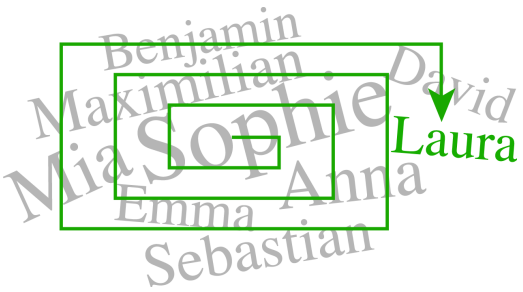

Figure 3: In order to find the closest available position to the center of the word cloud, the word is moved outward along a rectangular spiral path until no collisions are found.

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.

3.3 Color, rotation and other typographic options

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. [Cui et al. 2010] 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 for maximum color intensity of a gradient from a black base color.

Word rotation is another option 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 mostly 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.

3.4 Frame Interpolation

While methods described so far guarantee words not intersecting at the provided keyframes, interpolation is used to animate in-between states. This can occasionally cause words to still overlap (Figure 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. This further increases calculation time and therefore should be considered a luxury refinement. A single frame of interpolation can already serve as a compromise.

Figure 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.

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 technologies used are HTML5, JavaScript and the data visualization library D3 [Bostock 2011]. Words are rendered as SVG text elements on a website and can thus be easily selected, copied or stylized using a variety of formatting options. D3 provides interactive elements such as a slider and handles interpolation and animation between keyframes. Collision is done by rendering the SVG text onto a HTML5 canvas and reading the bitmap data. The code is available on GitHub[2].

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 inefficiencies still can be a problem. The main bottleneck lies in comparing bitmaps for collision. One possible optimization used in Wordle is using hierarchical bounding boxes. However, an existing implementation for static word clouds using D3 by Jason Davis [Davies 2012] suggests a faster option using 32-bit integers. The one-bit bitmaps retrieved from rendering the SVG text to an HTML5 canvas are simply stored as 32-pixel blocks, each pixel representing a bit in a 32-bit integer. This way, checking for collisions is reduced to only a single operation for 32 pixel values at once. Bit-shifting as well as simple AND and OR operations can be used to efficiently manipulate bitmaps stored as 32 bit blocks.

4.3 Data

While not a focus of this project, retrieving data is an important and often rather straightforward part of word cloud generation. Wordle [Steele and Iliinsky 2010] 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

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

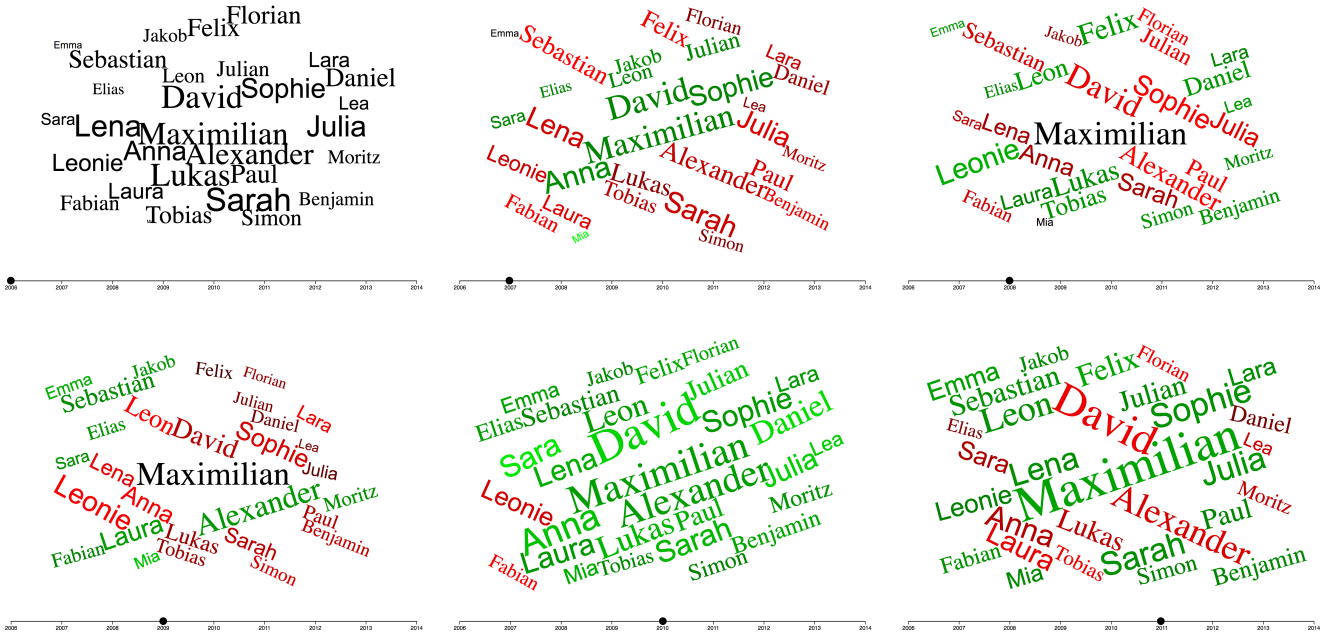

Figure 5: An example of a dynamic word cloud illustrating the most popular given names in Vienna between 2006 and 2011. Shades of red and clockwise rotation indicate shrinking, shades of green and counter-clockwise rotation growth. Font family (Arial for female, Times for male) indicates gender. The transitions are smoothly animated. Word overlaps can occasionally occur in interpolated frames.

in a database or Excel file. The original use for tag clouds relied exclusively on data provided in such an easily usable fashion.

Handling changing word frequencies over time 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.

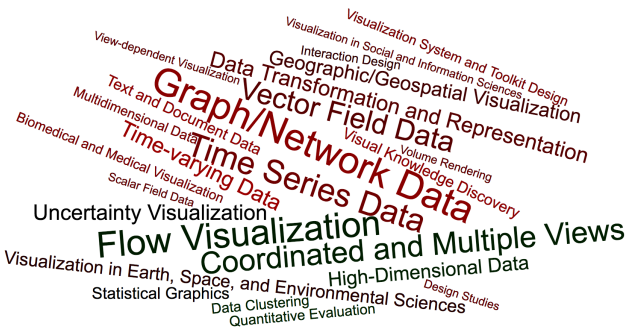

Figure 6: A visualization of submitted and accepted keywords from PacificVis 2016, weighted around a 0.5 ratio. For example, half of the submitted papers about "Uncertainty Visualization" were accepted while only about 24% of papers using the popular keyword "Graph/Network Data" were accepted.

Keyframes also have a label (for example, the year) which will be displayed in the interface.

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 by the Austrian government[3]. The preformating of the data was performed in Excel. The input was saved in the form of a CSV file. Different font families (Arial for female, Times for male) were used to indicate gender.

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.

In the resulting visualization (Fig. 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. 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.

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

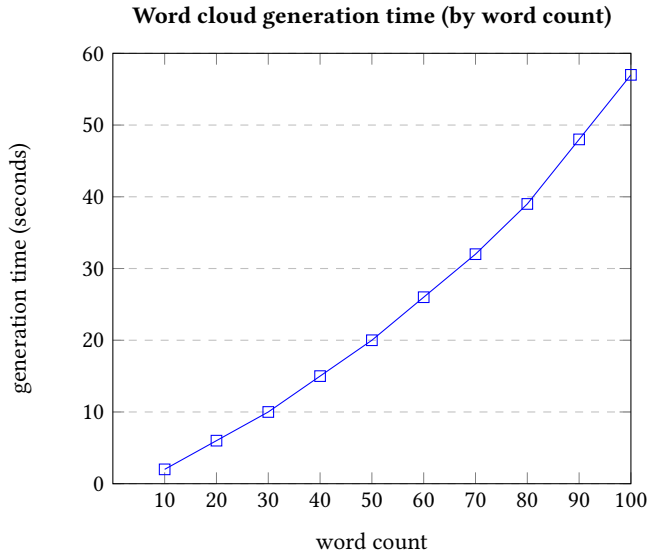

Figure 7: Word cloud generation time for a dataset of the most popular given names in Vienna between 2006 and 2014 (9 keyframes, no interpolation). While generation time grows as more words are used, the new words added are also smaller in size and thus easier to place, which keeps generation time almost linear within realistic word counts.

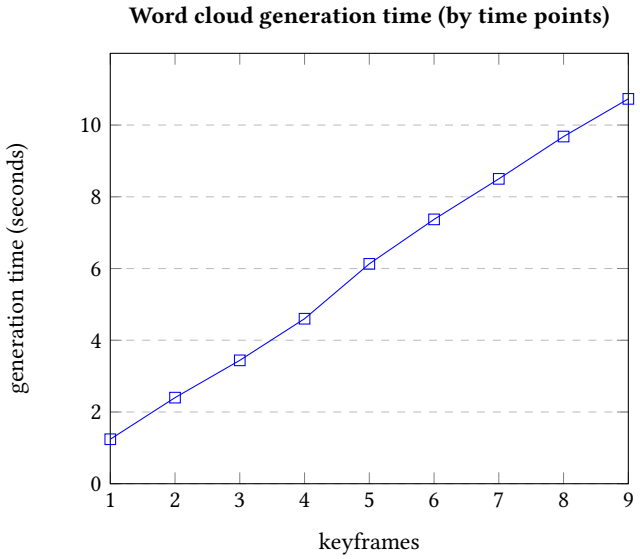

Figure 8: Word cloud generation time for a dataset of the 30 most popular given names in Vienna between 2006 and 2014 with different amounts of keyframes (1 to 9).

5 EVALUATION

A small user study was conducted. 12 participants were given a link to a web-implementation[4] 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. 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. Upon being asked, users could come up with examples for other data sets for which they could imagine the visualization 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" [Chi et al. 2015] 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.

6 DISCUSSION

Evaluation shows that users respond to the visualization with interest and can read certain trends within the dataset. The use of rotation, however, can be problematic as it might be distracting or overwhelming.

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. However, in our example dataset, adding more words did

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

not increase performance as drastically as feared (Fig. 7). This is probably due to the dataset being sorted 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. 8).

It is also questionable whether word clouds with such large amounts of words are even useful for analyzing time-varying data. The used library, D3, struggles animating so many words at once, which is another technical barrier for realistic use.

7 CONCLUSION

The visualization method leaves an overall positive impression among test users but likely requires further adjustments and evaluation before deployment for wider use. Users enjoy the general look and are able to recognize trends in the data. Problems of static word clouds, such as difficulties in making exact comparisons between words, remain in their dynamic representation. Secondary visualization elements such as size, rotation and font type have to be used with care as they can quickly overwhelm users.

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. Further optimization would be desirable. Since users can't be expected to wait up to a minute to see the results in most real-world uses, the word positions would have to be pre-generated and then reused. This way, the dynamic word cloud would appear almost instantly to most users. Because streaming of real-time data is not supported in the used methodology, anyway, pre-generation would not be a major limitation.

There are several ways of illustrating word change per key-frame typographically, without adding separate visualization methods that would go beyond what can be considered a "word cloud". Methods we explored 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 are noticeable even in a static frame of the result.

8 FUTURE WORK

There is room for further evaluation since there was unfortunately not enough time to conduct a more in-depth user study. The most effective use of visualization elements such as color and rotation could be determined by comparing a larger amount of different settings and combinations. More test data sets could give insight into which types of data are best suited for dynamic word cloud representation (for example, maximum number of words or word length). Other visualization methods for illustrating time-varying data could be compared to dynamic word cloud representations, especially in regard to the accuracy and speed at which users can form an impression.

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. [Chi et al. 2015] 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 the calculations over multiple keyframes for many words (50+) can become more and more tedious, 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. Unfortunately, other word cloud generation methods, especially when using more complex positioning requirements, suffer from considerable generation times as well [Buchin et al. 2016], which makes it likely that this is a problem that is hard to solve. There might be more efficient packaging algorithms which are applicable. One possible way of achieving better performance would be to give options for using simpler collision methods, 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.

ACKNOWLEDGMENTS

We would like to thank the Visualization Group at TU Wien, in particular Dr. Manuela Waldner and Meister Eduard Gröller, for their feedback and support.

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.

REFERENCES

Mike Bostock. 2011. D3 – Data-Driven Documents. https://d3js.org/. (2011). Accessed: 2016-02-15.

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

Ming-Te Chi, Shih-Syun Lin, Shiang-Yi Chen, Chao-Hung Lin, and Tong-Yee Lee. 2015. Morphable Word Clouds for Time-Varying Text Data Visualization. *IEEE Transactions on Visualization and Computer Graphics* 21, 12 (2015), 1415–1426.

Christopher Collins, Fernanda B. Viégas, and Martin Wattenberg. 2009. Parallel Tag Clouds to explore and analyze faceted text corpora. In *IEEE Visual Analytics Science and Technology*. IEEE Computer Society, 91–98.

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

Jason Davies. 2012. Word Cloud Generator. https://www.jasondavies.com/wordcloud/. (2012). Accessed: 2016-02-15.

Jaemin Jo, Bongshin Lee, and Jinwook Seo. 2015. WordlePlus: Expanding Wordle's Use through Natural Interaction and Animation. *IEEE Computer Graphics and Applications* 35, 6 (2015), 20–28.

Michael Kane. 2014. Word Swarm. \tolerance9999\emergencystretch3em\ relaxhttp://www.kdnuggets.com/2014/07/wordswarm-visualizing-word-trends-periodicals.html. (2014). Accessed: 2016-02-15.

KyungTae Kim, Sungahn Ko, Niklas Elmqvist, and David S. Ebert. 2011. WordBridge: Using Composite Tag Clouds in Node-Link Diagrams for Visualizing Content and Relations in Text Corpora. In *HICSS*. IEEE Computer Society, 1–8.

Kyle Koh, Bongshin Lee, Bo Hyoung Kim, and Jinwook Seo. 2010. ManiWordle: Providing Flexible Control over Wordle. *IEEE Transactions on Visualization and Computer Graphics* 16, 6 (2010), 1190–1197.

Bongshin Lee, Nathalie Henry Riche, Amy K. Karlson, and M. Sheelagh T. Carpendale. 2010. SparkClouds: Visualizing Trends in Tag Clouds. *IEEE Transactions on Visualization and Computer Graphics* 16, 6 (2010), 1182–1189.

Carmel McNaught and Paul Lam. 2010. Using Wordle as a supplementary research tool. *The Qualitative Report* 15, 3 (2010), 630.

Stanley Milgram. 1976. Psychological maps of Paris. *Environmental psychology: People and their physical settings* (1976), 104–124.

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

Terrell Russell. 2006. Cloudalicious: Folksonomy over time. In *JCDL*. ACM, 364.

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

Hendrik Strobelt, Marc Spicker, Andreas Stoffel, Daniel A. Keim, and Oliver Deussen. 2012. Rolled-out Wordles: A Heuristic Method for Overlap Removal of 2D Data Representatives. *Computer Graphics Forum* 31, 3 (2012), 1135–1144.

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