Geo word clouds

Geo Word Clouds

Kevin Buchin*
TU Eindhoven, The Netherlands

Daan Creemers†
TU Eindhoven, The Netherlands

Andrea Lazzarotto‡
Ca' Foscari University of Venice, Italy

Bettina Speckmann§
TU Eindhoven, The Netherlands

Jules Wulms¶
TU Eindhoven, The Netherlands

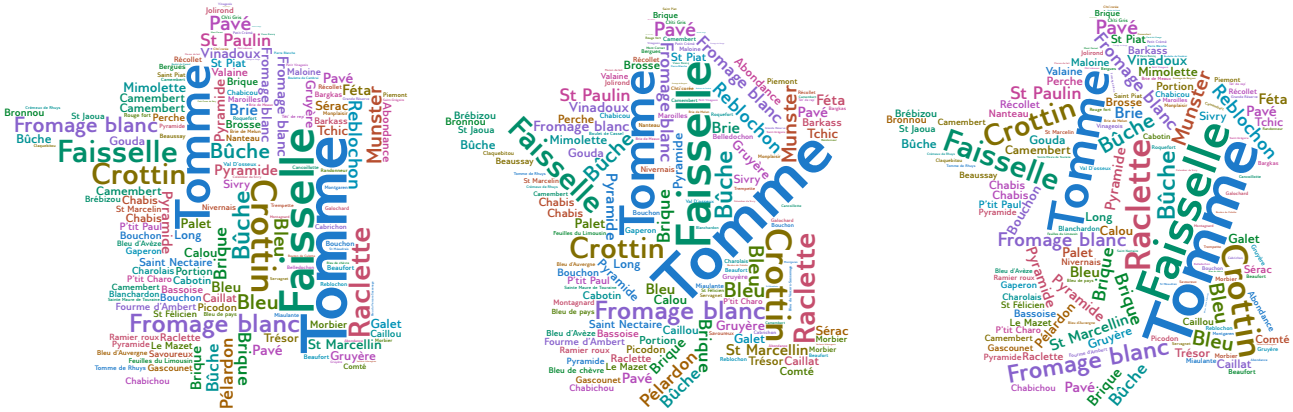


Figure 1: Geo word clouds of cheese production in France (left: orthogonal, middle: 45 degrees, right: 10 degrees).

Abstract

Word clouds are a popular method to visualize the frequency of words in textual data. Nowadays many text-based data sets, such as Flickr tags, are geo-referenced, that is, they have an important spatial component. However, existing automated methods to generate word clouds are unable to incorporate such spatial information. We introduce *geo word clouds*: word clouds which capture not only the frequency but also the spatial relevance of words.

Our input is a set of locations from one (or more) geographic regions with (possibly several) text labels per location. We aggregate word frequencies according to point clusters and employ a greedy strategy to place appropriately sized labels without overlap as close as possible to their corresponding locations. While doing so we "draw" the spatial shapes of the geographic regions with the corresponding labels. We experimentally explore trade-offs concerning the location of labels, their relative sizes and the number of spatial clusters. The resulting word clouds are visually pleasing and have a low error in terms of relative scaling and locational accuracy of words, while using a small number of clusters per label.

Index Terms: I.3.3 [Computer Graphics]: Picture/Image Generation

1 Introduction

Word clouds are a text-based visualization highlighting important keywords in large amounts of text, emphasizing more important or frequent words by a larger font size. Due to the popularity of this visualization method, there are many recent studies on word clouds, on topics ranging from usability issues [12], over the visualization of dynamic text content [4], to the appropriate ranking of the words to be visualized in large volumes of Twitter data [9].

Many text-based data sets, such as Flickr tags, have an important spatial component. It hence seems natural to incorporate this spatial component when creating word clouds. Consequently there are many (manually crafted) examples of "word cloud maps", that is, geographic maps which are composed solely of words[1] (often names of regions). However, existing automated methods to generate word clouds are unable to incorporate such spatial information.

In this paper we propose an automated method to create *geo word clouds*: spatially informative word clouds (see Fig. 1). Our word clouds capture not only the frequency but also the spatial relevance of words. Furthermore, they "draw" the geographic regions, from which the input is taken, to both attract attention and to give visual cues to identify locations.

In Section 2, we formally define geo word clouds, give exact requirements and specify corresponding quality measures. In Section 3 we then present a greedy algorithm to compute geo word clouds. An appropriate coloring scheme is an important part of word cloud design. In Section 4 we hence explain our design choices with respect to color in detail. There are several parameters to choose when generating geo word clouds, such as rotation angles for the words, (relative) font sizes and the amount of clustering of the input. In Section 5 we experimentally evaluate how different parameter choices influence the quality of the resulting geo word clouds and explore trade-offs between different requirements. We can conclude that the geo word clouds generated by our greedy algorithm are visually pleasing and satisfy our quality criteria well. Finally in Section 6 we discuss various ways to extend our current approaches to different scenarios and requirements.

Related work. Possibly the best-known algorithm for word cloud generation is the one by Wordle [17]. It utilizes a simple greedy

*email: k.a.buchin@tue.nl
†email: d.j.a.creemers@student.tue.nl
‡email: 833897@stud.unive.it
§email: b.speckmann@tue.nl
¶email: j.j.h.m.wulms@student.tue.nl

[1] Image search for *word world map*

approach to produce visually pleasing word clouds. Barth *et al.* [1] consider a scenario where a set of words is given with adjacency information, that is, the relative positions of words in a word cloud are pre-specified. They show that in general, the problem of creating word clouds, while respecting adjacencies between words, is NP-hard. They consequently present an approximation algorithm for this problem. Both of these algorithms do not consider spatial locations and hence cannot be used to produce geo word clouds.

There are various methods similar to our approach, but none are directly comparable. Chi *et al.* [3] propose *morphable word clouds*, where a sequence of spatial shapes is "drawn" with a time-varying set of words. To fill up the shapes they use a greedy algorithm, but when placing the words no spatial constraints are taken into account, while this is of importance for geo word clouds. Moreover, they use rigid body dynamics to displace the words over time while enforcing constraints. Their work is mostly focused on the temporal aspect, that is, arranging the time-varying sets of words within the morphing shapes such that the evolution of both word clouds and shapes can be easily tracked by the user. Our algorithm and constraints, however, focus on the spatial scenario where every input location can have multiple labels and each word can label many different locations. Our algorithm hence has to cluster labels appropriately and be more flexible when optimizing both the clustering and the placement of words.

Related to both our work and classic map labeling are *tag maps* [8, 13] which are intended to "expose textual topics that are tied to a specific location on a map". Tag maps place the tags at a fixed location on a map with a size corresponding to their importance (thus, in fact, creating a graduated symbol map with varying text symbols). Since the locations for all tags are fixed, overlaps between tags frequently occur. In contrast, it is central to geo word clouds to avoid overlap.

There is an extensive body of work on automated label placement for point features [18]. Map labeling is loosely related to geo word clouds, but there are some crucial differences. First of all, our word clouds *are* the map and as such, cover space (more or less) completely. Second, we do not label specific features but instead are summarizing spatial word data. Hence we have considerably more flexibility in placing words and in deciding how to aggregate data associated with the same word. Finally, when labeling maps, it is common to omit labels to avoid clutter. For our geo word clouds it is of utmost importance to show all labels of high relevance. So the common algorithmic approach of maximizing the number of labels placed is not applicable (instead one could consider a variant of maximizing the size of labels [6]).

Geo word clouds are an abstract representation of spatial data. Since the word size is scaled according to importance, they have a certain resemblance to cartograms. Cartograms convey values per region and several types of cartograms represent regions by simple geometric shapes, such as circles [5], squares or rectangles [11], scaled according to the value associated with the region. However, the challenges for generating word clouds differ considerably from those for cartograms: cartograms are based on a subdivision of space into regions, which provides a starting point for an overlap-free placement. In contrast, placing words very close to the corresponding data points in a word cloud, naturally will lead to considerable overlap, which needs to be resolved.

Ignoring the issue of overlap it would seem natural to use rectangular cartograms to generate word clouds. However, methods to generate rectangular cartograms (for example [2, 16]) place a strong emphasis on maintaining adjacencies between regions and as a result limit the freedom of choosing the aspect ratio of rectangles. Such region adjacencies are of little importance for geo word clouds, but being able to choose aspect ratios is of high importance. It hence seems unlikely that methods for rectangular cartograms can easily be adapted to generate geo word clouds.

2 Geo Word Clouds

Our input is a set of n points (locations) $p_1, \ldots, p_n$ in $\mathbb{R}^2$ which are located inside a geographic region M. Each point is associated with one or more words w_i from a list of m words $w_1, \ldots, w_m$. Each word has a specific shape, namely the outline of the word spelled in a specific font chosen by the user. Furthermore, the user can restrict the possible (degrees of) rotation of words in the output. As customary with word clouds, more frequent (important) words are drawn using a bigger font size. Finally, the words should be placed in spatial proximity to their defining points. Given this input, we want to create a geo word cloud (a set of word shapes M') which fills (and hence draws) the geographic region M.

2.1 Requirements

Requirement 1 (R_1) *All word shapes are disjoint.*

Requirement 2 (R_2) *The geo word cloud has a small number of shapes per word.*

R_2 prevents valid outputs that are not interesting. See Table 1 where every point in the input is represented by an individual word. Instead, if points associated with the same word are relatively close together, we want to place a single bigger word shape in our word cloud which ideally covers the points. To do so, we cluster the points associated with a single word (see Section 3) and place a single shape per cluster, sized according to cluster size. R_2 is hence intended to keep the clustering from *overfitting*.

Requirement 3 (R_3) *Shapes of a word cover its points well.*

To emphasize the spatiality of geo word clouds, we ensure that the shapes in our output are close to their corresponding data points. See Table 1: the more points are covered by a word and the smaller the distance between the word and the points that are not covered, the better the word represents the spatial data. In subsection 2.2 we show how we measure the error for the points that are not covered.

Requirement 4 (R_4) *A word with k points has a total area as close as possible to* $\frac{k}{n}M'$.

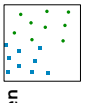
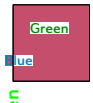
Requirement	Input	Bad	Good
R_1			
R_2			
R_3/M_1			
R_4/M_2			
R_5/M_3			
R_{6+7}			

Table 1: Requirements and measures.

The size of a word should give the reader a good indication of the importance of the word. See Table 1: one word has three points, while the other has only two. To express this in the word cloud, we want a ratio of about 3 : 2 when it comes to the area of the words representing these points.

Requirement 5 (R_5) *M' covers input area M well.*

R_5 ensures that the outline of M becomes visible in the output, and that the word cloud we generate is dense. We want the shapes of M', to stay inside M, but at the same time do not leave too much white space.

Requirement 6 (R_6) *Shapes of a single word have the same color.*

R_6 ensures that geo word clouds look consistent and that it is easy to find multiple instances of the same word. However, human users cannot distinguish many colors. Hence we developed a careful scheme to reuse colors, see Section 4 for details.

Requirement 7 (R_7) *Coloring should distinguish different words, but should not draw too much attention to particular words.*

The last requirement should help a reader of the geo word cloud to find different words, but at the same time make sure that only the font size and number of occurrences of a word inform the reader of its importance. R_5 and R_7 together put emphasis on the size of the word as the only indicator for its importance. See Table 1: we can see that the word *Green* has different colors in the first case, and the green color is a lot brighter than the blue one. The second color is a lot easier to read and visually pleasing. Section 4.2 explains how we ensure that different words with the same color are still clearly distinguishable.

2.2 Measures

To evaluate the quality of the geo word clouds generated by our algorithm we define three quality measures. Intuitively speaking, geo word clouds that score well on the first two measures consist of words which (mostly) have the right size and are (mostly) placed at the correct location. This is a necessary condition for the user to be able to use our word clouds effectively. The third measure ensures that the spatial shape is drawn appropriately, which is an important clue for the user to be able to accurately determine locations. Hence these measures ensure that our geo word clouds add spatial information to word clouds in a meaningful way.

When computing geo word clouds we need to make careful trade-offs between the various requirements. For example, to fill up the whole area (R_5), we might have to scale some words down to fit them (R_4). Furthermore, it might be better to make a reasonable coverage error on two words, instead of placing one perfectly and having a huge error for the other word (R_3).

As stated above, we cluster the data points of each word and then associate each cluster with a single (scaled) shape. To measure whether the shape of a cluster covers the points in that cluster well (R_3), we use the Hausdorff distance. For each point in the cluster, we measure the Hausdorff distance from this point to the bounding box of the shape. Points that are covered and are inside the bounding box of the shape have distance zero, while for points that are not covered we take the distance from the point to the bounding box. The sum of the distances for a single word is the error for that particular cluster.

Table 1 shows which points add to the coverage error and how a good placement of a word can decrease the error. The sum of the error for all points measures how well all shapes in M' cover the whole data set.

Let $Hausdorff(p,r)$ denote the Hausdorff distance between a point p and a rectangle r and let c_p denote the cluster to which point p belongs. We define $word(c_p)$ as the bounding box of the word associated with cluster c.

Measure 1 (M_1) *The coverage error for all points in M is* $\sum_{p \in M} Hausdorff(p, word(c_p))$.

It might be necessary to slightly reduce the size of some words, to find a suitable position for them. We want to measure how well the words represent the number of points they visualize. For a word with k points, which is scaled by factor x, the number of points that are not represented is $|k - k \cdot x|$. Let C be the set of all clusters, and define $Rep(c)$ the value we defined for the number of points that are not represented by the word of this cluster.

Measure 2 (M_2) *The percentage of points not represented is* $\frac{\sum_{c \in C} Rep(c)}{n}$.

If words may be scaled by a factor $x > 1$, M_2 can be over 100%.

Next we want to measure R_5: how well does M' resemble M. An effective way to do this is calculating the *symmetric difference* between M' and M: parts of M that are covered by M' do not add to this measure, but area that is not covered and covered area outside of M increases the measure. This is indicated by the colored area in Table 1. We can also see that choosing an appropriate global scaling for the words helps reducing the symmetric difference. Since we require the words to be disjoint, we know that the symmetric difference will not be lowered by overlapping words. We use a bounding box per shape for this measure, since users tend to mentally aggregate the letters, the space between letters and also the space around the ascenders and descenders of the word with the word itself.

Measure 3 (M_3) *The symmetric difference between M' and M.*

3 Placement algorithm

Placing a label for each point would clutter the map (see Section 5 for a more detailed discussion). Since we want to place bigger words whenever possible, we use k-means to cluster points with the same label and place for each cluster a single word. The size of the word depends on the number of points in that cluster. The points are clustered multiple times with an increasing number of desired clusters, to optimize the number of clusters. The best number of clusters is found when the error, the distance to the centroid of a cluster, is small. We add a penalty term to the error for adding extra clusters to ensure that we do not keep on adding clusters to decrease the error. We select the clustering that minimized the error including the penalty term.

After clustering we determine suitable values for rotating the word of each cluster. We determine the eigenvector of the points in the cluster and use the principal component of this vector. This gives the rotation of the word which minimizes the error to the bounding box of the word. We allow the user to restrict the possible rotations: orthogonal, 45 degrees, etc. For each word we then choose the rotation that is closest to the orientation of the principal component (see Fig. 1 and 2 for examples).

We use a greedy strategy to place the words. For this the words are sorted by the size of their point cluster and the placement is performed from the bigger to the smallest cluster to avoid large errors for bigger clusters. We compute the word placement pixel-based. Per pixel we store whether it is occupied by a word or not. The positions in which a word can be placed are obtained by a uniform filter, i.e. a convolution that considers only the bounding box of the word. When we actually place a word, we only set the pixels that overlap with the actual shape of the word.

At each iteration a word is selected from the top of the list and its *best placement* is computed. The best placement is a trade-off between the size of the word and the error we make: if a word cannot be placed at its exact position, we try to scale it down, so that the distance between the center of the word and the centroid of the cluster can be smaller. If we scale a word down, we again place it in the list and process it again when the scaled word is the biggest

Figure 2: Geo word clouds of Flickr tags in Great Britain and Ireland (left: orthogonal, middle: 45 degrees, right: 10 degrees).

word in the list. Furthermore, if we do not place the word close to the cluster, we decrease its size, since we cover less points with this word now. If we would keep the word at its original size, we would give the user the impression that there are a lot of points around the position of the word, while there might not be many at all.

The scaling of the words uses a finite number of values uniformly distributed from $s = 0.05$ to $s = 1.0$ times the desired size (with steps of 0.05). A penalty is introduced both for scaling a word and for moving it away from its desired position. The final penalty is computed as a weighted average $\alpha \cdot (1-s) + (1-\alpha) \cdot \delta / \sqrt{A}$ where δ is the distance between the word placement and the centroid of its cluster and A is the area of the rectangle bounding our region M. α is used to influence how important each part of the penalty is. Currently we use $\alpha = 0.5$. The best placement can now be defined as the placement with the lowest penalty. In the case of different options with the same penalty, the one with the highest value of s is preferred. If a map consists of two disjoint regions (see Fig. 2) we ensure that words that should be placed in one region cannot be placed in the other one.

It remains to discuss how the optimal size of each word is determined. As R_4 suggests, the preferred font size is determined by computing the font size which renders the word in a bounding box of area $k \cdot A_M / n$, if the cluster for that word has k points and the whole data set has n points. The value A_M is the area of M in our pixel-based image, measured in pixels. We stop the algorithm if the words become too small, either because their cluster size is small, or they are scaled down by a large amount. Algorithm 1 shows pseudocode for the greedy approach.

Algorithm 1 Greedy geo word clouds

```
M ← acquire shape
D ← load data set
C ← create clusters from D
for all c ∈ C do
    assign font size to c
    assign rotation to c
P ← sort C (biggest first)
for all c ∈ P do
    p ← find best placement for c
    if p.size < minimal size then
        break
    if p.percent < 100% then
        adjust font size of c
        keep c in P
        reorder P
    else
        place c inside M
        remove c from P
        update the available space
return M
```

4 Coloring

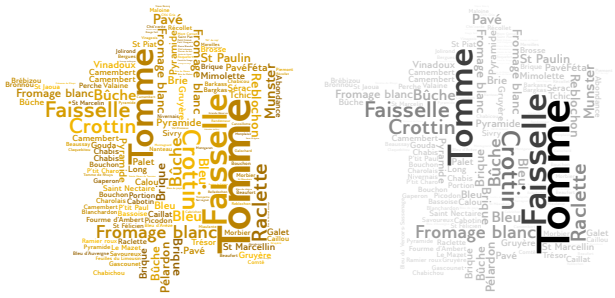
The first figure in [17] shows a Wordle with its standard coloring. We can quickly see that R_6 is not satisfied, since the word *Wordle* and *Visualization* both occur twice, but have different colors. Furthermore, except for the big occurrence of *Wordle*, nearly all big words use the light grey color, and nearly all smaller words a shade of red/brown. Below we describe our approach to compute a better coloring.

Two of our requirements, namely R_6 and R_7, concern the coloring of the geo word clouds. Our algorithm adheres to these requirements (see, for example, Fig. 1, the word *Tomme* has the same color everywhere). We choose our colors from a Hue-Chroma-Luminance (HCL) color model to ensure that no color (and hence word) is perceived to be more important than others. At this point we do *not* use color to group semantically similar words or to evoke associations (*sea* and *sky* in Fig. 2 are not both blue, in fact, neither is). Our only goal is to assign colors in such a way that the different words and their sizes in the word cloud can be identified as easily as possible. We cannot use a different color for each word, since the number of colors needed would exceed what users can perceive. Instead we use a small fixed colour palette and ensure that words which are similar in size are unlikely to have the same color.

4.1 HCL color maps

HCL color maps have "well balanced perceptual properties" as argued in [15]. By keeping the Chroma and Luminance on the same level, we can create a qualitative color map containing colors that are uniformly distributed over the whole Hue spectrum. These colors are very similar in brightness and none of them attracts more attention than others. We choose such a qualitative color map of 13 colors which are still distinguishable enough.

Another way of creating HCL color maps is by keeping the Chroma on the same level, choosing a single Hue and having a variable Luminance. This results in single Hue, sequential color maps. Fig. 3a shows a geo word cloud when such color map is used. This is a nice variation, but arguably qualitative color maps make it easier to distinguish words.



(a) Sequential "cheese" colors. (b) Grayscale dependent on font size.

Figure 3: Geo word clouds of France: different color maps.

4.2 Color distribution

We want to use a small number of colors, but still distinguish words well. Therefore, we use an algorithm that assigns different colors to words of similar size, but reuses them after they have all been used. Assume the use of a color map of l colors uniformly distributed over the whole Hue spectrum $[col_0, \ldots, col_{l-1}]$. Colors are assigned in decreasing size of the words. We start from the biggest word and assign a color col_0 in our color map to this word.

The next word in decreasing order of word sizes, gets a color $col_{(i+c) \mod l}$, where c is relatively prime to l and i is the index of the previously used color. Hence we ensure that all colors are used before using a color twice, etc. For most of our maps, we have used a color map of $l = 13$ colors, which is a prime itself. Therefore we can choose any value for c, for example $c = 3$ (see Fig. 4). Using this algorithm to assign colors in descending order of font size, we make sure that words that are similar in size get a color that is at least c colors away from the previous color. Since we have chosen $c > 1$, colors that are assigned consecutively will be quite different.

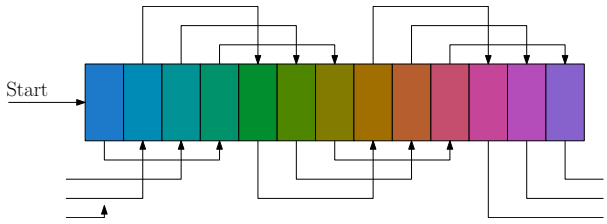


Figure 4: HCL color map, selecting consecutive colors.

By the time that we use the same color, or a similar one again, the words that we assigning the color to have shrunk in size and are distinguishable by size already. For the smallest words the previous might not hold. However, there are many small words and the chance of two adjacent words getting the same color is very small.

These claims are supported by Fig. 3b: We see that there are not many black or dark gray words, which means the biggest words will all get very distinguishable colors. The space between these big words is filled with light gray words and the space that is left is filled with nearly white words. The chance of two adjacent light gray or two adjacent white words to have the same color is small.

5 Experiments

We tested the algorithm on two different data sets. The first data set is about cheese produced in France and has been constructed manually from different websites, starting from a list of producers[2]. The data set contains 95 producers of cheese with 125 unique tags, i.e. types of produced cheese. The second data set is filtered from a Flickr data set with geo-tagged photos [10]. We filtered the data set keeping only Great Britain and Ireland, selected the most frequent tags and removed those that are about the camera or mode in which the photo was taken. The resulting data set contains 545,140 images with 126 unique tags.

The algorithm described above was implemented using Sage [14], a Python based mathematical software package. We use Sage Math Cloud to run the experiments. The platform is a web based solution which provides a hosted instance of Sage. The hardware quota consists of 4 Intel(R) Xeon(R) processors with a clock speed of 2.30 GHz and 4 GB of RAM.

In the experiments we use the measures as described in Section 2. However, we make some changes to the measures to make maps of different sizes comparable: the coverage error is measured as the total sum of distances and we divide this error by the diagonal of the map. Looking at the error in percentages expresses the amount of displacement relative to the diagonal. We measure how much of the map is filled by measuring the symmetric difference.

The symmetric difference is normalized to the unit interval and expresses the percentage of the map that is not covered by words. We measure the percentage of points that is not represented by the words as stated earlier: for each cluster we look at the number of points that is not represented, take the sum for all the clusters and divide by the total number of points.

Below we describe several experiments. We start by testing the influence of the allowed rotations on the geo word cloud. We then test different settings for the initial font size and the clustering. We also illustrate the word placement of our algorithm and discuss the running time of our computations.

5.1 Allowed rotations

One of the input parameters determines the allowed rotations that words may have in the output. The algorithm determines the orientation of a cluster using the principal component of its points. We test the algorithm by allowing rotations of 90°, 45° or 10°. Fig. 1 and Fig. 2 show respectively France and Great Britain plus Ireland, with different parameter values for the allowed rotations. In general, the choice of orthogonal words produces a map which is more evenly spaced. The words are tightly packed together and the texture is more uniform.

The part of the map concerning Ireland looks sparse because the amount of tags included in the data set was more dense in Great Britain and the algorithm tries to preserve the relative scaling of words. For this reason, not every region is guaranteed to be filled uniformly, when the map consists of multiple separate areas.

Adding more allowed rotations decreases the compactness and has the risk of producing a non-uniform texture. This is especially true in the case of France, where the data set contains less words. Such choice for the rotations has a minor impact in the case of Great Britain and Ireland. This happens because the presence of many more tags allows to fill the gaps left by rotated words.

A positive effect of more rotation options is that words in Ireland tend to spread more and the result is less sparse. Finally, we note that the structure of a specific data set may help. Since most clusters in Great Britain have a strong vertical orientation, the result looks more uniform, even when having more freedom in the rotations.

Table 2 and Table 3 show the coverage average error per point, percentage of words not represented and the symmetric difference for the allowed rotations for the data sets France and Great Britain plus Ireland. The coverage error is smallest when allowing rotations of 90°. The difference in error between the geo word cloud that uses 90° and the one that uses 45° is only 0.69%, which is acceptable.

By allowing more rotations words are placed in a lot of different orientations such that placing all words becomes difficult. The result of this is that words get scaled down, which explains why

[2] http://www.monmenu.fr/s/exploitations/Liste-Producteurs-De-Fromages-cat03500

Allowed rotations	Coverage error	Words not represented	Symmetric difference
90°	7.31%	11.28 %	22.26 %
45°	8.00 %	14.19 %	24.08 %
10°	8.74 %	15.88 %	25.41 %

Table 2: Metrics on allowed rotations (France)

Allowed rotations	Coverage error	Words not represented	Symmetric difference
90°	10.43 %	25.18 %	35.93 %
45°	10.90 %	27.55 %	37.93 %
10°	10.63 %	30.40 %	38.53 %

Table 3: Metrics on allowed rotations (Great Britain and Ireland)

the percentage of words that are not represented increases when the number of allowed rotations increases.

The symmetric difference is smallest when only allowing rotations of 90°, which matches our observation that the geo word cloud looks more packed.

Comparing the measures of the two data sets shows that France is covered better than the Great Britain plus Ireland, based on the symmetric difference. Ireland has a large influence on this value, because the data set contains less points for Ireland relative to Great Britain and we want to preserve the relative scaling.

We can aim to combine the positive effects of fewer and of more rotation options by restricting the rotations based on the importance of a word. Fig. 5 shows such a word cloud: the biggest 10 words are orthogonal, 40% use 45°rotations and all remaining words 10°.

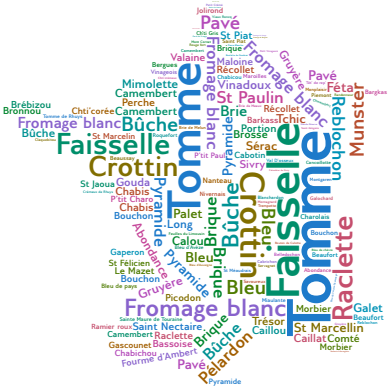


Figure 5: Allowed rotations based on size of words with smaller words having more allowed rotations.

5.2 Initial font size

R_4 of the problem formalization states that a word with k points should cover a total area as close as possible to kM'/n, where n is the total number of points and M' is the total area of the map. This suggests to start with a font size of a word such that this requirement is satisfied. We define this as the 100% initial font size.

During the algorithm some words cannot be placed nearby. A word of this kind is scaled down and reinserted into the list of words to handle. Scaling a word without also scaling other words introduces relative scaling errors. We try to prevent this if possible. This can be done by starting with smaller initial font sizes.

Table 4 shows the coverage average error per point, percentage of words not represented and the symmetric difference when starting with 100, 90 or 80 percent of the optimal size, for the France data set. We observe that decreasing the initial font size from 100% of the optimal size to 90% increases the average coverage error. The reason for this is that all words get smaller, including the large words such that they probably cover less points of their cluster. The distance from the points to those large words is now also increased. Decreasing the initial font size further to 80% gives a smaller error again, because the words are placed closer to their desired location.

Initial font size	Coverage error	Words not represented	Symmetric difference
100%	7.31 %	11.28 %	22.26 %
90%	8.39 %	9.13 %	25.94 %
80%	7.59 %	4.17 %	27.42 %

Table 4: Metrics on initial font size (France)

By decreasing the initial font size more words can be placed at that initial font size. Therefore, the percentage of words not represented decreases. While this percentage goes down when carefully decreasing the initial font size, the symmetric difference increases. This can be explained by the fact that all the words are simply smaller, which leads to less coverage of the map as a whole.

Although experimental data favors selecting an initial font size of 80%, visual inspection shows that there are regions which are not filled nicely. Particularly for France, the Bordeaux region starts to contain white areas if we lower the initial font size.

5.3 Clustering

Placing a word for each point in the map will generate a word cloud where it is difficult to find frequent words, as can be seen in Fig. 6a. Some measures for these kind of maps will be low, but these maps do not give an informative view of the data. The other extreme is to use a single cluster for each word type, as shown in Fig. 6c. This groups all words of a tag together. The disadvantage is that no regional differences can be seen anymore. Our approach performs clustering but preserves clusters that are far apart. The resulting geo word cloud, as can be seen in Fig. 6b, mediates between these two extremes. Multiple regions where tags are used can be observed, but the number of words per tag are not too large.

Table 5 shows the coverage average error per point, percentage of words not represented and the symmetric difference for three different types of clustering. The average error increases when points are clustered. This means that more points are placed further away from the position where they should be placed. This is as expected, since now a point may have to be placed further away to cover other points of the bigger cluster as well.

If no clustering is applied multiple words cannot be placed at their position. These words have the same position, such that the words have to be scaled down in order to place them. When words are clustered less scaling is required, because the word can be placed closer to their position. The result is that the percentage of words not represented decreasing when applying clustering.

The symmetric difference is lowest when k-means clustering is applied (see Table 5). When more clustering is used the clusters are

Clustering type	Coverage error	Words not represented	Symmetric difference
No clustering	6.02 %	23.62 %	27.05 %
k-means clustering	7.31 %	11.28 %	22.26 %
One cluster per word	11.18 %	11.22 %	24.51 %

Table 5: Metrics on type of clustering (France)

(a) No clustering. (b) *k*-means. (c) Single cluster per word.

Figure 6: Geo word clouds of French cheeses: extreme cases for clustering.

bigger, the words will have a larger font size, making them harder for words of similar size to stack nicely and fill up the map.

We conclude that the extreme cases of clustering do not work well for our problem and that our clustering is a better fit.

5.4 Word placement

To illustrate how our algorithm places words, we generated point maps which show the clusters and placements for the first three words of the French cheese data set (see Fig. 7). *Tomme* is the most frequent cheese and hence the first to be placed. The clustering generates two clusters of points, indicated by disks and crosses in Fig. 7a. The right *Tomme* is placed next to the boundary of France such that it covers many disks. The locations for the left *Tomme* are more widespread, so there is no unambiguous location for *Tomme*. It is placed as well as possible.

The second most frequent cheese is *Faisselle*. Also here the clustering returns two clusters (see Fig. 7b). The right *Faisselle* cannot be placed at its optimal location, because *Tomme* is already placed and *Faisselle* is too large for the space left. Given this restriction, *Faisselle* is placed such that it covers as many points as possible.

The third most frequent cheese is *Crottin*. Its locations are also divided into two clusters. The points for the right *Crottin* are spread. Given that *Tomme* and *Faisselle* are already placed, *Crottin* is placed such that it covers as many points as possible (see Fig. 7c).

5.5 Running time

The experiments are performed using Sage Math Cloud. The placement algorithm runs on average in 30 minutes on the France data set and 60 minutes on the Great Britain plus Ireland data set when the allowed rotations are 90°. Increasing the allowed rotations to 10° increases the running time by 60%. Preprocessing the data set including clustering and normalizing takes only seconds for the France data set, but takes several minutes for the Great Britain and Ireland data set, which has thousands of annotated images.

6 Extensions

Our current algorithm focusses on finding a word cloud where the distance from data points to their visual representation is low. We discuss several extensions that expand or change the current requirements. Furthermore, we discuss different kinds of data that would also be suitable for this kind of visualization.

A natural extension of our current requirements would take into account the directional relations between words. A second possibility is to determine the font size of a word not by the size but by the area of the corresponding point cluster. Hence a spread out cluster of points would be represented by a larger word than tightly clustered points. This kind of requirement deviates further from traditional word clouds (where size directly corresponds to frequency), but would put more emphasis on the spatial properties of the data. Alternatively, we could use a Voronoi diagram of the data points and use the area of the Voronoi cells of a cluster to determine the font size. As the size of the point set increases, the sum of the areas can be expected to approximate frequency.

We could also create geo word clouds for statistical data which is aggregated by area (like wine production or population). We have two clear options for spatially informative word clouds using this

(a) Placement of Tomme. (b) Placement of Faisselle. (c) Placement of Crottin.

Figure 7: Point maps for Geo word clouds of French cheeses: placing the first three words.

kind of data. First, we could define a good placement of the word as one where the word overlaps with the area in some way. If the word fits in the area, the best placement would be inside it, otherwise completely covering the area would be the best placement.

We can fit this new problem in the problem description we are currently using. If we overlay the map with a grid and put a data point on each grid point, we get more data points in big areas and less data points in smaller areas. We can use our current algorithm to produce an output on the data points we get in this way, to solve the problem for areas. Another option is to always require words to be inside the area. This way, we create small instances of the bigger problem within the different areas of the map.

Fig. 2 shows geo word clouds where the relative scaling between Great Britain and Ireland is maintained. Here Ireland is sparsely filled, because the data set contains less points for Ireland. We could extend the algorithm to scale words relative to the area of their component instead of to the complete map (see Fig. 8). We could also combine our approach with contiguous area cartograms [7], to distort regions to match the required size.

Figure 8: Geo word cloud of Flickr tags in Great Britain and Ireland after manually merging the countries (rotations of 10 degrees).

7 Conclusions

We introduced geo word clouds and experimentally evaluated them with respect to trade-offs between different requirements and quality criteria. We found that orthogonal orientation of words yields the most compact drawings. By using an initial font size of 90% of the optimal size we can reduce the displacement error at the cost of a slightly less filled map. We achieved this using a greedy algorithm. We also considered a force-directed approach to place the words without relative scaling, but this did not improve the placement, in particular since we have to deal with initially massively overlapping words.

The coverage of the map is also influenced by the clustering algorithm chosen, we found that k-means clustering performs best. The final geo word clouds produced by our algorithm add spatial information lacking in regular word clouds. Further studies should establish how this extra information can be utilized.

Acknowledgements. K. Buchin and B. Speckmann are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 612.001.207 and 639.023.208, respectively. We thank the primary reviewer for helpful comments on the paper.

References

[1] L. Barth, S. I. Fabrikant, S. G. Kobourov, A. Lubiw, M. Nöllenburg, Y. Okamoto, S. Pupyrev, C. Squarcella, T. Ueckerdt, and A. Wolff. Semantic word cloud representations: Hardness and approximation algorithms. In *Proc. 11th Latin American Theoretical Informatics Symposium (LATIN)*, volume 8392 of *Lecture Notes in Computer Science*, pages 514–525, 2014.

[2] K. Buchin, B. Speckmann, and S. Verdonschot. Evolution strategies for optimizing rectangular cartograms. In *Proc. 6th International Conference on Geographic Information Science (GIScience)*, volume 7478 of *Lecture Notes in Computer Science*, pages 29–42, 2012.

[3] M.-T. Chi, S.-S. Lin, S.-Y. Chen, C.-H. Lin, and T.-Y. Lee. Morphable word clouds for time-varying text data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 21(12):1415–1426, 2015.

[4] W. Cui, Y. Wu, S. Liu, F. Wei, M. X. Zhou, and H. Qu. Context preserving dynamic word cloud visualization. In *Proc. 3rd IEEE Pacific Visualization Symposium (PacificVis)*, pages 121–128, 2010.

[5] D. Dorling. *Area Cartograms: their Use and Creation*, volume 59 of *Concepts and Techniques in Modern Geography*. University of East Anglia, Environmental Publications, Norwich, 1996.

[6] M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th International Symposium on Computational Geometry (SOCG)*, pages 281–288, 1991.

[7] M. Gastner and M. Newman. Diffusion-based method for producing density-equalizing maps. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 101(20):7499–7504, 2004.

[8] A. Jaffe, M. Naaman, T. Tassa, and M. Davis. Generating summaries and visualization for large collections of geo-referenced photographs. In *Proc. 8th ACM International Workshop on Multimedia Information Retrieval*, pages 89–98, 2006.

[9] M. Leginus, C. Zhai, and P. Dolog. Personalized generation of word clouds from tweets. *Journal of the Association for Information Science and Technology*, 2015.

[10] H. Mousselly-Sergieh, D. Watzinger, B. Huber, M. Döller, E. Egyed-Zsigmond, and H. Kosch. World-wide scale geotagged image dataset for automatic image annotation and reverse geotagging. In *Proc. 5th ACM Multimedia Systems Conference*, pages 47–52, 2014.

[11] E. Raisz. The rectangular statistical cartogram. *Geographical Review*, 24:292–296, 1934.

[12] C. Seifert, B. Kump, W. Kienreich, G. Granitzer, and M. Granitzer. On the beauty and usability of tag clouds. In *Proc. 12th International Conference on Information Visualisation*, pages 17–25, 2008.

[13] A. Slingsby, J. Dykes, J. Wood, and K. Clarke. Interactive tag maps and tag clouds for the multiscale exploration of large spatio-temporal datasets. In *Proc. 11th International Conference on Information Visualization*, pages 497–504, 2007.

[14] W. Stein et al. *Sage Mathematics Software (Version 6.7)*. The Sage Development Team, 2015. `http://www.sagemath.org`.

[15] M. Tennekes and E. de Jonge. Tree colors: color schemes for tree-structured data. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2072–2081, 2014.

[16] M. van Kreveld and B. Speckmann. On rectangular cartograms. *Computational Geometry: Theory and Applications*, 37(3):175–187, 2007.

[17] F. B. Viegas, M. Wattenberg, and J. Feinberg. Participatory visualization with Wordle. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1137–1144, 2009.

[18] A. Wolff and T. Strijk. The map labeling bibliography. `http://liinwww.ira.uka.de/bibliography/Theory/map.labeling.html`, 2009.