EmojiCloud: a Tool for Emoji Cloud Visualization

Yunhe Feng¹, Cheng Guo¹, Bingbing Wen¹, Peng Sun², Yufei Yue³, Dingwen Tao⁴

University of Washington¹
The Chinese University of Hong Kong, Shenzhen²
Amazon³
Washington State University⁴

14th July, 2022
Motivation

EmojiCloud Design & Implementation

EmojiCloud Evaluation

Future Work
Representation Problems of Word Cloud of Emojis

Figure 1: Word cloud of emojis

- colors (⚽️❤️)
- directionalities (😂💕)
- textures (🙌❤️)
Inaccurate Representations Lead to Misunderstanding

- When emojis 🙄 👍 are upside down, they turn into 😞 👎 that conveys different sentiments and meanings.
- Miscolored emojis such as 🧐 🧘 🧝 may cause the problem of personal identity representations.
1 Motivation

2 EmojiCloud Design & Implementation

3 EmojiCloud Evaluation

4 Future Work
EmojiCloud Basic Idea

Figure 2: EmojiCloud basic idea
EmojiCloud Challenges

- Where and how to collect emoji images?
  - Emoji appearances are different across different vendors.
  - Many platform vendors exist (e.g., Twitter, Apple, Meta, and Google).
- How to determine emoji plotting sizes?
  - Emoji frequency and the canvas size.
- How to design a flexible canvas that supports various shapes?
  - Rectangle, square, ellipse, circle, and masked images.
- How to design the emoji layout?
  - Make EmojiCloud dense and beautiful.
  - Highlight the most important emojis.
### Emoji Image Retrieval

**Figure 3: Unicode full emoji list**

Source: [https://unicode.org/emoji/charts/full-emoji-list.html](https://unicode.org/emoji/charts/full-emoji-list.html)
# Emoji Image Retrieval

### (a) Emoji vendors

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appl</td>
<td>3/17/2022 11:31 AM</td>
</tr>
<tr>
<td>DCM</td>
<td>3/17/2022 11:31 AM</td>
</tr>
<tr>
<td>FB</td>
<td>3/17/2022 11:31 AM</td>
</tr>
<tr>
<td>GMail</td>
<td>3/17/2022 11:31 AM</td>
</tr>
<tr>
<td>Goog</td>
<td>3/17/2022 11:31 AM</td>
</tr>
<tr>
<td>Joy</td>
<td>3/17/2022 11:31 AM</td>
</tr>
<tr>
<td>KDDI</td>
<td>3/17/2022 11:31 AM</td>
</tr>
<tr>
<td>Sams</td>
<td>3/17/2022 11:31 AM</td>
</tr>
<tr>
<td>SB</td>
<td>3/17/2022 11:31 AM</td>
</tr>
<tr>
<td>Twtr</td>
<td>3/17/2022 11:31 AM</td>
</tr>
<tr>
<td>Wind</td>
<td>3/17/2022 11:31 AM</td>
</tr>
</tbody>
</table>

### (b) Emoji image examples

**Figure 4:** Emoji organized by vendors and emoji image examples
Emoji Image Preprocessing

(a) Raw image  
(b) Bounding box  
(c) Unoccupied pos.

**Figure 5:** Preprocessing original emoji images by determining bounding boxes and marking unoccupied pixel positions (colored as black in Figure 5(c))
Emoji Size Calculation

We use a quintuple $e = (a, b, w, E, U)$ to represent an emoji, where $a$, $b$, $w$ are the width, height, edge-level frequency weight of emoji $e$. $E_{x,y}$ and $U_{x,y} \in \{0,1\}$ represent the pixel value (RGBA) and the pixel unoccupied status at the coordinate $(x, y)$.

$$s \geq \sum_{i=1}^{|e|} w_i^2 \ast a_i \ast b_i \ast r^2 = \sum_{i=1}^{|e|} w_i^2 \ast c \ast r^2$$  \hspace{1cm} (1)

where $s$ is the drawable canvas area and $c$ is the constant area of preprocessed emojis.

- a possible maximum edge rescale ratio $r$ can be $\sqrt{s/(c \ast \sum_{i=1}^{|e|} w_i^2)}$.
- the rescaled width and height: $a'_i = a_i \ast w_i \ast r$; $b'_i = b_i \ast w_i \ast r$.
- The edge rescale ratio $r$ decays at a rate of 0.9 if there is not enough room to plot all emojis on the canvas.
Emoji Canvas Definition

- We use a quintuple \((m, n, s, C, V)\) to represent a canvas, where \(m \times n\) defines a rectangle bounding box of the canvas; \(s\) is the drawable canvas size; \(C\) and \(V\) represent pixel values and the painting eligibility.
- \(C_{x,y}\) represents canvas pixel values at the coordinate \((x, y)\).
- \(V_{x,y} \in \{0, 1\}\) indicates the painting eligibility of \((x, y)\), where \(x \in [1, m]\) and \(y \in [1, n]\). The design of \(V\) controls the drawable shape (e.g., a circle or an ellipse) on the canvas.
EmojiCloud Layout

Motivation

EmojiCloud Design & Implementation

EmojiCloud Evaluation

Future Work

Algorithm 1: EmojiCloud Layout

```plaintext
Input: n: a list of emojis, (m, n, c, V): a canvas with width m, height n, drawable size c, pixel values c, and pixel painting eligibility V on the canvas; c: the standardized size of emoji images;

Output: C: an emoji cloud image;

1. s ← sort the emoji list by emoji weights w = [w1, w2, ..., wn] in reverse order;
2. r ← \( \sqrt{1/(c + \sum w^2)} \); // rescale ratio in Equation
3. for x = 1 to m do // x coordinate of canvas
4.     for y = 1 to n do // y coordinate of canvas
5.         if Vxy = 1 then // canvas pixel is eligible for painting
6.             append (x, y) into the canvas pixel coordinate list p0; // build p0
7.         end if
8.     end for
9.     end for
10. p1 ← sort p0 by the Euclidean distance between (x, y) ∈ p0 and the canvas center (m/2, n/2);
11. count ← 0; // count of plotted emojis
12. while count < |n| do // not all emojis have been plotted
13.     if count = 0 then // no emoji has been plotted
14.         count ← 0;
15.     end if
16.     for i = 1 to |n| do // iterate the emojis sorted by weights in reverse order
17.         if wni = 0 then // the emoji is not unoccupied
18.             1. skip
19.         end if
20.         if wni > 0 then // the emoji is unoccupied
21.             for x = 1 to m do // x coordinate of emoji image
22.                 for y = 1 to n do // y coordinate of emoji image
23.                     if Vxy = 1 then // canvas pixel is not eligible for painting
24.                         if flag = True then // the emoji can be plot at (x, y)
25.                             Cxy ← Cxy + \( \sum w^2 \); // iterate temporal coordinates
26.                             Vxy ← Vxy - 1; // set painting eligibility as negative
27.                             remove (x, y) from p0; // delete (x, y) for computing efficiency
28.                             count ← count + 1; // increase the number of plotted emojis by 1
29.                         end if
30.                         if Vxy = 1 then // canvas pixel is eligible for painting
31.                             flag ← False; // no room to plot at (x, y)
32.                         break; // iterate the next (x, y) in p0
33.                     end if
34.                     end if
35.                 end for
36.             end if
37.         end if
38.         count ← count + 1; // iterate the next emoji
39.     end for
40.     end if
41. end while
42. return C; // decay the edge rescale ratio by 0.9
```

Yunhe Feng, Cheng Guo, Bingbing Wen, Peng Sun, Yufei Yue, Dingwen Tao

EmojiCloud: a Tool for Emoji Cloud Visualization
Default and Arbitrary Canvas

- We set the default canvas shape as an $m \times n$ rectangle, and all pixel coordinates within the rectangle are eligible to draw emojis. The painting eligibility $V_{x,y}$ is set as 1 for all $x \in [1, m]$ and $y \in [1, n]$.
- Users are allowed to specify arbitrary canvas drawable shapes by configuring the painting eligibility $V_{x,y}$ for pixel coordinate $(x, y)$ on the canvas.
Suppose we have a drawable ellipse area within an $m \times n$ rectangle bounding box for plotting emojis. The semi-major and semi-minor axes' lengths are expressed as $m/2$ and $n/2$. The center pixel coordinate is expressed as $(m/2, n/2)$. If pixel coordinate $(x, y)$ on canvas satisfies the following inequality, $V_{x,y}$ is set as 1.

$$\left(\frac{x - m}{m/2}\right)^2 + \left(\frac{y - n}{n/2}\right)^2 \leq 1 \quad (2)$$

Otherwise, the coordinate $(x, y)$ is outside of the ellipse, and the corresponding $V_{x,y}$ is set as 0. When $m$ equals $n$, a circle canvas is defined.
Masked Canvas

- We determine a $m \times n$ bounding box of the image by removing the surrounding transparent pixels.
- We detect the image contour and draw a boundary accordingly (e.g., 🐦 → 🦅).
  - We scan the alpha values of pixels in the preprocessed image by row and by column respectively.
  - We identify pixels that cause an alpha value change greater than a threshold $\theta$ (by default $\theta = 10$) as boundary pixels.
  - After all boundary pixels are determined, they will be colored by specified colors.
EmojiCloud Inclusive Design

- EmojiCloud is flexible and inclusive to handle emoji images designed by seven vendors (i.e., Apple, Google, Meta, Windows, Twitter, JoyPixels, and Samsung).
- Users can customize and combine emojis based on their requirements.
  - a red apple emoji (U+1F34E) 🍎 can be replaced by 🍎 for marketing campaigns.
  - The sauropod (U+1F995) 🦖 and T-Rex emoji (U+1F996) 🦕 can be combined as 🦖 if it is not necessary to distinguish dinosaur species.
Implementation and Open Source

• EmojiCloud has been published through Python Package Index (PyPI).
  
  `pip install EmojiCloud`  `from EmojiCloud import EmojiCloud`
  
• An EmojiCloud tutorial is available at
  https://pypi.org/project/EmojiCloud/.
1 Motivation

2 EmojiCloud Design & Implementation

3 EmojiCloud Evaluation

4 Future Work
Code for Different Canvases

```python
from EmojiCloud import EmojiCloud

# set emoji weights by a dict with key: emoji by codepoint, value: weight
dict_weight = {'1f1e8': 1.1, '1f4a7': 1.2, '1f602': 1.3, '1f6f4': 1.4, '1f6f5': 1.5, '1f6f6': 1.6, '1f6f7': 1.7, '1f6f8': 1.8, '1f6f9': 1.9, '1f6fa': 2.0, '1f6fb': 2.1, '1f6fc': 2.2, '1f7e0': 2.3, '1f9a2': 2.4, '1f9a3': 2.5, '1f9a4': 2.6, '1f9a5': 2.7, '1f9a6': 2.8, '1f9a8': 2.9, '1f9a9': 3.0}

# emoji vendor
emoji_vendor = 'Twitter'

# masked canvas
img_mask = 'twitter-logo.png'

canvas_w = 72*10
canvas_h = 72*4

# rectangle canvas

# ellipse canvas
```

Figure 6: EmojiCloud for plotting different canvases
Visualization on Different Canvases

(a) Rectangle  (b) Ellipse  (c) Mask

Figure 7: EmojiCloud on different canvases
Code for Different Emoji Vendors

```python
from EmojiCloud import EmojiCloud


# emoji vendors
list_vendor = ['Google', 'Windows', 'Apple', 'Twitter', 'Meta', 'JoyPixels', 'Samsung']

for emoji_vendor in list_vendor:
    # circle canvas
    canvas_w = 72*10
    canvas_h = 72*10
    saved_emoji_cloud_name = 'emoji_cloud_circle_' + emoji_vendor + '.png'
    EmojiCloud.plot_ellipse_canvas(canvas_w, canvas_h, emoji_vendor, dict_weight, saved_emoji_cloud_name)

Figure 8: EmojiCloud for plotting different vendors
Visualization for Different Emoji Vendors

(a) Twitter  
(b) Apple  
(c) Google  
(d) Windows

(e) JoyPixels  
(f) Meta  
(g) Samsung

Figure 9: EmojiCloud for different vendors
Code for Customized Emojis

```python
1 from EmojiCloud import EmojiCloud
2
3 # set emoji weights by a dict with key: emoji by codepoint, value: weight
4 dict_weight = {'1F1E6-1F1F7': 1.1, '1F1E7-1F1EA': 1.2, '1F1E7-1F1F7': 1.3, '1F1E8-1F1E6': 1.4,
   '1F1E8-1F1F4': 1.5, '1F1E8-1F1F5': 1.6, '1F1E9-1F1EA': 1.7, '1F1E9-1F1F0': 1.8, '1F1EA-1F1E8': 1.9,
   '1F1EA-1F1F8': 2.0, '1F1EC-1F1ED': 2.1, '1F1EC-1F1F7': 2.2, '1F1ED-1F1F7': 2.3, '1F1EE-1F1F7': 2.4,
   '1F1EF-1F1F5': 2.5, '1F1F0-1F1F7': 2.6, '1F1F2-1F1FD': 2.7, '1F1F3-1F1F1': 2.8, '1F1F5-1F1F1': 2.9,
   '1F1F5-1F1F9': 3.0, '1F1F6-1F1E6': 3.1, '1F1F7-1F1F8': 3.2, '1F1F8-1F1E6': 3.3, '1F1F8-1F1F3': 3.4,
   '1F1FA-1F1F8': 3.5, '1F1FA-1F1FE': 3.6, '26BD': 3.7, '1F3C6': 3.8}
5
6 # emoji vendor
7 emoji_vendor = 'Twitter'
8
9 # rectangle canvas
10 canvas_w = 72*10
11 canvas_h = 72*4
12 canvas_color = 'green'
13
14 saved_emoji_cloud_name = 'emoji_cloud_customized.png'
15 EmojiCloud.plot_rectangle_canvas(canvas_w, canvas_h, emoji_vendor, dict_weight, saved_emoji_cloud_name,
   dict_customized, canvas_color)
```

**Figure 10:** EmojiCloud for plotting customized emojis
Visualization of Customized Emojis

![EmojiCloud for FIFA World Cup Trophy](image)

(a) Original  
(b) Customized

**Figure 11:** EmojiCloud for FIFA World Cup Trophy
Using Emoji Unicode as Input

```python
from EmojiCloud import EmojiCloud

# set emoji weights by a dict with key: emoji by unicode, value: weight
dict_weight = {'Ac': 1.1, '💧': 1.2, '💕': 1.3, '🐺': 1.4, '🔥': 1.5, '😢': 1.6, '😠': 1.7, '💫': 1.8, '🤖': 1.9, '🎶': 2.0, '💡': 2.1, '깄': 2.2, '🌞': 2.3, '🔥': 2.4, 'ㅅ': 2.5, '❤': 2.6, '🐱': 2.7, '🐱': 2.8, '烝': 2.9, '🍵': 3.0}

# emoji vendor
emoji_vendor = 'Google'

# circle canvas
canvas_w = 72*5
canvas_h = 72*5

saved_emoji_cloud_name = 'emoji_cloud_circle.png'
EmojiCloud.plot_ellipse_canvas(canvas_w, canvas_h, emoji_vendor, dict_weight, saved_emoji_cloud_name)
```

Figure 12: EmojiCloud for plotting customized emojis
Running Time Evaluation

![Running time of EmojiCloud](image)

**Figure 13**: Running time of EmojiCloud
Motivation

EmojiCloud Design & Implementation

EmojiCloud Evaluation

Future Work
Future Work

- Keep updating the open-source EmojiCloud based on the users’ feedback, such as adding new functions and covering more emoji vendors.
- Provide an online EmojiCloud service via www.emojicloud.org.
- Explore the possibility of merging words and emojis in a unified word-emoji cloud.
Thank you!