Tilden Winston
Monday, May 4, 2020
Hacking for Humanists – ENGL 3500
Code: https://github.com/TildenWinston/Emojis-and-Poetry

🐳 and the Use and Applicability of Emojis to Poetry

A comparative literary analysis of *Emoji Dick* and *Moby Dick*

Emoji Dick was a project by Fred Benenson to translate the entirety of *Moby Dick* into emojis using Amazon Mechanical Turk, a micro job service that pays people to complete small, quick tasks. Every sentence of *Moby Dick* was separated out and sent to be translated into emojis. The translators only had 471 unnamed emojis to choose from. The project was successfully completed, but the question remains, how similar is *Emoji Dick* to *Moby Dick*.

To start off, I am going to run some basic literary analysis techniques on the data. I am working with the raw Mechanical Turk data which contains every sentence translated 3 times, so counts will be higher than normal. The first analysis I am running on the data is frequency analysis comparing the most frequent emoji to the most frequent words. I am doing this both with and without stop words. Emoji don't include basic words like "the" and "a" so I don't expect to have to remove any stop emoji.

Emoji													
👲	💁	👸	🚑	☕	🔑	🎓	🎡	🏬	✖	❔	👀	🐳	😠
2064	1990	1897	1890	1861	1765	1746	1712	1704	1370	1304	1130	1080	1034

With Stopwords														
the	of	and	a	to	in	that	his	it	i	but	he	as	is	with
39051	17861	17396	12611	12499	11227	8061	6938	6547	5377	4926	4848	4695	4690	4686

Without Stopwords														
whale	one	now	like	upon	man	old	ahab	ship	will	though	sea	yet	time	still
2483	2419	2113	1571	1546	1218	1199	1170	1040	1034	1020	1005	949	886	872

Looking at the tables we see that unsurprisingly, whale is the most common word in the dataset with stopwords removed, but whale comes 14th in emoji frequency. 👲, or “Person with skullcap”, comes in first with 2064 uses. The spouting whale emoji is used half as much as the word whale in the story. An initial look at the emoji table does not easily offer up a set of stopwords, but it would be an interesting problem to pursue further. Based on the brief analysis, *Emoji Dick* seems fairly distinct from *Moby Dick*.

Emoji														
👲	💁‍♀️	👸	☕	🚑	🎓	🔑	🎡	🏨	✖	❔	🐳	❕	👀	👦
743	724	669	637	626	607	598	574	556	537	511	496	442	439	438

With Stopwords														
the	of	and	a	to	in	that	his	it	i	but	he	as	with	is
14027	6401	6237	4519	4466	4045	2893	2479	2354	1922	1766	1724	1690	1679	1671

Without Stopwords														
whale	one	now	like	upon	man	old	ahab	ship	will	sea	though	yet	time	long
891	867	756	566	558	435	428	420	374	374	362	360	338	321	312

While the raw task data does not look much like *Moby Dick* it is worth considering that the final translation improved the similarity. As expected, the word frequency tables remain the same, but there is some shifting in the emoji frequency table with tea ☕ overtaking ambulance 🚑. Spouting whale 🐳 rose two places to 12th, but it is still used barely half as much as the word whale. The final translated text remains distinct from the original work.

It would be interesting to compare the emoji translations to each other. Did any one translator favor a particular emoji? Did a translator repeatedly misunderstand what an emoji represents? While we don't know what worker translate which piece, we do know 812 workers took part in the translation effort. Such analysis could prove useful, but given how short the samples are that we are working with, it is unlikely the groupings would be meaningful.

A difference between an original and translated work is almost inevitable, but the human translators leave something to be desired. Can we automate the translation process and provide better results? First, lets take a moment to discuss emojis.

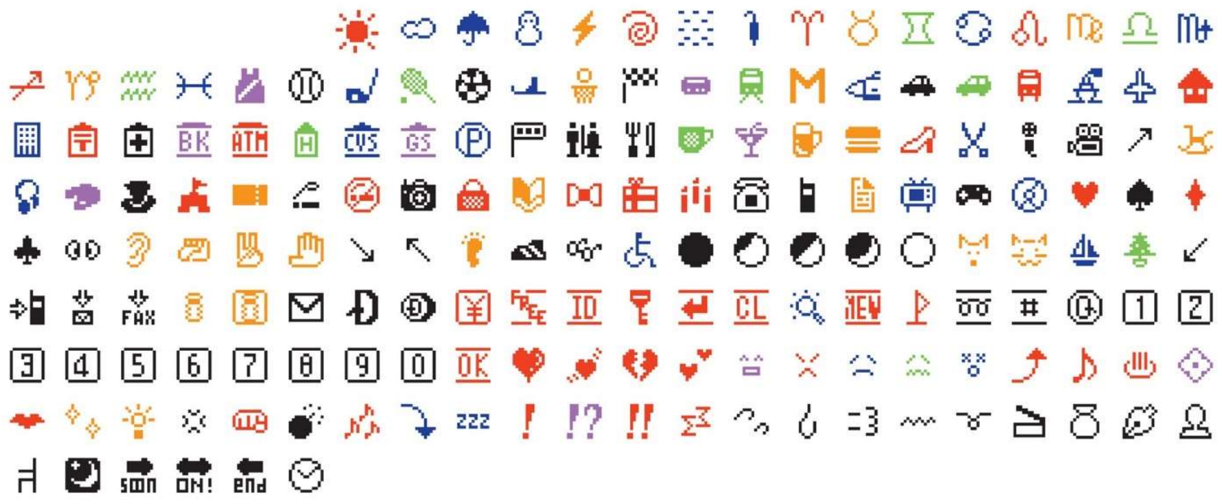


Figure 1: Shigetaka Kurita, NTT DOCOMO. Emoji (original set of 176)

What Are Emoji?

The term Emoji refers to the tiny images that speckle electronic communications. Emoji originated in Japan where the first emoji was created by Shigetaka Kurita in 1999 for DOCOMO, Japan's main mobile carrier at the time. The emoji were considered individual characters much like emoji are today. In Japan, emoji usage grew quickly through the early 2000s. It was not

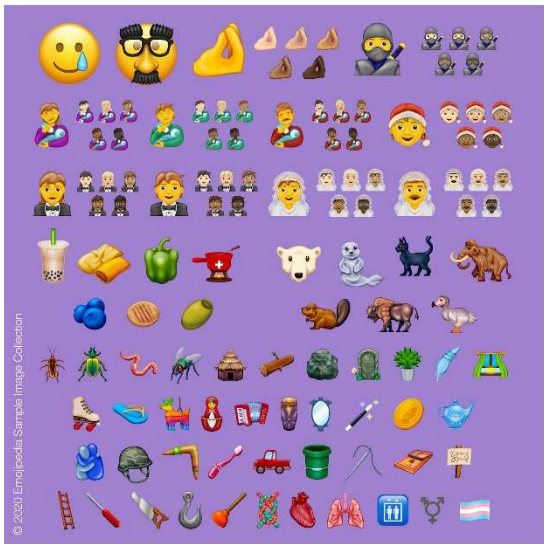


Figure 2: Emojipedia, Unicode 13.0 New Emoji

until 2010 that emoji were accepted into Unicode and standardized. Before being standardized, different companies had unique encoding methods. Apple helped bring emoji to use US in 2011 when it incorporated emoji into iOS. Android followed suit in 2013. New emoji are added to the Unicode Standard every year, as of Unicode 13.0, 3304 emoji are included in the standard set.

These images or pictograms depict faces 😀, foods 🍕, plants 🌴, and animals 🐳. Emoji are used to express meaning, add emphasis, and convey emotions. According to Brandwatch, a "social intelligence company," on average 250 million emojis are posted online every month on twitter alone. Emojipedia notes that over 700 Million Emoji are used every day in Facebook Posts. Emoji are used in relatively informal person to person communication such as on social media, via SMS, or on workplace messaging apps such as Slack, Teams, or Mattermost.

As a point of clarification, this paper is looking at emoji as being distinct from emoticons which are similar. Emoticons, often seen as the forefathers of modern emoji were gestures or faces represented by multiple characters, for example :), or ¯_(ツ)_/¯.

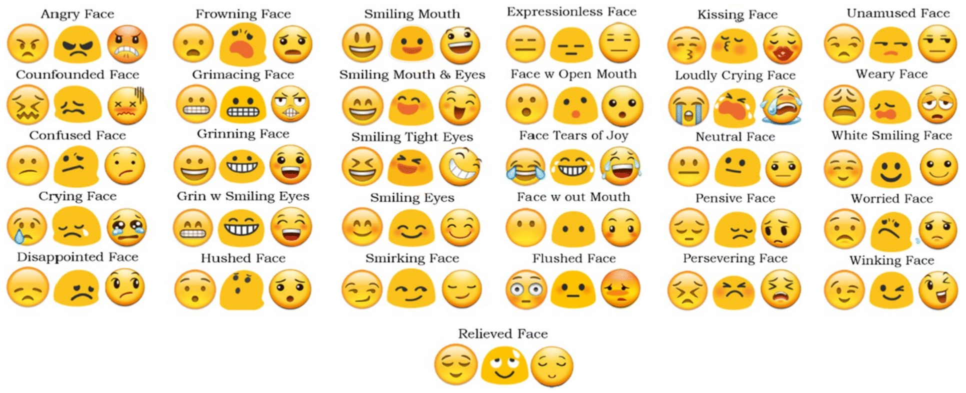
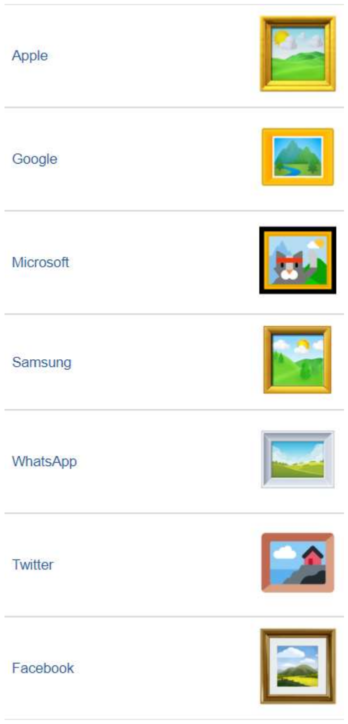


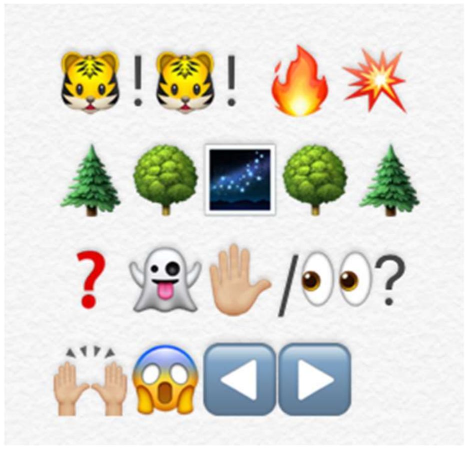
Figure 3: Apple, Anroid, and Samsung Emoji Renderings

So Emojis are standardized right? Thanks to Unicode a 🖼 is a 🖼 is a 🖼, right? Unfortunately not. As displayed in the above table, emojis are displayed differently or not at all on different platforms. For example, flags like CA don't display on windows devices. And the framed pictures shown above are slightly between devices. The difference between a framed painting of a field and a house on a cliff is minimal, the difference between a water pistol and a revolver can be much more significant.



More work has been done to standardize emoji encodings than on their visuals. Originally, when emoji were first introduced they were not cross platform, an emoji on one cellular carrier and cellphone brand couldn't be sent to someone on a different system. The adoption of the Unicode standard has made great strides towards eliminating this issue, it is still present. Take the spouting whale 🕵 emoji 🐳 for example. The Hex Code Point is e054, PHP Src is "\xee\x81\x94", and in Unicode it is U+1F433. Not all programs can work with each encoding. R seems to favor the UTF-8 Hex Bytes. Part of the challenge is to be able to take the encoding and actually turn it into a picture. On many desktop platforms such as github or skype, inputs like ":bug:" 🐛, ":shipit:" (GitHub exclusive) or ":thumb:" 👍 are used to display emoji.

Literary Analysis



1. "The Tyger" by William Blake

Tyger Tyger, burning bright,
In the forests of the night;
What immortal hand or eye,
Could frame thy fearful symmetry?

Figure 4: Paris Review Emoji Poetry Contest

By applying emoji to poetry, we are adding an additional dimension that brings the medium into modernity. Emoji can be used to add meaning and to emphasize emotion. Poetry can be done solely with emoji or as a mix of emoji and words. Poems can be written with emoji or they can be added later. Above, in Figure 3 we see an example of what poetry made solely from emojis can look like. The example is "The Tyger" by William Blake and it was manually translated by Zai Divecha. In a scenario like above where the emoji version and plain text version can be referenced side by side, the emoji version can be leveraged to help explain the poem and reveal deeper meaning. The use of emoji in poetry could also create an interesting aging effect as emoji icons change and evolve over time, similar to how the definitions and uses of words change. Much like how words vary regionally and by dialect, emoji vary platform to platform with each having its own distinct dialect.

The Process

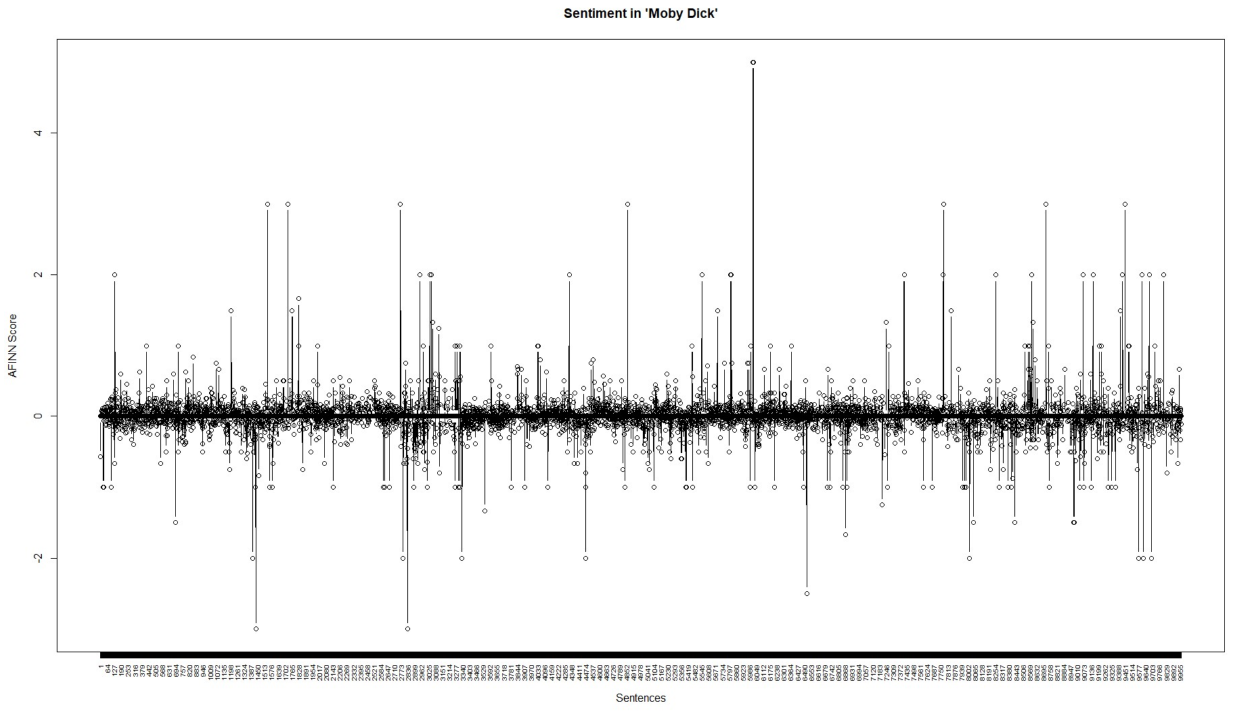
My process began with cleaning the *Emoji Dick* data, then I did basic analysis. Subsequently, I tackled the easier data set and read in the keywords. Once the keywords were imported, I took in the plain text. The keywords were run against the plain text with translations being made during the process. Results were then exported to a text file where emoji are better supported.

Data Cleaning

id	input_sentence	translation	number_of_emoji
2	MOBY DICK; OR THE WHALE By Herman Melville		0
3	MOBY DICK; OR THE WHALE By Herman Melville	E009;E037;E11E;E148;E14B;E318;E41E;E517;	8
4	MOBY DICK; OR THE WHALE By Herman Melville	E044;E32C;E324;E319;E311;E311;E13A;E24D;	8
5	CHAPTER 1.	E00E;E21C;E211;E00E;E00E;	5
6	CHAPTER 1.	E328;E01A;E239;E240;E339;E52E;	6
7	CHAPTER 1.	E21C;E337;E337;E337;E00E;	5

To start, I had to clean my data. The *Emoji Dick* data was originally stored in an SQL database, so my first step was to extract it to a simpler to work with data format. I hosted the database locally, imported it, and then used QueryPie, an SQL database GUI to look at the data. From QueryPie I was able to export all of the tables into Excel spreadsheets. Little did I realize; CSVs are easier to work with on R. From Excel I converted the amt_results.xlsx and the final_translation.xlsx into their respective CSVs. The data conversion fun was not over because the translated emoji were stored as hex code points. Hex code points are difficult to deal with so I needed to convert to Unicode. Finding a conversion table was difficult, but I managed to find a wiki entry on GitHub. From there, I read the CSV into R, then I read in the translation file. I turned the conversion file into a list, which allowed me to grep for a hexcode and know the translated value is at the second index of the vector at the index returned by grep. In other words, grep returns the index of a list, and then I know where in that list the translation is. I then ran through the csv converting each of the hex codes in the translation column into Unicode. To finish cleaning the data, I saved the data.frame back to a csv. While not strictly necessary, the translated files will be more useful in the future than their untranslated counterparts.

Basic Analysis



I performed basic word frequency analysis looking at the final translation and the raw dataset. Word counts both with and without stop words were evaluated. Following this I ran sentiment analysis on every sentence in the novel. Eventually I hope to use this to inform emoji selection.

Keywords input

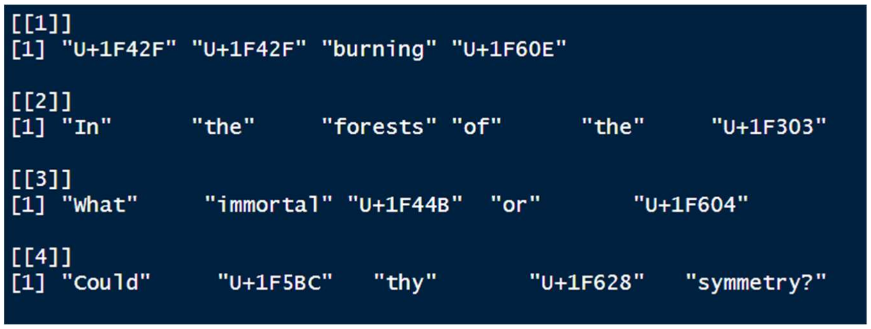
The emoji keywords are read in from a csv file to a data frame. The keywords are separated by “|” characters so I used a for loop to split the keywords on the pipe, trim white space, and return the data to the data frame. The data is put into column 6, a new one created for the purpose. To search this data frame for words, I used grep to find occurrences and regex to minimize false positives. When searching the column, the contents of each cell behave much like a character vector of length one, so different regex could be used.

Input

Texts for translation are input as text files. I scan the file into a vector spitting on new lines. I then run the vector through strsplit which gives me a list of vectors allowing me to keep track of line breaks.

Keyword translation

The next step is to run all words and group of words through searching the keyword vector. The keywords can actually be multi-word phrases so I needed to get all consecutive groupings of N letters. In my list of keywords, the longest phrase was 5 words long so that is the longest phrase I had generated. I realized after writing this section that textcnt() in the Tau library does essentially the same thing. Each of the text segments had punctuation removed before being pasted together and passed into the search. The search returns the Unicodes of all emoji associated with these key words. I noticed that many plural words were being missed so I added a feature that performs basic plural detection and S removal. Once Unicodes were found for a word, I decided to replace that word or group of words with the Unicode for the



```
[[1]]
[1] "U+1F42F" "U+1F42F" "burning" "U+1F60E"

[[2]]
[1] "In"      "the"     "forests" "of"      "the"     "U+1F303"

[[3]]
[1] "What"     "immortal" "U+1F44B"  "or"       "U+1F604"

[[4]]
[1] "Could"     "U+1F5BC"   "thy"       "U+1F628"   "symmetry?"
```

representing emoji. In doing so I had to overcome a challenge making sure I didn't have duplicate codes when phrases were found. Almost as bad were the issues that arose from only changing one of the words of a found phrase. If they were not properly removed the words would continue getting searched for in the keyword vector during subsequent runs. The solution was to rebuild the line minus the word or phrase. The end result of this was a list composed of normal words and Unicode.

Displaying Emoji and Output

When choosing this project, I was aware that reading and writing emoji with R was going to be a challenge, but I managed to underestimate just how much of a challenge it was going to be. The short of it is that I needed a way to take the encoding data for an emoji and display the emoji. Rstudio cannot handle or display emoji which made the task more difficult. I decided that to get around this limitation, I would write results to a text file which is able to properly display the emoji. Writing a more normal Unicode character such as "ỏ" to the text was fairly easy using iconv and cat, but I needed a way to do it from the Unicode. In the end, the solution I found converted the Unicode to UTF-8, a text encoding, and then writing that out to the file. While this works for most outputs, some emoji are actually multiple Unicodes, for example 1️⃣ is represented as U+31, U+20E3. My method for outputting the Unicode to a file does now support these emojis and it is a known limitation.

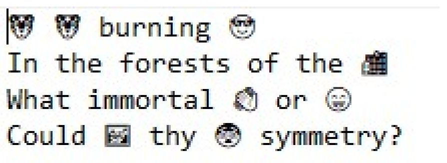
To write the text list to a text file I run it through nested for loops much like when checking for keywords. This time, I added an if statement that uses grepl to check if the word is a Unicode. If it is, it properly wraps it in UTF-8 and adds it to the output vector. After each time the inner for loop finishes, a new line character is added to retain as much of the original formatting as possible. The output vector is then written to the output.txt file where the emojis can be viewed for the first time. I personally recommend copying and pasting from the text file into a word document due to notepad’s limitations when displaying emojis. Below on the left is how the poem displays in word while on the right is an image of the same text in notepad.

🐯 🐯 burning 😎

In the forests of the 🌃

What immortal 👋 or 😄

Could 🖼 thy 😨 symmetry?



```
🐯 🐯 burning 😎
In the forests of the 🌃
What immortal 👋 or 😄
Could 🖼 thy 😨 symmetry?
```

Points of improvement

In the end I was not able to translate full poems or pieces of literature into emojis, but I had success replacing based on keywords. The next step would be to replace the remaining words based on their sentiment and the sentiment assigned to each emoji. My plan is to use the raw data to assign this.

Another point that could be improved is when a keyword matches multiple emoji. Ideally, the emoji could be selected based on sentiment from the vector of possibilities. Currently, it is set so the first emoji in this vector is always chosen.

As a final improvement I should add a feature that normalizes spellings of input texts and adds synonyms to the keyword lists. When translating The Tyger by William Blake, I had to manually correct the spelling.

Sonnet XVI

But wherefore do ⛔ you 🅰️ mightier way
Make war upon this bloody tyrant, Time?
And fortify yourself in your decay
With means more blessed than my barren rhyme?
Now 🧍 you 🔛 the 🎩 of 🙋 hours,
And many maiden 🏡 yet unset
With virtuous wish would 🐻 your living 💐
Much liker than your painted counterfeit:
So should the lines of 🧬 that 🧬 repair,
Which this, Time's 📝 or my pupil 🖋️

Neither in inward worth nor outward fair,

🥫 make you live yourself in 🤩 of men

To give away yourself keeps yourself still,

And you must live, drawn by your own 🍠 🤹

Bibliography

🖼️ *Frame with Picture Emoji*. *emojipedia.org*, /framed-picture/. Accessed 4 May 2020.

"(2) Emojipedia 📙 on Twitter: '📝 Now Approved: 117 New Emojis for 2020 #Emoji2020 Https://T.Co/SojQuXZvv6 Https://T.Co/SHp7GDsSal' / Twitter." *Twitter*. *twitter.com*, https://twitter.com/Emojipedia/status/1222615557524115459. Accessed 2 May 2020.

Bich-Carrière, Laurence. "Say It with [A Smiling Face with Smiling Eyes]: Judicial Use and Legal Challenges with Emoji Interpretation in Canada." *International Journal for the Semiotics of Law - Revue Internationale de Sémiotique Juridique*, vol. 32, no. 2, June 2019, pp. 283–319. *Springer Link*, doi:10.1007/s11196-018-9594-5.

Emoji Counts, V13.0. https://unicode.org/emoji/charts/emoji-counts.html. Accessed 2 May 2020.

"Emoji Use in the New Normal." *Emojipedia*, 1 May 2020. *blog.emojipedia.org*, https://blog.emojipedia.org/emoji-use-in-the-new-normal/.

"Facebook Reveals Most and Least Used Emojis." *Emojipedia*, 13 July 2018. *blog.emojipedia.org*, https://blog.emojipedia.org/facebook-reveals-most-and-least-used-emojis/.

Franco, Courtny L., and Jennifer M. B. Fugate. "Emoji Face Renderings: Exploring the Role Emoji Platform Differences Have on Emotional Interpretation." *Journal of Nonverbal Behavior*, vol. 44, no. 2, June 2020, pp. 301–28. *link.springer.com*, doi:10.1007/s10919-019-00330-1.

Galloway, Paul. "The Original NTT DOCOMO Emoji Set Has Been Added to The Museum of Modern Art's Collection." *Medium*, 2 Apr. 2019. *stories.moma.org*, https://stories.moma.org/the-original-emoji-set-has-been-added-to-the-museum-of-modern-arts-collection-c6060e141f61.

"Iamcal/Emoji-Data." *GitHub*. *github.com*, https://github.com/iamcal/emoji-data. Accessed 2 May 2020.

Novak, Petra Kralj, et al. "Sentiment of Emojis." *PLOS ONE*, vol. 10, no. 12, Public Library of Science, Dec. 2015, p. e0144296. *PLoS Journals*, doi:10.1371/journal.pone.0144296.

Rankin-Gee, Nadja Spiegelman and Rosa. "Emoji Poetry Contest." *The Paris Review*, 18 Aug. 2017. *www.theparisreview.org*, https://www.theparisreview.org/blog/2017/08/18/emoji-poetry/.

---. "Emoji Poetry Contest, Part 2." *The Paris Review*, 31 Aug. 2017. *www.theparisreview.org*, https://www.theparisreview.org/blog/2017/08/31/emoji-poetry-contest-part-2/.

Sampietro, Agnese. "Emoji and Rapport Management in Spanish WhatsApp Chats." *Journal of Pragmatics*, vol. 143, Apr. 2019, pp. 109–20. *ScienceDirect*, doi:10.1016/j.pragma.2019.02.009.

Snelgrove, Xavier, and Whirlscape CTO. *Dango - Your Emoji Assistant*. *getdango.com*, https://getdango.com. Accessed 2 May 2020.

Sonnet XVI. http://shakespeare.mit.edu/Poetry/sonnet.XVI.html. Accessed 4 May 2020.

"The Complete History of Emoji." *Wired*. *www.wired.com*, https://www.wired.com/story/guide-emoji/. Accessed 2 May 2020.

"The Emoji Report." *Brandwatch*. *www.brandwatch.com*, https://www.brandwatch.com/reports/the-emoji-report/view/. Accessed 2 May 2020.