Beyond Just Text: Semantic Emoji Similarity Modeling to Support Expressive Communication 👫 📲 😃

HENNING POHL, CHRISTIAN DOMIN, and MICHAEL ROHS, University of Hannover

Emoji, a set of pictographic Unicode characters, have seen strong uptake over the last couple of years. All common mobile platforms and many desktop systems now support emoji entry, and users have embraced their use. Yet, we currently know very little about what makes for good emoji entry. While soft keyboards for text entry are well optimized, based on language and touch models, no such information exists to guide the design of emoji keyboards. In this article, we investigate of the problem of emoji entry, starting with a study of the current state of the emoji keyboard implementation in Android. To enable moving forward to novel emoji keyboard designs, we then explore a model for emoji similarity that is able to inform such designs. This semantic model is based on data from 21 million collected tweets containing emoji. We compare this model against a solely description-based model of emoji in a crowdsourced study. Our model shows good performance in capturing detailed relationships between emoji.

CCS Concepts: • **Human-centered computing** → **Keyboards**; **Text input**; • **Information systems** → *Clustering*; *Texting*

Additional Key Words and Phrases: Emoji, mobile text entry, emoticons

ACM Reference Format:
Henning Pohl, Christian Domin, and Michael Rohs. 2017. Beyond just text: Semantic emoji similarity modeling to support expressive communication 👫 📲 😃. ACM Trans. Comput.-Hum. Interact. 24, 1, Article 6 (March 2017), 42 pages.
DOI: http://dx.doi.org/10.1145/3039685

1. INTRODUCTION

For more and more users, the mobile phone is their primary, or even only, computing device. On *Facebook*, e.g., mobile-only users make up a growing percentage of users, currently already accounting for more than 50% of their monthly active users.[1] Facebook is but one of many messaging and social networking applications, that, overall, dominate the rankings for most used applications on mobile devices [Church et al. 2015]. As such, a critical aspect of mobile systems is how they can support the expression and creativity of their users, enabling them to connect with those dear to them.

Text input is a dominant aspect of this expression and has hence been a research focus for many years. For example, researchers have designed a large number of input methods to optimize text entry speeds (for a comparison, see, e.g., Kristensson and Vertanen [2014]). But text input is not necessarily restricted to actual text. Instead of using characters to compose words, they can also be repurposed in *emoticons*, such

[1]http://venturebeat.com/2016/01/27/over-half-of-facebook-users-access-the-service-only-on-mobile/.

Authors' addresses: H. Pohl (corresponding author), C. Domin, and M. Rohs, Human-Computer Interaction Group, University of Hannover, Appelstraße 9A, 30451 Hannover, Germany; emails: {Henning, christian.domin, michael.rohs}@hci.uni-hannover.de.

ACM 1073-0516/2017/03-ART6 $15.00
DOI: http://dx.doi.org/10.1145/3039685

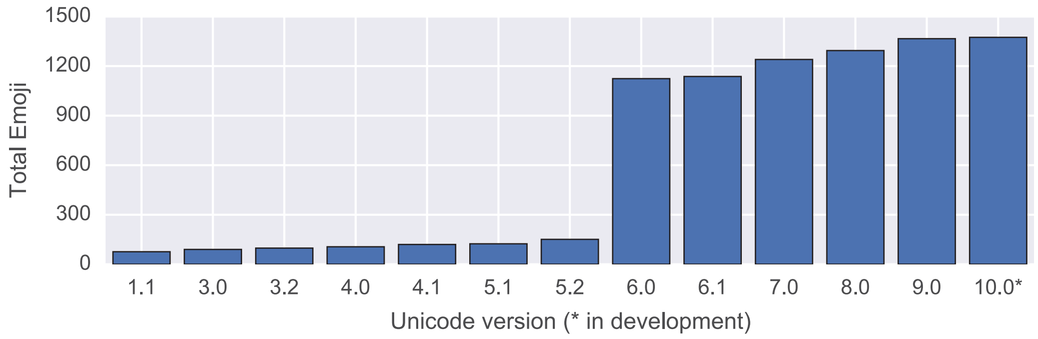


Fig. 1. Emoji were first specified in version 6.0 of the Unicode standard (with some characters retroactively promoted to emoji). Since then their number has continuously grown. The total given here is a conservative number as it does not include every possible combination of compound emoji. For a list of included emoji per version, see http://emojipedia.org/unicode-6.0http://emojipedia.org/unicode-[VERSION].

as `:)`, or `<3`. Here, characters are put together in a way that disposes of their actual meaning and makes use of their *look* to assemble larger shapes. Hence, the colon turns into a set of eyes and the parenthesis becomes a mouth.

In the Western world, emoticons traditionally mostly use a small number of punctuation characters. But non-Western writing systems offer many characters that enable more complex and expressive emoticons, such as `¯\_(ツ)_/¯` (shrugging emoticon). While this potentially allows users to assemble intricate messages, this is not straightforward on current mobile phone keyboards that do not generally provide access to arbitrary Unicode characters. Emoticons have filled a need for a more casual [Pohl and Murray-Smith 2013] and playful form of communication (e.g., adding a `;)` to a text), but can sometimes be hard to use, especially on mobiles. As recently shown by Janssen et al. [2014] emoticon use in chats does indeed increase *perceived intimacy* between chat participants. This further underlines how there is inherent value in adding emotional cues to ones textual communication.

This desire for expression beyond just text is likely what has also been driving adoption and use of emoji.[2] Where emoticons assemble "pictures" from characters, with emoji, each character is itself pictographic. Instead of sending `:D`, users can then send 😀. While still just text, this character is usually rendered as a colorful visual icon. Emoji not just allow for the expression of many emotional states (e.g., 😉, 😓, or 😱) but also enable users to decorate messages (e.g., 🌟, 💯, or 💠), or replace words with visual stand-ins (e.g., 🍆, 🐳, or 🚄). Using a visual icon instead of a word enables users to introduce ambiguity and playfulness where they see fit. Hence, this means that emoji meaning is fluid and subject to contextual and cultural (as shown for pictograms by Cho and Ishida [2011]) interpretation. It is this malleability that makes emoji attractive from an expressive point of view, but also makes it hard to organize emoji into a set keyboard layout.

Emoji have been growing both in popularity and number over the recent years. As shown in Figure 1, new emoji continue to be introduced.[3] Current mobile devices all make a large number of these characters available for users to enter with dedicated emoji keyboards. Yet, while the number of emoji has been growing, those keyboards have stuck with one input mechanism: selecting emoji from large, scrollable lists. However, as we will show in this article, this approach is slow and prone to user

[2]A set of pictographic Unicode characters: http://www.unicode.org/emoji.
[3]For a full list, see http://www.unicode.org/emoji/charts/full-emoji-list.html.

confusion. But while data and methods are in place to optimize non-emoji keyboards, there is currently nothing available to do the same for emoji entry. In this article, we investigate emoji similarity modeling as a method to support the design of future emoji input methods.

Being able to compute the level of similarity between two emoji allows clustering and organization of them. In contrast to keyboards for Latin scripts, where it is important to optimize for *sequences* of keys, emoji keyboards need to be optimized for search. Being able to place related emoji close to each other is then a way to aid the user in said search. For example, when searching for 🍔, it is sensible to expect this emoji to be close to others like 🍟. On the other hand, if one sees ☢, one could expect hamburgers to not be nearby. This structuring of the emoji presentation is particularly necessary as the large number of emoji makes it unfeasible to memorize all emoji locations. Instead, a more realistic approach is guided search, which requires the capability to organize or subset this search space—something we contribute in this article.

In the remainder of this article, we first take a closer look at the state of the art by presenting an introduction to emoji, a quantitative exploration of emoji use (based on a large number of scraped tweets with emoji from Twitter), and an evaluation of the current Google emoji keyboard. The results from the study of the Google emoji keyboard show that emoji entry can be slow, but more importantly that search is a critical problem of emoji entry. This finding motivates us to focus on the aspect of emoji layout to support users in this search. To guide emoji arrangement, we then build a model for emoji similarity. Our model makes use of semantic information, gathered from the tweets we collected earlier. We compare this model to a purely description-based model in a crowdsourcing study. Our results show significant correlation between human raters and both models. Instead of manually designing emoji input methods, future designs can thus also be informed by such models. By making it easier and more convenient to enter emoji, input methods would directly support users' ability for more personal and playful expression.

2. AN INTRO TO EMOJI

Though in some instances they supplant words entirely, they also open up new vistas of exchange and creativity. They are in a sense, the words that got away, and then returned. As smiles and frowns and jetliners. [Lebduska 2015]

In the late 90s, the first emoji were created in Japan for *NTT DoCoMo*. Emoji allowed sending small pictograms to other phones by only transmitting two bytes—the corresponding character code. The Japanese origin of emoji still manifests itself in emoji such as 🏣 (Japanese post office), 🗾 (outline of Japan), 🎏 (*Koinobori* wind socks—flown during *Children's Day* celebrations in Japan), or 🎍 (*Kadomatsu* decoration used for Japanese New Year). Other Japanese carriers followed suit and several partially overlapping sets of characters were in use for a while. However, this situation posed problems when designing for interoperability beyond Japan, e.g., with email systems. To resolve this situation, the Unicode consortium in 2010 standardized 722 emoji in version 6.0 of the Unicode standard, while also promoting many earlier characters[4] to the status of emoji as well. In addition to symbols from the Japanese carriers, this included characters from *Zapf Dingbats* (e.g., ♠ or ☎), *Microsoft's Wingdings* font (e.g., 🐈 or ⛱), and Japanese TV symbols (*Association of Radio Industries and Businesses* set, e.g., ⛄ or ⛈). In fact, many symbols were part of multiple original sources and were merged to one Unicode code point. With Unicode standardization, interoperability

[4]For an overview of emoji sources see http://www.unicode.org/emoji/charts/emoji-versions-sources.html.

between systems was secured—a necessary prerequisite for the rise of emoji to broad popularity. Unicode allows proposals for additional emoji and hence the set has grown over the years. For example, Unicode version 9.0 is bringing emoji such as 🤳 (*Selfie*), 🦆 (*Duck*), and 🛶 (*Canoe*).

2.1. Uptake In Emoji Usage

Data on emoji uptake are available from Instagram and Twitter. Instagram saw a sharp rise in emoji usage from 0% of texts using emoji to 20% of them within less than half a year of the introduction of the iOS emoji keyboard.[5] Currently, about 40% of Instagram messages contain emoji, and this number is even higher in some markets (e.g., more than 60% in Finland). Twitter has reported data on the usage of emoji in TV-related tweets between April 2014 (when emoji were introduced on Twitter) and July 2015.[6] In that timeframe, the share of TV-related tweets containing emoji grew from 9.8% to 14%. This percentage highly varies by genre and, e.g., up to 22% of tweets on music programming contain emoji. Twitter also found that younger and female users are more likely to include emoji in their tweets—hinting at why tweets on sport talk shows are least likely to include emoji (only 4% do). Of course, uptake of emoji in social media and messaging is not representative of overall frequency of emoji in other forms of writing. Likely, much fewer emoji make their way into essays, annual reports, or news articles. Yet, personal communication is an important area and, as shown above, supporting emoji use here is an aspect of growing importance.

Growth of emoji popularity also shows in how much attention they receive. For example, *Oxford Dictionaries* prominently made an emoji, "😂," their *Oxford Dictionaries Word of the Year 2015*.[7] While emoji allow users more visual expression in their messaging, that same quality is also attracting advertisers. One mobile marketing company, e.g., saw an 777% increase of emoji usage in campaigns running on their platform.[8]

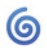
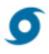
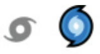
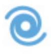
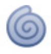
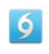
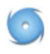
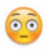
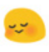
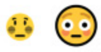
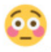
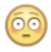
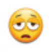
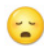
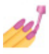
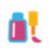
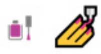
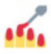
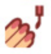
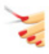
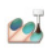
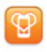
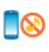
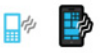
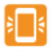
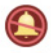
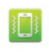
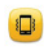
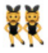
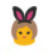
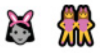
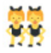
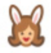
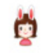
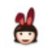
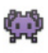
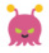
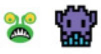
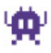
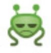
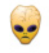
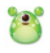
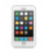
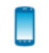
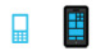
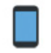
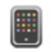
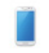
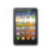
2.2. The Nature of Emoji

What makes emoji special as a means of adding visuals to texts is that they *are* text. Instead of sending images of smileys or airplanes, characters representing them are transmitted (they form a logographic writing system). Hence, in contrast to images, they can be used in places such as URLs, email subjects, or usernames. The Unicode standardization only defines a mapping between a character code and an abstract emoji description. It is up to individual platforms to provide fonts that render the individual emoji as a graphical representation (note that some emoji, such as ‼️, can also optionally be rendered as text: !!). Hence, emoji can look different on different systems and even between versions of the same system. This can be problematic where graphical representations strongly differ, a problem just recently investigated by Miller et al. [2016]. Table I shows several such examples wherein emoji vary so much in appearance that there could be misunderstandings between users of different platforms.

But the fact that they are text also allowed them to spread at the speed that they did. After all, emoji are not the first instance of visually augmented texts. Instant messengers like *Yahoo! Messenger* have for a long time supported inline smileys. Similarly, common forum software, like *phpBB*, have their own smileys. With no standard encoding for those smileys though, there was no interoperability. Forwarding a forum post via instant message would mean losing the embedded smileys. Emoji, however,

[5]http://instagram-engineering.tumblr.com/post/117889701472/emojineering-part-1-machine-learning-for-emoji.
[6]https://blog.twitter.com/2015/emoji-usage-in-tv-conversation.
[7]http://blog.oxforddictionaries.com/2015/11/word-of-the-year-2015-emoji/.
[8]http://venturebeat.com/2016/03/24/marketers-might-over-doing-it-with-all-of-the-emojis/.

Table I. While Most Emoji Look Similar on all Major Platforms, They Each Have Their Own Unique Style
Some emoji, as shown here, differ considerably between platforms which could lead to unintended interpretations in cross-platform messaging. For example, some vendors chose to represent the *nail polish* emoji not as the *object* but instead as the *action* of applying it. This could introduce misunderstandings in messages such as *"Went to get some ."* More severe inconsistencies are, e.g., exhibited by some vendors representing *alien monsters* with the classic video game sprite , or showing two females dancing where the standard calls for one *"woman with bunny ears."* But vendors also use the chance to leave their personal mark. The mobile phone emoji, for example, shows phone designs by the respective companies. Note that we show two versions of the microsoft emoji to highlight changes even within one vendor's designs.

Official name	Apple	Google	Microsoft	Twitter	Facebook	Samsung	LG
Cyclone							
Flushed face							
Nail polish							
Vibration mode							
Woman with bunny ears							
Alien monster							
Mobile Phone							

now allow copying text with smileys freely between systems. This does not necessarily mean the emoji will show up on each system, though (users browsing the web on an older phone without emoji support would, e.g., just see empty boxes).

The textual nature of emoji, in a sense, allows them to *"sneak into"* applications. If an application renders text through an API like Microsoft's *DirectWrite*,[9] emoji characters are automatically rendered correctly (if supported on the specific platform). Thus, while supporting custom smileys adds additional work, applications that can handle text essentially get emoji capabilities for free. In this case, no app-specific input method needs to be designed, but emoji entry is done via the system's keyboard.

2.3. A Combinatorial Explosion of Emoji

While many emoji are defined as a $1 : 1$ character code to pictogram relations, others break this pattern. For example, the Unicode standard specifies flags emoji, which are not each encoded as distinct characters. Instead, regional indicator letters are combined to spell out country codes that are then supposed to be rendered as flags: $+$ $\Rightarrow$. This approach makes the standard more flexible, as no list of flags needs to be updated with changing geopolitics. Which codes define a flag emoji is delegated to *ISO 3166*, specifically the two-letter codes defined in *ISO 3166-1 alpha-2*. There are currently 249 officially assigned country codes, yet support varies a lot between devices. While , , , , , , , , , and are commonly supported, some systems show no flags at all (e.g., Windows Phones), while others support many more (e.g., current versions of iOS). This includes non-country flags (such as) and proposed extensions for regional flags[10] (such as a flag for Wales:). The large number of possible flags is bound to result in long blocks of visually similar emoji on keyboards—an aspect that is already an issue on some platforms.

[9]https://msdn.microsoft.com/en-us/library/windows/desktop/hh802480(v=vs.85).aspx.
[10]http://www.unicode.org/review/pri299/pri299-additional-flags-background.html.

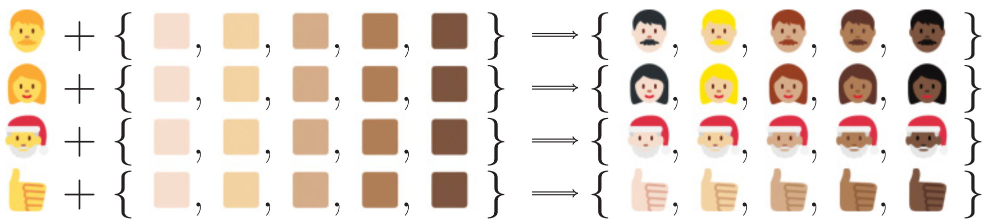


Fig. 2. Since Unicode Version 8.0, the standard specifies that emoji showing people should have a generic color (such as yellow, blue, or gray). Those emoji can then be combined with one of five skin tone modifiers to produce a more diverse set of emoji.

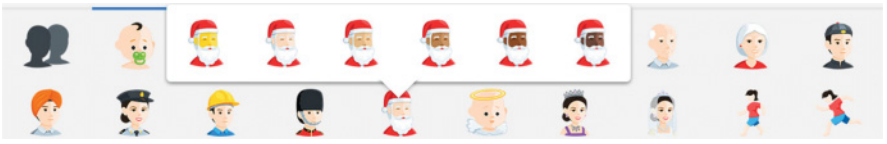


Fig. 3. To accommodate the large number of skin tone variant emoji, emoji keyboards have changed to pop-up selection menus for corresponding emoji when touched for a short while. Users can then pick a skin color from the overlayed menu.

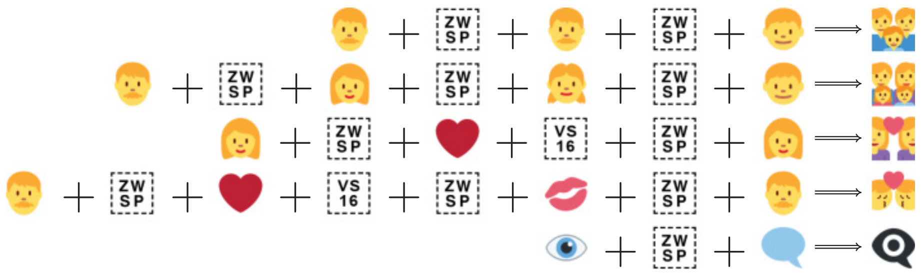


Fig. 4. Some emoji can be combined into groupings. Joining multiple person emoji with ZWSP (U+200D: zero width joiner), e.g., results in family emoji. The same approach is used to generate couple and kissing emoji for all gender pairings. Here, a VS16 (U+FE0F: variation selector-16) character is used to force emoji style for the heart character (hearts can also be rendered as monochrome text). All this can potentially be combined with skin tone modifiers. This mechanism has also been exploited to create non-standard emoji. The last row, e.g., shows an emoji Apple included for the *"I Am A Witness"* anti-bullying campaign (see http://www.wired.com/2015/10/i-am-a-witness-emoji-ios-9/).

In an approach similar to regional indicator letters, Unicode version 8.0 brought the introduction of skin tone modifiers. While the emoji standard did initially not specify skin color, most platforms rendered people emoji (such as 👱‍♀️) with white skin color. Skin tone modifiers now allow (depending on platform availability) changing the appearance of emoji to one of five levels (Type I and Type II are combined in one level) on the Fitzpatrick [1988] scale (see Figure 2). Along with this change, all major platforms have moved to neutral color (e.g., yellow) people emoji when no modifier is used (such as 👱‍♀️). This inclusion of skin tone brought a large increase in available emoji. However, instead of including those emoji directly in the overall emoji list (as with flags), all current platforms chose to make skin tone selection a separate interaction (hold down on emoji and select variant from popup, see Figure 3). While this somewhat limits growth of the emoji list, it also makes skin tone variants less discoverable and take more time to access.

The standard also allows assembling couple and family emoji to, e.g., allow sending an emoji of a gay couple with two daughters (see Figure 4). This can be combined with skin tone modifiers to represent interracial families (the Windows 10 *Anniversary Update* supports more than 52,000 such combinations[11]). While Unicode thus allows to specify arbitrarily complex family groupings, there is no guarantee these characters would be rendered as one emoji. Those changes to the standard brought some diversity to emoji, but also further increased the number of emoji. The general approach—to define emoji as a combination of several other characters—however is recurring in the standard (e.g., 5️⃣ is a combination of the code points for *5* and that of a box). While this allows for many combinations, it also means the number of emoji to choose from is potentially very large.

2.4. Common Usage of Emoji

As we have seen, emoji use in textual communication is growing. Yet, at the same time, the number of emoji itself is expanding rapidly through inclusion of additional symbols, and also through the introduction of emoji modifiers. What is available to users thus is currently a moving target. It remains to be seen how adoption and user behavior change once the larger set of emoji sees wider availability. However, we can make some observations on how emoji are currently used, based on the many messages we inspected during work on this article (for a quantitative view, see Section 3). We observed the following five different patterns of use.

Decorative use. Here, emoji are used as a sort of flourish, or decoration, for accompanying text, yet are not an integral part of it. For example, emoji can be used when congratulating: *“Happy birthday! 🎂🎁.”*

Stand-in use. Here, an emoji replaces an actual word, such as in: *“Out for a 🚬.”*

Emotional use. Instead of just decorating a message, emoji can be used to change the tone or meaning of a message. One example is sarcasm, such as in: *“Sure, go ahead 😠.”* Another example is communicating feelings about something, such as in: *“Got my test results 😃,”* which would be a very different message when ending with a 😠.

Reaction use. Here, the emoji stands on its own and communicates a direct reaction to a previous statement, such as: “👍” (*alright*). This is mostly used in chat conversations.

Stand-alone use. This is a generalization of *reaction use* for messages that contain only emoji. This presumes either familiarity of the recipient with this kind of use (similar to use of texting abbreviations) and/or context. In mid-December, a user might, e.g., send: “😰🎄🎅🎁” (∼ *I’m stressed out by Christmas shopping*).

3. QUANTIFYING EMOJI USAGE

So far, we have given a general overview on emoji and their use. However, ultimately we would like to have data to reason about and work with emoji. We thus set out to collect a large amount of real-world data on emoji and how they are used. An ideal candidate would be instant messaging logs, but those pose privacy problems and are also not generally available. Instead, we turn to publicly available data and collected a large number of tweets containing emoji. Using data from Twitter has several advantages over other data sources: (1) it is available via an easy-to-access API, (2) in contrast to forums, which usually concentrate on one topic, tweets cover a much wider range of topics and use, from casual communication between friends, to curated marketing by social media experts, and (3) there is a large amount of data, with roughly several thousand tweets send out per second.

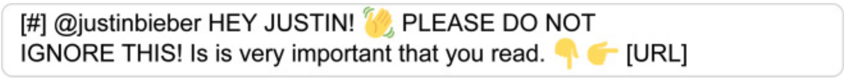
[11]http://blog.emojipedia.org/diverse-emoji-families-come-to-windows/.

There is also previous work on emoji in Twitter data that validates this approach. For example, Suttles and Ide [2013] analyze tweet sentiment analysis and explicitly include emoji. Vidal et al. [2015] concentrated specifically on food-related emotion expression, as they analyze emoticons and emoji in a dataset of 12,600 tweets containing the words *breakfast*, *lunch*, *dinner*, or *snack*. Instead of assessing the sentiment of a complete message, Novak et al. [2015] try to quantify the sentiment of individual emoji and other symbols. They used human sentiment ratings of about 70,000 tweets to rank 751 emoji and other symbols from most negative to most positive sentiment. They also find that more frequently used emoji have significantly more positive sentiment than less frequently used ones. While those papers use tweets to infer sentiment, we instead use tweets to first investigate use of and then establish similarity between emoji.

As the amount of emoji in use is constantly changing (device updates bring new emoji), we decided to limit our data collection to a well-defined set of emoji. We hence only collected tweets containing one of the 845 *"level 1"* emoji, i.e., those that are *"commonly supported as emoji by vendors at present"*[12] (note that this definition has just recently been removed from the report). This is a subset of the emoji specified by the Unicode 7.0 standard. While we limit ourselves to this well-defined subset, there are no technical roadblocks to extending this methodology to more emoji. However, including very new emoji could introduce a bias as only very few devices support them and their use would thus mostly be restricted to early adopters.

In total, we collected almost 21 million tweets over a period of about 29 days in July/August 2015 via the Twitter public streaming API.[13] To limit the number of tweets, we only collected tweets for 3 minutes at a time, or until 10,000 tweets were gathered—whichever came first. We use Twitter's keyword filtering to only gather tweets with emoji. However, as Twitter only allows specification of up to 400 keywords per request, we split the set of emoji in three equal-sized blocks, randomized block order within a run, and scraped each of those blocks in sequence. A new scraping run was started every 15 minutes starting at five past the hour. Hence, we had a large number of individual scraping sessions (96 per day), which were limited in duration though. At most, this would allow for 2,880,000 collected tweets per day, yet we collected just slightly more than 720,000 tweets on average.

For each tweet, we only retain some data (primarily id, text, date, and username) and save it to a database. We only collect tweets in English (as identified by Twitter) and reject retweets (i.e., all tweets starting with *"RT @"*). While most tweets we collected appear genuine, we noticed some spam. We define a tweet as spam if the tweet (or slight variations of it) reappear a large number of times. Such tweets often contain a running number but otherwise repeat the same text as earlier tweets. We set a conservative threshold of labeling tweets as spam if we can find more than 200 matching other tweets (we do not remove very short tweets such as *"Goodnight* 😴*"*). For example, we collected 1,336 tweets like:


[#] @justinbieber HEY JUSTIN! 👏 PLEASE DO NOT
IGNORE THIS! Is is very important that you read. 👇 👉 [URL]

While the share of such tweets of the overall dataset is less than 2%, we do remove them from further analysis. For many emoji, this does not make a difference, but for a small subset of emoji (e.g., the hand emoji used in the tweet above), those tweets do skew the data as there is not a large number of tweets containing those emoji to begin with.

[12]http://www.unicode.org/reports/tr51/#def_level1_emoji.
[13]https://dev.twitter.com/streaming/public.

After initial tweet collection, we had 243 emoji occurring less than 1,000 times. As we intend to reason about the context of emoji, having only few samples per emoji would limit this capability. To gather more data for those emoji, we collected an additional 106,618 supplemental tweets (including 3,083 we later removed as spam) between early August and mid-September 2015. We dropped emoji from this scraping once we had collected more than 1,000 samples for them. While this ran for almost 6 weeks, we were not able to gather 1,000 samples for the 76 least frequently used emoji. However, our least represented emoji, 🈚, now has 211 samples (compared to 27 samples before supplemental scraping). After supplemental scraping and spam removal, our final dataset contains 20.6 million tweets.

We used the opportunity to investigate the gender distribution of users tweeting with emoji. For this, we randomly sampled 1,000 users and extracted the name they use on Twitter (not their Twitter handle, which is a unique user identifier). Note that this assumes people use their real name (or at least first name) on Twitter, as people are free to enter anything they wish in that field. We parsed these data to extract first names and fed those names to the genderize.io[14] service, which, based on a database of more than 200,000 distinct names, tries to assign gender. In our sample, 27% of the names were labeled as male and 34% as female. However, 39% of names (such as, *labrena*, *Utter Amateur*, and *Minnion*) could not be assigned a gender. While this hints at a larger share of females using emoji (as indicated in previous research), more research is necessary to confirm.

We also took a closer look at usage patterns (Section 2.4) in the collected tweets. For this purpose, we randomly sampled 200 tweets and hand annotated them as using emoji either *decoratively*, as *stand-in*, for *emotional* expression, or *stand-alone*. We found some overlap between categories (some tweets, e.g., make use of both decorative and emotional emoji at the same time). However, with 66% of tweets, most emoji were used to add emotional expression. While 34% of tweets made use of emoji as decoration, only 8% of tweets used emoji to replace a word. Surprisingly, we found no tweets containing only emoji and no text in our dataset. However, we discovered this is due to the streaming API filtering out such tweets. For example, we would receive the tweet *"Cute 💠"* but not the tweets "💠" or "💠💎." Unfortunately, this behavior is undocumented, and we can only speculate why we observe this. However, for our further use of the Twitter data, this is not a large problem. As we are specifically interested in context around emoji, single emoji tweets do not add to the model. We do miss out on tweets with multiple emoji (yet no Latin characters) though, which would help with training. Yet, we believe our dataset is sufficiently large to allow us to fill those gaps with emoji sequences from tweets with Latin characters included.

From our initial, unbiased, tweet sample (excluding supplemental tweets), we can estimate how many and which emoji are used. We observed that 83% of tweets only contain one emoji (see Figure 5). Hence, it is necessary to draw on the surrounding word context in establishing emoji similarity across tweets. Considering only tweets in which multiple emoji occur side by side would severely limit the usable data. While there are some outliers with larger numbers of emoji, these are negligible overall. We also see a small set of emoji dominating in actual use (see Figure 6). The top 10 emoji together appear about 9 million times, while the bottom 10 only have 379 occurrences overall. This is a strong indicator that not all emoji are created equal, but might also be due to current keyboards making it too hard to discover less frequently used emoji. Emoji keyboards could take this skewed distribution of emoji into account and place emphasis on supporting common emoji especially well. Yet this should not be read as a reason to omit some emoji from the keyboard. The set of used emoji likely varies by

[14]https://genderize.io/.

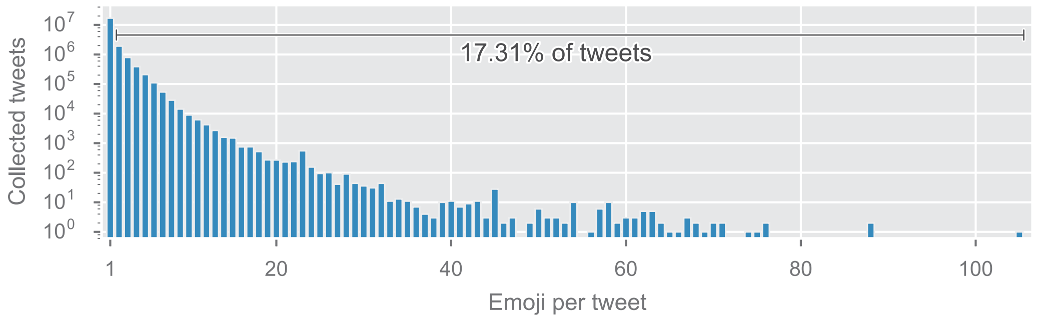


Fig. 5. In our collected tweets (excluding spam), about 83% (∼17 million) only contain one emoji (we do not collect tweets without emoji). At the other extreme, we scraped one tweet containing 105 emoji.

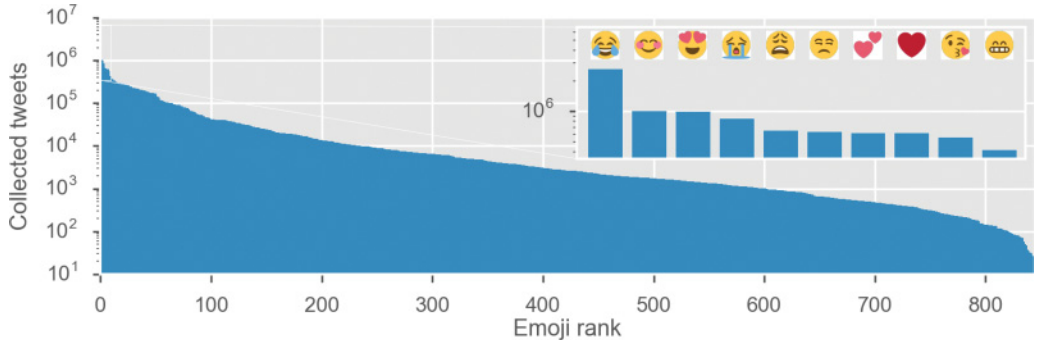


Fig. 6. Emoji frequency in our collected tweets (excluding spam) follows a power-law distribution. While we saw over 2.6 million instances of 😂, we only collected 27 instances for each of 🈚 and 🚟.

person, situation, topic, or chat partner. Exclusion from a keyboard should thus not just be based on the frequency of use. The relative frequency of the “z” character in English texts, e.g., is only 0.096%,[15] yet all English keyboards include it.

Comparative data on emoji frequency and use are, e.g., available from *SwiftKey*—a third party keyboard for iOS and Android. In their *SwiftKey Emoji Report*,[16] they describe findings based on typing data gathered between October 2014 and January 2015. For example, they note that 44.8% of emoji use are only *happy faces*, something we also see in our data. They also find a large amount of differences according to users’ location. Russian users, e.g., are twice as likely to use the ❄️ emoji compared to the average user. On the other hand, Australian users are 66% more likely to use ☀️. The live updating *emojitracker*[17] also monitors emoji usage on Twitter. We compared their emoji ranking to ours and found a strong correlation; Spearman’s rank correlation coefficient $r_s = 0.95$, $p < 0.0001$.

4. EVALUATING THE STATE OF THE ART OF EMOJI ENTRY

As we have shown, the use of emoji has grown rapidly over the last couple of years. However, design of emoji keyboards has so far mostly stuck with long lists of emoji. The sheer number of emoji has made it impossible to show them all at once at a selectable size—a problem shared with other scripts containing many characters. While we will

[15]Computed from the Brown corpus [Kucera and Francis 1967].
[16]https://blog.swiftkey.com/americans-love-skulls-brazilians-love-cats-swiftkey-emoji-meanings-report/.
[17]http://www.emojitracker.com/.

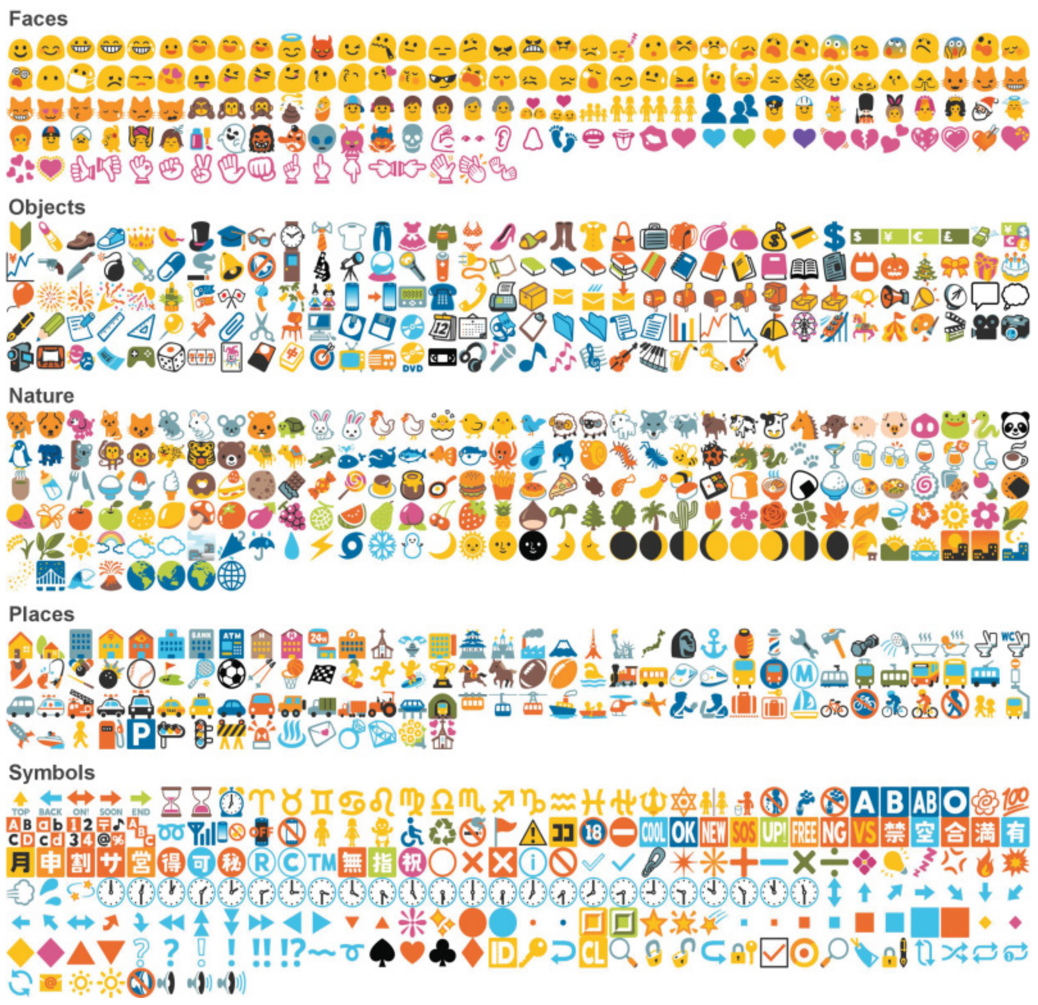


Fig. 7. An overview of all emoji available in the Google keyboard on the Nexus 5 and thus included in our evaluation. The emoji are shown in the same category and order as they appear on the keyboard.

later look at other approaches to the problem of entering characters from such large volume scripts, here we first take a look at the common approach. With lists of emoji being used everywhere, an analysis of such keyboards aids in identifying problems and setting a baseline for any future improvements to emoji entry.

In this section, we will investigate how the default Google keyboard on a Nexus 5, running Android 5.1.1, fares when entering emoji. We chose to investigate emoji entry on Android, as it is the most widely used mobile operating system. While there are many different manufacturers of Android devices, the Nexus 5 is a generic middle ground without OEM customizations. It is also one of the most common devices.[18]

The version of the Google keyboard we tested offers 822 emoji (shown in Figure 7), split into five categories. All emoji are arranged in grids and span multiple pages, which users can swipe through horizontally. Each page is associated with a category

[18]According to *OpenSignal's* August 2015 *Android Fragmentation Report*: http://opensignal.com/reports/2015/08/android-fragmentation/.

and selecting a category jumps to a page belonging to it. However, users can also transition between categories by continuing swiping on the last page of a category. This is different than many other emoji keyboards wherein emoji in each category are shown in a list of their own and no continuous swipe-through is possible. Thus, the Google keyboard actually has an advantage when used for exploration where users want to quickly scroll through all emoji.

The Google keyboard employs two mechanism to facilitate entry of common emoji: (1) it maintains a list of recently used emoji, and (2) also remembers the last used page per category. If users only enter emoji from a small number of pages, this approach often presents them with the target page when opening the keyboard or switching categories. However, this mechanism can be disorienting sometimes when users jump to the middle of a category on category selection but are unclear whether the emoji they are looking for is in a previous or following page. The worst case occurs, when the emoji is on the first page of the category, yet after jumping to the middle users search for it in the other direction. Now users traverse half the pages to reach the wrong end and then need to backtrack all the pages of the category to find the actual emoji. In this situation, always jumping to the start of a category (as in most other keyboards) would have probably fared better. This issue likely resolves though, once users have a better mental model of emoji ordering.

Each page of the Google emoji keyboard shows seven columns by three rows of emoji. This is a common arrangement, e.g., also found in the Samsung keyboard on a Galaxy S4 and the WhatsApp keyboard on a Lumia 920. Many newer devices even show four rows of emoji or more. The number of usual emoji per screen thus is roughly between 18 and 50 (21 in the case of the Google keyboard).

4.1. Challenges when Testing Emoji Keyboards

Evaluating emoji keyboards comes with several challenges. With the set of possible entries commonly as big as 845 emoji (level 1 emoji), or even bigger (e.g., all Unicode 9.0 emoji), exhaustively testing the entry of every emoji in a lab setting is prohibitively costly. However, only testing some, e.g., the most common ones, can bias the results. Drawing a limited random test set of emoji is one way to approach this. But even then, the question of how much coverage of the full set is needed for representative results remains open.

The bigger challenge, however, is picking an appropriate testing procedure. If one were interested in natural user behavior, a chat study wherein two participants exchange messages (such as in Hancock et al. [2007]) would be an appropriate choice. This could also be designed as a longitudinal study that monitors users' chatting behavior (such as in Tossell et al. [2012]). However, this approach risks that only a small number of emoji are actually typed and does not generate a lot of data as much of the time is spend not entering emoji. A longitudinal study also raises privacy concerns, as recording typed emoji can capture user mood. Larger amounts of data can be generated by deploying prototypes to an app store (such as in Böhmer et al. [2014]). However, this still would not give control over the share of emoji in the data. One way around this is to find a game format that allows control of the task, while still engaging users, such as in Henze et al. [2012]. Most commonly, though, text entry methods are tested with a task wherein participants have to copy text verbatim. That is also the approach we chose for our investigation, as this allows for control of which emoji are to be typed.

We adopt an approach wherein we test with an emoji test set sampled from Twitter. To generate a test set, we scrape 10,000 tweets containing emoji present on the Google keyboard (822 different emoji). From the scraped tweets, we only keep the emoji and store how often they occurred. This set includes 502 different emoji (as some emoji never occurred in the tweets scraped for testing). During testing, we sample emoji with replacement from this dataset. The dataset is hence heavily skewed toward the most common emoji. Thus, in order to broaden the range of emoji occurring during the evaluation, we log transform the emoji frequencies to boost the likelihood of rarer emoji appearing. When computing final keyboard performance, we reverse that transformation. Thus, while we favor rarer emoji during selection, we make sure they only influence scores per their original likelihood.

4.2. Participants

For the study, we recruited 12 participants (three female, age 21–41, $\bar{x} = 27.4$, $\mathrm{SD} = 6.3$) from around our institution. The study took ≈30 minutes and after completion, participants received a small non-monetary gratuity. All participants owned a smartphone; however, none of the participants owned a Nexus 5 as used in the study. Only two participants stated they had the same emoji design on their phone. Most participants had phones by vendors with custom UIs (e.g., the Samsung keyboard), or with older Android versions (which used a different design). However, while they were not intimately familiar with the Google one, default keyboards do not differ much currently. We asked participants to indicate whether they often use emoji on their phones on a five-point Likert scale. While one participant strongly disagreed, four agreed, three strongly agreed, and four did not lean either way.

4.3. Procedure

Before participants started the session, they were given time to try out the Google keyboard. In this phase, participants were shown emoji to enter that are not part of the evaluation set. While this prevents participants already searching for emoji which are later to be tested, they nonetheless gain an initial overview of the keyboard as they look for the evaluation set emoji. We chose the training emoji such that they *reset* the Google keyboard's category state. As mentioned earlier, the keyboard remembers the last entered emoji for each category and jumps back to that position when reopened. By always having an emoji from the first page of each category last in the training phase, all category jump targets are reset to the respective first page of the category. While this ensures a consistent starting state for all participants, note that once they entered emoji in the testing phase, the category jump targets differ between them.

After entering 10 emoji, we considered a participant to be sufficiently familiar with the interface and move on to the main study. The interface used here (see Figure 8) is the same as in the training phase but shows emoji from the test set. If participants took more than 1 minute to find an emoji in this phase, we aborted the trial as pilots showed that very long search times frustrated participants.

During emoji entry, we draw emoji to enter by sampling with replacement from the log-transformed emoji set. Overall, 100 emoji are drawn from this set, resulting in 100 trials per participant. We only did 100 trials, as this allowed testing in a reasonable amount of time, after piloting indicated long trial times. In each trial, we start taking the time once the emoji keyboard is activated via the emoji button in the lower right of the Google keyboard. During the trial, we record any page transition, activated either by swiping left or right or by selecting one of the categories and jumping to a page. A trial is complete once the user commits the emoji with the button next to the text field. Upon completion of a trial, the keyboard resets to the QWERTY view.

4.4. Results

As described earlier, we favor rare emoji in the evaluation to have a broader test set. In the analysis, we reverse that log-transform and give weights to emoji equivalent to their observed frequency in the test set. Thus, all time results reported here come in a

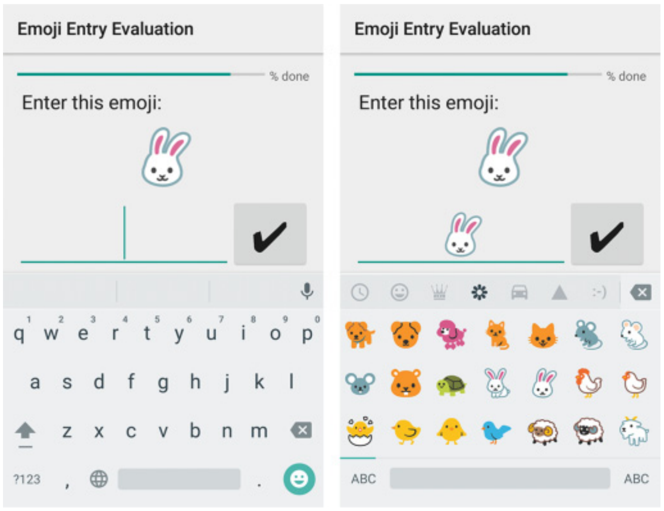


Fig. 8. Layout of our evaluation application while testing the Google keyboard. Participants are shown the emoji to enter at the top of the app. In each trial, they need to activate emoji mode, find the respective emoji, and click the commit button. Committing a selection also switches the keyboard back to QWERTY mode.

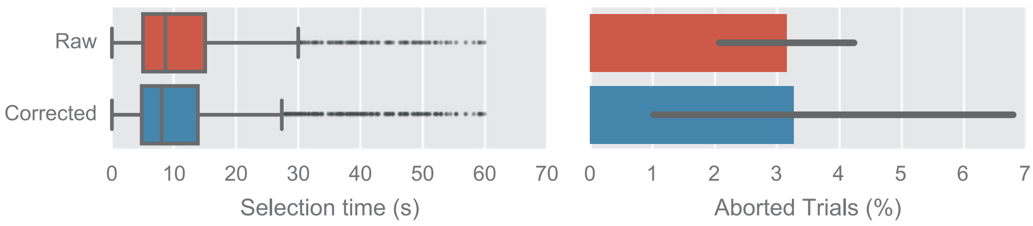


Fig. 9. This figure shows (left) how fast users were able to enter emoji, and (right) how many trials were aborted because they took longer than 1 minute (error bars show 95% confidence intervals). As we log-transform emoji frequency during evaluation, the raw results do not accurately reflect expected performance. We remove this bias toward rare emoji for the corrected values, which only slightly changes the results.

raw and a corrected version, where the corrected version accurately captures expected performance for emoji use in the wild.

Figure 9 shows the outcome of the evaluation. The raw selection times over all successful trials was 8.8 s (median) and 12.5 s (mean), respectively. Once we correct for the skewed frequency, this changes to 8.2 s (median) and 11.4 s (mean). This equals about five to seven characters (while emoji can replace a word, from an entry perspective each emoji corresponds to one selection) per minute. Note that these times only take into account successful trials, the average would be higher, had we not stopped trials after 1 minute. In our fastest trial, the participant selected 💋 in 864 ms. This was possible because no page change was necessary for the selection (the emoji was already on the initial page). In fact, 64% of the 100 fastest trials did not require a page change at all. Correspondingly, there is a strong linear relationship between the number of page transitions and the resulting selection time of a trial, $p < 0.0001$. About 3.2% of trials took longer than that 1 minute and were aborted (one participant entered all emoji within a minute, while all other participants exhibited failure rates between 1% and 6%). If we take into account the emoji frequency, the expected average failure rate corrects only slightly to 3.3%. This shows that for most trials, selection time is within a reasonable range.

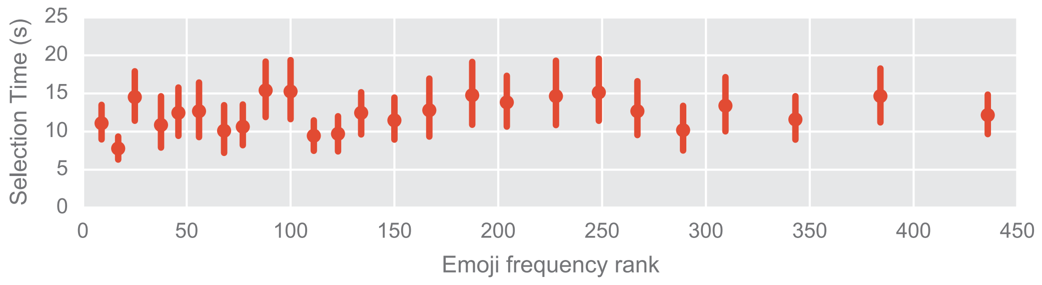


Fig. 10. Selection time for entered emoji, ordered by rank. Error bars show 95% confidence intervals for each of the 25 aggregated bins. There is no clear pattern of often-used emoji being selected faster than rarer emoji. Note that this plot only includes data for successfully selected emoji, excluding failed trials.

In some trials, selection time is much larger than others. We compare selection times for emoji with at least five trials (65 different emoji). This shows a large difference between the fastest and slowest emoji with 😍 entered in an average 4.3 s (five trials), while ☀️ (also five trials) took an average of 35.5 s. In fact, the 10 fastest emoji (😍, 😎, 😬, 💞, 🙋, 💋, 👅, 💜, 😤, and 🍆) were entered more than five times faster than the 10 slowest emoji (📷, 😋, 🎥, 🍌, 🌚, 💌, 🚶, 🔥, ⚽, and ☀️). The faster emoji were more commonly used (per our sampled frequencies on Twitter) than the slower ones. But while they are used more than twice as often, a linear regression on mean selection time and log-transformed emoji frequency shows no significant relationship, $p = 0.3$. Hence, we cannot identify good criteria to define a subset of emoji that are faster to enter than others.

The failure rate also varies by emoji. For example, no trial of 👣, 💅, 💢, 🐺, ❇️, or 📈 was successfully completed. As we randomly sample from a large set of emoji, those six emoji only account for a total of eight trials though. These trials are thus not representative for general performance. For a closer look, we only consider emoji for which we have collected data from at least five trials: a smaller set of 70 emoji. From those 70 emoji, 61 were always entered successfully (385 trials). The remaining nine emoji (🌚, 😂, 🍃, 🔥, 📖, 💘, 💯, 👏, and 💦) each only led to one failure (in a total of 58 trials). There is no clear pattern in this set of emoji to suggest that some specific subset of emoji are more likely to fail than others.

One might assume that frequently used emoji would be easier to enter. However, when plotting selection time as a function of emoji rank (see Figure 10), no effect of rank is visible. To confirm, we ran an independent-samples t-test, comparing selection time between the 50 lowest ranked (i.e., the emoji is not frequently used) samples and the 50 highest ranked samples. There is no significant difference in the selection times for lower ranked ($M = 12.1$ s, SD = 11.3 s) and higher ranked ($M = 10.2$ s, SD = 8.1 s) emoji; $t(98) = 0.95$, $p = 0.35$.

Finally, we checked how users progressed toward the target emoji. Figure 11 shows an overview of all trajectories toward the target emoji's page. As their first action after opening the emoji keyboard, most users immediately jump to a different category. This was the case in 640 of the trials (∼57%) and is visible as the large share of category jumps (shown in red) at the left of Figure 11. But that initial jump was not always the right one and in 14% of successful trials participants jumped to a category more than once (up to six times). Overall, though, there was fast progression toward the target page. However, there is a long tail, as some trials took much longer. This is also visible in Figure 12 that highlights how the majority of trials already reach the target emoji after about 10 page transitions.

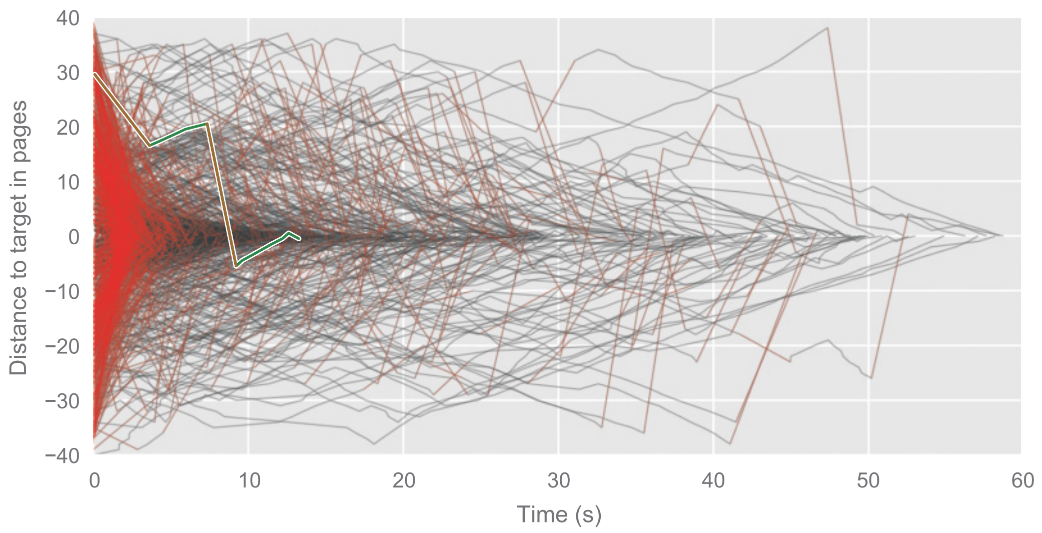


Fig. 11. Here we show how participants progressed toward the target emoji. In each trial, users start at the last active page of the keyboard. From there, they need to navigate to the target page (here shown in the middle). Ideally, users would jump to the proper category (category jumps are shown in red) and then swipe through some pages (page transitions shown in black) until finding the target emoji. Here, we see that there is indeed a fast approach of the target, yet users in many trials get lost. The plot also shows page transitions from one trial highlighted in green. That participant immediately selected a wrong category, searched that category for a while, jumped to the right category, missed the target page while searching within that category, but finally selected the target emoji after about 13 s.

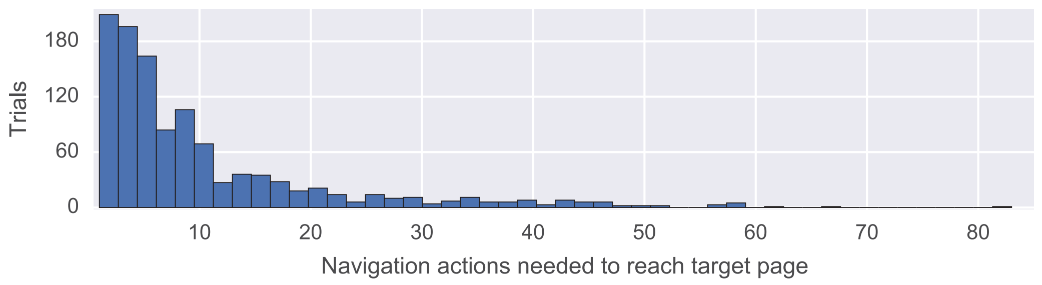


Fig. 12. When we look at how many jumps to different categories and swipes between pages (here aggregated as navigation actions) it takes users to reach the target emoji, it can be seen that most successful trials end after about 10 actions. However, there is a long tail where users navigate through many more pages before finally arriving at the desired one.

We can also investigate whether participants missed the target emoji (navigated to the proper page, but continued their search). This was the case in 230 trials (∼19%). Users would often quickly swipe through the pages and then backtrack to the target emoji. One example of this is also shown as a highlighted trajectory in Figure 11. As can be seen, the participant in that trial started far to the right of the target emoji's page, selected a category (big jump) and checked several pages of that category. After not finding the emoji there, she jumps to another category and continues searching there. This user actually overshots the target page by one page, but immediately goes back and selects the emoji.

We find that in 802 (66.8%) of the trials, participants made use of the category buttons to jump to a different page. In the remainder of the trials, they either swiped through many pages or were already close to the target emoji. However, we also find that participants often chose the wrong category and had to pick another one. In 191 trials (15.9%), participants picked a category at least twice. The number of category jumps quickly declines though (6.6% trials contain more than two jumps, 3.3% more than three, and only 1.2% more than four). The most extreme trial we recorded had the participant select a different category eight times—rechecking already visited ones as well—and visiting 83 pages in the process.

While the above analysis only considered successful trials, we can also take a look at the 38 failed trials to see how those searches progressed. Most surprisingly, we find that in 66% of failed trials, participants actually visited the page containing the target emoji. The participants hence often missed it and continued their search elsewhere. Participants also visited many more pages in failed trials than in successful ones. On average, 53 pages were visited (50 median) in failed trials, while participants in successful trials visited only 10 pages (6 median). However, the worst recorded failed search spanned a total of 98 page visits, even though the keyboard only has 42 pages.

4.5. Discussion

Overall, emoji entry performance with the Google keyboard is adequate. Selection time can be slow, but only becomes very large in a limited number of cases. But in some situations, the Google keyboard exhibits problems. Especially, the fact that users miss the emoji 19% of the time is concerning. While the Google keyboard shows only 21 emoji per page, the emoji keyboard on the iPhone 6+ shows up to 50. A higher density of emoji makes it easier to miss one, or at least increases the time needed to scan a page of emoji. With the number of emoji growing, this search problem is bound to intensify.

We can also see that users often pick the wrong category for an emoji. In almost 16% of the trials, they jumped to a different category more than twice. This indicates that there might be a room for improvement in the category assignment, as users' model of emoji location does not always match the actual location. Presently, there are little data that can be used to inform category assignment. Later on, in this article, we will look at using large amounts of tweets to inform which emoji should be close to each other. In actual use, most users would probably not spend the time to find one specific emoji, though. Instead, they would settle on a different one or eschew emoji use altogether. However, as users should be able to enter any text, they should also be able to properly enter any emoji.

The fact that users sometimes failed at finding emoji in a reasonable amount of time also made us wonder about the base cost of using the Google keyboard interface. In our tested version of the Google keyboard, the 822 emoji are split over 42 pages. As an animation is shown for each page transition, just visiting a page already takes some time. The overall large number of pages then makes exploration and search of emoji cumbersome. To quantify this, we ran a quick informal study with seven participants (one female, age 22–34 years). We had participants start on the first page of the Google keyboard and asked them just to swipe through all the pages. This already took them about 21 s. If additional visual search effort is necessary, it is clear that exploring the available emoji can take a long time.

The current Google emoji keyboard does not favor more common emoji over less common ones. Hence, we could not observe faster selection for more frequently used emoji. Depending on the specific design goals, this can be perfectly acceptable behavior. If we consider all emoji as equally important, then we would indeed not want to favor any of them. In fact, favoring the already more popular ones would only reinforce this difference. However, novel emoji designs could decide to bias selection efficiency slightly toward more common emoji. The critical aspect in such an endeavor would be how to find the right balance of bias to support.

Table II. Differences Between Traditional and Emoji Text Entry

Traditional text entry	Emoji entry
• Characters are entered from a small set of well-known symbols (the ISO basic Latin alphabet, e.g., has 26 characters).	• Characters are entered from a large set where users might be unfamiliar with many of the symbols.
• The meaning of each character is well defined and there is no character-level ambiguity.	• Multiple interpretations per emoji are possible and ambiguity can allow for choice between several emoji (e.g., several emoji can connotate enjoyment).
• Input needs to be optimized for speed. Focuses on composition of words.	• Speed and efficiency are less important. Instead, exploration of the available symbols needs to be easy.
• Designed for frequent and, sometimes, prolonged use.	• Designed for intermittent and sporadic use.
• A means for general purpose expression. Text is neither inherently playful, nor somber.	• Emoji are visual (and often playful and *"cartoony"*) in nature, making them immediately noticeable when embedded in text. In text, emoji can provide a sort of emotional annotation (e.g., pointing out intended sarcasm).

Compared to traditional text entry, the five to seven emoji per minute rate, currently achievable with the Google keyboard, seems slow. We can also compare this number to performance data from other large character systems. With pinyin (Chinese) text entry, for example, characters can be entered much faster with performance varying between 15 and 35 (~25 on average) characters per minute, depending on the input method used [Liu and Wang 2007]. This is even though there are more Chinese characters than emoji. With pinyin, users still enter Latin characters, yet those numbers show that other approaches to large character set text entry fare much better than the list selection one currently used for emoji.

5. INPUT METHODS FOR EMOJI AND OTHER LARGE CHARACTER SETS

Having taken a closer look at the predominant method for emoji entry—selection from lists—here, we take a step back and look at the problem from a broader perspective. Even just for emoji, while list selection is the predominant form of entry, other input methods do exist. But entering emoji is not the only text entry area where the set of characters to enter is large. Especially East Asian languages, such as Chinese, deal with similar problems of mapping characters to a format that allows for easy entry. With these scripts, there is not a clear best approach and thus there are, e.g., several different kinds of input methods for Chinese characters. Note that any character set can be considered *large* in some contexts. For example, in early phones methods such as *T9* are used to input Latin characters via the smaller numeric keypad. Similarly, current research on text entry methods for smartwatches shows again how just Latin script can already be a lot of characters to support.

Compared to text entry for large character sets, text entry methods for sets of up to ~30 characters are well researched. For those small sets, frequently each character is assigned to a button, and the buttons are arranged in a "simple" layout (e.g., QWERTY). However, this approach does not work for entering emoji—there is no layout that shows all emoji at the same time at a selectable size. In fact, in addition to distinct layout problems, there are several general differences between entering text and entering emoji, as shown in Table II.

In this section, we present an overview of classes of text entry methods currently available for entering emoji. Furthermore, we take a look at existing input methods for other kinds of large character sets and how they could influence designs of emoji input methods. Particularly, East Asian scripts have inspired a wide range of input methods, e.g., based on the shape or phonetic of the characters to enter.

5.1. Enumeration

Listing all possible characters and then allowing users to browse that list is the approach currently used by emoji keyboards. This approach has several advantages: (1) it does not require any ordering of the characters, (2) it allows users to browse all available characters and thus aids in discovery, (3) as users can see the available emoji, they do not need to remember exactly what it looks like, and (4) it is easy to implement and extend for new characters. However, as stated earlier, this approach also does not scale well to larger character sets. The absence of an ordering also means that it can be hard to remember the location of a character.

A slight tweak of just showing all characters in a large list is the *EmojiZoom* input method [Pohl et al. 2016]. Here, all emoji are shown at once, but only are selectable after users zoom in. High-resolution screens on current mobile devices enable users to still make out sufficient details, even at the zoomed out level. As the start view is always the same, this design also allows users to build spatial memory and, e.g., learn that smiley emoji are always found in the top left corner. The spatial mapping also allows for good exploration of the available emoji. Users can zoom in slightly, to a level where individual emoji are clearly legible, and then pan around to explore the space. *EmojiZoom* organizes emoji according to the Unicode order in snaking layout. This still leaves cases wherein emoji location is ambiguous. The ordering is also only one-dimensional and the mapping to two-dimensional space thus also limited. For a two-dimensional layout of emoji more detailed relationship information is necessary. The semantic similarity model we later present in this article would enable a more spatially optimized version of *EmojiZoom*.

5.1.1. Categorization. To alleviate the problems of large lists, emoji keyboards split the emoji into different categories. In Android 5.0, e.g., those are *faces*, *objects*, *nature*, *places*, and *symbols*. However, while this limits the number of pages per category, splitting introduces new problems. The assignment to categories is often arbitrary and a compromise. For example, 🐱 is part of the *faces* category, while 🐱 is part of the *nature* category with most other animals. Yet, while 🐴 is also in the *nature* category, 🐎 and 🏇 are found in the *places* category with other sports and activities. Such ambiguities are common. Should 🇺🇸 be with the other flags or near 🗽? Why is 🔉 not next to 🎶 and 🎧 on the Google keyboard? Because emoji often allow multiple uses and interpretations, imposing a strict ordering is bound to produce such cases, where one would expect one emoji to be close to another, but it is found elsewhere. One could put emoji in multiple categories (not currently done), but this would also inflate categories and might negatively impact search time.

5.2. Querying and Prediction

Instead of selecting characters from large lists, users can be enabled to query the set of characters to find the one they are looking for. While this allows for simple interfaces (e.g., only a text box), the main problem with this approach is that users have to know the character they are looking for. Thus, query systems fare poorly when users need to discover characters or cannot exactly remember the shape of a character. However, such systems do integrate well with the existing Latin script keyboards as they require no mode switch to a dedicated emoji keyboard.

5.2.1. Handwriting/Drawing. Handwriting recognition [Tappert et al. 1990] is an established method for entering characters. While it theoretically allows for arbitrarily large character sets, it is effectively limited by how well users can remember and draw each character. Earlier research already extended this to pictogram retrieval [Lopresti and Tomkins 1993]. Google has also been experimenting with using handwriting input for entering emoji.[19] While their emoji mode works well for emoji with an easy to draw silhouette (e.g., 🟧 or 💗) or only few features (e.g., 😊 or ✔️), it fails with emoji where no clear handwriting equivalent exists or where the emoji is too complex (e.g., 👌, 🌇, or 🌍). Outline drawings are also not able to differentiate between emoji that only differ on how they are filled. For example, the flags of Germany 🇩🇪, Estonia 🇪🇪, and Hungary 🇭🇺 only differ in the color of their stripes and are not distinguishable in outline sketches.

5.2.2. Textual Search and Replacement. When users know the name or description of a character (e.g., 🚍 is called *oncoming bus*), they can search for it with text queries. While we are not aware of any emoji keyboard implementing this, some do offer a related method. For example, on *Windows Phone 8.1* users can select words and then pick an emoji for replacement (e.g., replacing *"dragon"* with 👹). This approach uses a curated subset of emoji annotations as given in the *Unicode CLDR*.[20] In fact, this approach of linking emoji with their respective keywords is becoming more common. Apple, e.g., is planning to introduce the capability to replace words with emoji on tap to *iMessage* with *iOS 10*. Some systems such as the *Slack*[21] messaging platform, the *Discourse*[22] forum software, the *Github*[23] repository service, and many others,[24] allow emoji entry via text codes. Here, entering `:thumbsup:` in a text field would, e.g., result in that text in the output to be mapped to the 👍 emoji.

5.2.3. Prediction. Instead of making emoji entry a dedicated task, some keyboards try to roll it into the autocorrection mechanism. An early example of this can be found on *Windows Phone 8.1* where, after entering *"birthday,"* the keyboard will suggest entering 🎂 next. Such emoji suggestions can also be found in some third party keyboards. The *Minuum* keyboard is one example, featuring *smart emoji prediction.*[25] Instead of integrating into a keyboard, *Dango* floats on the screen, analyzing what is being typed to suggest emoji, GIFs, and stickers to add to the current message.[26] Like our semantic emoji model, *Dango* uses a neural network to embed emoji and text in the same space. Instead of emoji, Urabe et al. [2013] analyzed affect in input text and then infer appropriate emoticons. User thus implicitly control which emoticons are available with the text they enter. Our work in this article can support existing methods in this category. As we will show later, we are able to predict semantic similarity between emoji, which could directly support prediction.

One aspect of prediction-based methods that so far has seen little focus, though, is how they could support exploration. Currently, the existing systems only present users with a shortlist of matching emoji. However, going beyond this list is not supported yet. If similarity data for emoji are available, keyboards could be extended to allow users to grow a result set with other emoji, similar to the ones already shown. Similarly, systems could present an initial coarse but wider prediction, and then only after users pick the rough direction they want to go in, present a more fine-grained list of predicted emoji.

5.2.4. Query by Picture. While not used as an actual text input method, *Image2Emoji* demonstrated how to select a set of emoji based on an input image [Cappallo et al.

[19]https://play.google.com/store/apps/details?id=com.google.android.apps.handwriting.ime.
[20]*Unicode Common Locale Data Repository*: http://cldr.unicode.org/.
[21]https://get.slack.help/hc/en-us/articles/202931348-Emoji-and-emoticons.
[22]http://blog.discourse.org/2015/12/emoji-and-discourse/.
[23]https://github.com/blog/1289-emoji-autocomplete.
[24]See, e.g., http://www.emoji-cheat-sheet.com/.
[25]http://minuum.com/exploring-emoji-the-quest-for-the-perfect-emoticon/.
[26]http://getdango.com/emoji-and-deep-learning.html.

2015a].[27] Input images are mapped to textual descriptions via a convolutional neural network, combined with any accompanying text (title, description, tags). Emoji are also mapped to text by representing each with its name. Which emoji are then similar to an image is determined by similarity of their textual descriptions in a semantic embedding. Such a method could be useful for generating a shortlist of emoji for use in captions when sharing images on social media or when commenting on media. However, for a general text entry method, requiring users to have an image of what they are trying to express handy is likely too cumbersome.

5.3. Methods for East Asian Languages: Mapping to Latin Script

So far, we have taken a look at methods already in use (even though only experimentally) for emoji entry. For East Asian scripts, there are several more methods in use that work by defining a mapping from each character to a sequence of Latin characters. Such a mapping is generally (1) phonetic, or (2) based on the shape of the character. Here, we detail those two approaches and describe how those could relate to emoji entry.

5.3.1. Phonetic Mapping. Some of the most common methods here are based on *Pinyin*, the phonetic system for Mandarin. Instead of entering a character directly, users enter a phonetic representation (which is possible using an extended set of Latin characters). For example, to enter "`DengXian`," users can just input *"nihao."* However, while such methods work well for, e.g., East Asian scripts, they are not directly applicable to entry of emoji or symbols where no clear phonetic representation exists. Instead, one could spell out the name of an emoji (e.g., *"tropical drink"* ⇔ 🍹). But many of these names are not obvious and there would need to be a way to indicate whether emoji or text entry is desired. A basic version of this is found on Windows Phone 8.1, where saying *"smiley"* inserts a `:)`, while *"frowny"* instead puts a `:(` into the text. However, speech is generally poorly suited to the exploration required for entry from the larger emoji set. One possible approach here, though, would be to dictate text first and then only use voice input to insert emoji afterward. Textual context could then limit the number of candidate emoji to a manageable set for speech input.

5.3.2. Graphological. In graphological mappings, characters are decomposed into parts that can then be mapped to Latin characters. In the Cangjie input method, for example, all Chinese characters are represented by 24 basic character components. For example, the "水" (water) radical is used in characters, such as "泰", "永", or "冰." Each of those components is mapped to one Latin character (the water radical, e.g., is represented by the letter *"E"*). To enter a Chinese character, users then need to input several radicals in a specific sequence. Users thus need to be familiar with the decomposition rules that define how to map characters to sequences of radicals.

This concept could potentially be used for emoji as well, if suitable decompositions are defined. For example, 😗, 😣, and 😴 all share a common visual component (the basic smiley face) with varying facial features. We can imagine entering emoji by chaining together descriptors (e.g., specifying the combination of *face* and *happy* already reduces the set of possible emoji to a manageable number, which could then be displayed for selection). Instead of showing a list of all emoji, emoji keyboards could then just show a much smaller list of emoji radicals. However, this requires manual or automated tagging of emoji radicals, where it is not clear what the set of radicals would be.

[27]They have also shown the reverse: searching for video based on a set of input emoji [Cappallo et al. 2015b].

6. MOTIVATING EMOJI SIMILARITY MODELING

Keyboards have always brought with them questions of optimization: After all, the whole point of the invention of the typewriter was to speed up writing. But optimization is particularly important, as there is a large potential within the nature of the device. As the letter-to-key assignment is arbitrary, in the sense that any such assignment is possible to build or implement, there is a large amount of leeway in the design. Common optimization goals for keyboards are, e.g., to make text entry faster, or to make disambiguation between neighboring keys easier.

Early keyboard layouts, such as QWERTY, were hand designed. But the number of possible layouts is very large and design intuition was eventually replaced by computational approaches. An early example is Zhai et al.'s [2000] Metropolis keyboard that tries to optimize for movement time between keys, weighted according to the respective bigram probabilities. In fact, this formulation, based on Fitt's Law, is a common optimization target, as it optimizes for the travel distance of the fingers and yields results were keys often used after each other are close to each other. Of course, speed is only one aspect to optimize for and thus there are other keyboards that optimize for multilingual input [Bi et al. 2012], touch input [Weir et al. 2014], or try to find the best layout for multiple optimization goals at once [Dunlop and Levine 2012]. In addition to different objective functions, researchers are also applying more complex optimization methods, such as integer programming [Karrenbauer and Oulasvirta 2014], to search for the best layout.

However, the optimization goals for text entry and emoji entry are completely different. For text entry, a common goal to optimize for is speed of entry, or the related travel distance between keys. Hence, objective functions for this purpose generally include bigram probabilities as a weighting term. The underlying assumption here is that many characters are entered in series. If users only ever entered one key such an optimization would not help at all. But while longer sequences are common in text entry, they are not a common case for emoji entry. Instead of entering many emoji after each other, users usually only want to add one or a small number of emoji to a message.

The most important aspect of emoji entry is thus the search for individual emoji and not the entry of emoji sequences. As such, we do not need to optimize travel distance between subsequently used emoji but instead need to optimize for search time. This aspect of emoji entry connects more with work in visual search than work in text entry. Where text input methods have been less concerned with searching for the right key (users are expected to internalize the layout and only be restricted in their entry speed by travel distance at some point), research in menu or icon selections focuses specifically on this.

The benefit of pictoral presentation was investigated, e.g., by Niemel and Saarinen [2000], who showed that searching for icons is easier than searching for text labels. They also saw a decrease in search time when related icons were grouped together. Grouping-related emoji together could thus potentially also lead to reduced search times. There are also ongoing efforts to model visual search, similar to Fitt's Law capturing movement, in order to make performance predictions for interfaces. Such modeling generally falls into two categories: (a) trying to fit a mathematical model to experimental data, or (b) trying to model cognitive processes. An example of the former is Bailly et al.'s [2014] work, who fit a mathematical model to predict search time in menus. Instead, Kieras and Hornof [2014] try to model the cognitive process of vision, in order to predict where people look and, in effect, how long it will take them to find a target. Their model, e.g., captures that color is the strongest influence on how well targets can be distinguished. We could thus assume that pages of emoji very similar in color, (e.g., the smiley emoji: 😗, 🙂, 😃, 😬, . . .) make it harder to find a specific emoji than pages with stronger color differences (e.g., the hearts: ❤️, 💙, 💚, . . .).

We can relate this back to the input methods presented in Section 5. For categorization style input methods, we need to optimize the arrangement of emoji so that related emoji are in the same category. Within a category, we would also expect more related emoji to be close to each other. For example, it is sensible to expect 🚑, 🚒, and 🚓 to be not just in the same category, but also next to each other. A model for categorical emoji input methods would thus need to help with both category assignment and within-category ordering. If we only look at the more relaxed enumeration input methods, only the ordering aspect is necessary. This can be a one-dimensional ordering for linear lists but could also mean a two-dimensional ordering for methods like *EmojiZoom*.

Predictive methods also require an emoji similarity model. For example, if a user entered *"Hitting the road,"* a model should be able to predict emoji such as 🛣️, or 🚙. In this example, this would require capturing relationships between emoji and English language words. But even just modeling relationships between different emoji, excluding other text, is helpful. This could, e.g., be used to predict which emoji are likely to be added to a multi-emoji response, with one or more emoji already entered. After typing 🎁, it is likely users would also want to add 🎉 or 🎂 to the message. This can also facilitate exploration where users can limit the search space to a smaller set of candidates by providing a reference emoji.

In this article, we concentrate on a between-emoji similarity model that can be used to guide emoji arrangement. The underlying motivation is that users would have an easier time finding an emoji if it is close to similar ones. Instead of looking at each individual emoji, they then only need to look at one to decide whether they are in the vicinity of the one they are looking for. For example, when searching for the 🏀 emoji, users can assume they are close once they see any one of ⚽, 🏐, or 🏈. It would also be fair to assume other sports-related emoji are nearby, with ⛹️, e.g., as close as possible to 🏀. On the other hand, if users see 🍣, 🛰️, or 🤖, they should be able to safely assume that 🏀 is somewhere else. Building a similarity model of emoji is a necessary step to optimize for these criteria.

7. TOWARD A MODEL OF EMOJI SIMILARITY

As the number of emoji is currently growing with every new version of the Unicode standard and more emoji make it into vendors' emoji keyboards, the current presentation of emoji in one or several lists is getting more and more problematic. The Google keyboard on the Nexus 5, e.g., already contains 42 pages of emoji for its 822 emoji (the latest version of iOS comes with about 1,300 emoji). In our study in Section 4, we found that it is easy to miss an emoji. Furthermore, exploration of available emoji is expensive and a good ordering of them thus is necessary to achieve a reasonable average search time. While categories subdivide the space, which speeds up search somewhat, the assignment of emoji to categories, as shown earlier, can itself be problematic.

As we have described, many keyboards already make use of models to inform their design. For example, when designing a keyboard for the English language, key arrangements can be chosen so that expected travel time between keys is minimized according to bigram probabilities for letter pairs. However, there is no clear equivalent of model-based optimization for emoji entry. Yet, as outlined in Section 6, a similarity model for emoji would help in guiding design of emoji input methods. In order to make model-driven improvements of emoji keyboards possible, we thus set out to build such a model.

In this section, we will look at two ways to build emoji similarity models: (a) based on emoji annotations, as defined by the Unicode standard, and (b) based on semantic information derived from the large number of tweets we collected. Both similarity

Table III. Jaccard Similarity Coefficient for a Set of Example Emoji Pairs

Emoji_A	Emoji_B	Similarity	Emoji_A	Emoji_B	Similarity
🎥	📷	1.0	👊	✊	1.0
🎻	🎸	0.75	🚺	🚹	0.75
🚜	🚕	0.5	🍚	🍕	0.5
📙	📐	0.25	🎨	🎦	0.25
😃	🐩	0.0	👋	🐬	0.0

measures allow us to quantify which emoji are related and should be close together. We compare performance of both models in a crowdsourcing study and explore differences in what aspects of emoji they capture.

7.1. Deriving Emoji Similarity from Unicode Annotations

As mentioned earlier, all emoji are already tagged with some annotations.[28] For example, 🐈 is tagged as *animal*, *pet*, *nature*, and *cat*. Those annotations are intended to help users winnow down the set of emoji by entering the tag and then selecting the matching emoji.

We initially hypothesized these annotations could be used to establish similarity between any two emoji: by comparing their annotations. Similarly, Aoki and Uchida [2011] checked for co-occurrence of emotional words in blog posts to derive feature vectors (and thus a way to compare) for emoji. Let T_{Emoji_A} be the set of all annotations for Emoji_A. Emoji are considered similar if they share many terms and disagree on few. We thus chose the Jaccard similarity coefficient J as quantifier, which is defined as follows:

$$J\left(T_{\text{Emoji}_A}, T_{\text{Emoji}_B}\right) = \frac{\left|T_{\text{Emoji}_A} \cap T_{\text{Emoji}_B}\right|}{\left|T_{\text{Emoji}_A} \cup T_{\text{Emoji}_B}\right|}. \tag{1}$$

Some examples of the resulting similarity values can be seen in Table III. This similarity measure works well for clustering around concepts, such as *food*, *animal*, or *vehicle*. However, it does, e.g., not include a notion of two emoji coinciding in an activity. For example, the shown dissimilarity of 😃 and 🐩 only tells us that smiling and poodles are not the same *thing*. From a different perspective, though, one could certainly be happy about a new dog or a walk with a dog. This aspect is not captured by this categorical similarity, which does not take context into account. Hence, we set out to find an approach for establishing similarity that captures a wider notion of what emoji relatedness means.

7.2. Emoji Model Building from Tweets

To move beyond categorical similarity, we set out to build a similarity measure based on how emoji are actually used. By building on actual use, similarity is hence defined only by whether users enter emoji together or whether they enter emoji in similar contexts. We hypothesize that this would be able to capture connections between emoji that are not reflected in just their tags. For example, we would expect 😃 and 🐩 from

[28]Available at: http://unicode.org/repos/cldr/trunk/common/annotations/. Note that these annotations have recently been updated. The annotations we use in this article are per Summer 2015. Some tags were removed and 🐈, e.g., is now only tagged as *pet* and *cat*.

our previous example to exhibit some connection once we take into account the context around emoji. We derive this contextual information from the large number of tweets we collected earlier. Because most tweets only contain one emoji, we need to include all text to build a strong model of emoji similarity. For example, if we were to see both 👭 and 👯 individually in tweets containing the word *"besties,"* we could conclude that they are probably related. We chose to use a word-embedding approach to map each emoji (and every other token) to a word vector. Similarity between two emoji can then be determined through the similarity of their respective word vectors.

In a word embedding, individual tokens are each mapped to n-dimensional numeric vectors. For example, 😃 might map to $[0, 0]^T$, 😺 to $[0, 1]^T$, 😢 to $[1, 0]^T$, and 😿 to $[1, 1]^T$. In this example, one dimension encodes sentiment, and one dimension encodes whether the emoji is a face or a cat. Other emoji could then also be represented in this space, e.g., 😐 might be represented as $[0.5, 0]^T$. We can see how this basic embedding directly leads us to a similarity measure. If two emoji have similar numeric representations in this space, they are conceptually close. For example, an emoji at $[0.2, 0]^T$ and an emoji at $[0.3, 0]^T$ both would be rather cheery faces, quite far away from an emoji at $[0.9, 0.8]^T$ that would be some sad animal. In the word embedding, we can compute the distance between two emoji 😶_a and 😶_b with any norm. We chose to use the Euclidean distance function, given as

$$\sqrt{\sum_{i=1}^{n} (\text{😶}_{a_i} - \text{😶}_{b_i})^2}. \tag{2}$$

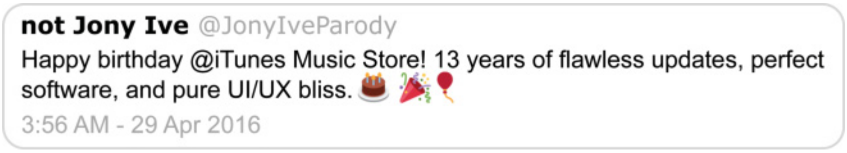
Instead, one could use other norms, such as the Manhattan distance, or similarity measures like cosine similarity. Note that we are not interested in the absolute distance between two emoji but aim to rank emoji to find the x most related ones.

However, in our example, embedding it would be hard to represent an emoji like 📺 as it does not align to the chosen dimensions. A television set is neither face nor animal while also not having a clear place in a sentiment dimension. To represent the diverse set of emoji, we need many more dimensions. If we were to hand-pick the dimensions we might, e.g., use concepts such as *sentiment*, *seriousness*, *edibility*, or *gendered*. But picking the right dimensions and then manually rating emoji would be a very complex and error-prone task. Instead, we use a neural network to learn appropriate dimensions for our word embedding. While those dimensions then do not individually map anymore to easily understood concepts, automating this process enables us to make use of much more data than humans could incorporate in a manual design.

To create the word embedding, we use a Python implementation[29] of *word2vec*[30] [Řehůřek and Sojka 2010; Mikolov et al. 2013]. Word2vec implements two versions of a neural network language model: continuous skip-grams and continuous bag-of-words. For both algorithms, there is only one hidden layer in addition to the input and output layers. We use skip-grams, where the vocabulary forms the input layer of the neural network and context words form the output layer. Thus, after learning, the hidden layer is tuned to predict the context (surrounding words) for an input word.

During training of our model, input words and their context are taken from individual tweets. For example, consider the following tweet:

[29]http://radimrehurek.com/gensim/.
[30]https://code.google.com/p/word2vec/.


not Jony Ive @JonyIveParody
Happy birthday @iTunes Music Store! 13 years of flawless updates, perfect software, and pure UI/UX bliss. 🎂 🎉🎈
3:56 AM - 29 Apr 2016

For the input token 🎂, the context is given by surrounding other emoji and words. Thus, when this tweet is used for training, it would strengthen connections in the hidden layer that result in predictions of tokens like *bliss*, *pure*, or 🎉. In this example, we would also train for a relationship between *UI* and 🎂. However, this pair is very unlikely to occur elsewhere in our dataset, while the tokens *birthday* and 🎂 likely occur together many more times.

After training, the neural network is tuned to predict the likely context for every input token. Hence, the output layer is a softmax regression classifier, outputting the probability for each output token. If we feed ☔ to the neural network, it might, e.g., provide an output vector that assigns a high probability to the *rain* token. Keep in mind that the input is a one-hot vector: All values are set to zero, except the position representing the input token. While the hidden layer is a large matrix, a multiplication with the input vector essentially just selects one row from this matrix. If two input vectors result in the same output in the hidden layer, then their predicted context is also equal. From this follows that if two tokens share a similar context, their hidden layer weights must also be similar. We can thus use those weights directly to form the word vector for our word embedding.

As discussed above, we need a higher number of dimensions for our embedding to capture concepts of emoji. We chose a balanced 300-dimensional space for this: between the 100-dimensional word2vec default and the 500-dimensions used in the similar setup of Image2Emoji [Cappallo et al. 2015a]. We experimented with different hidden layer sizes in the process, yet did not find noticeable differences. However, we only tested layer sizes between 200 and 400 dimensions. Very small layer sizes are likely not able to capture relationships between emoji that well. While we do not limit the vocabulary of the model (a word vector is trained for each word in the corpus), we ignore all words that occur less than 50 times. Given the large size of our dataset (after splitting all tweets, based on whitespace, we are working with 228 million tokens), this is very unlikely to affect proper words. However, it helps limit the size of the input and output layers and thus helps with memory and time requirements of word2vec.

One advantage of word2vec is that the amount of training data that can be used is not limited by memory. Thus, while our tweet database is 27GB large, we can stream each tweet to the model during learning. We tokenize tweets by breaking them up at whitespace and emoji positions after removing all punctuation (excluding punctuation emoji such as ⁉). This ensures, each emoji is an individual token, even when no whitespace separates it from a word or multiple emoji follow directly after each other. From the stream of tokens, we filter out tokens that are hyperlinks or Twitter handles (e.g., *"@_CHINOSAUR"*).

In a final step, we split the model into two versions: one retaining all tokens and one limited to only the 845 emoji tokens. For each token, we keep the 300-dimensional feature vector, which are supplemented with frequency data for the emoji tokens. While we concentrate on the analysis of the emoji-only model in this article, both versions of the dataset are available as supplemental data to this article. We hope that they will be useful as a base upon which to build novel emoji entry methods.

7.2.1. Exploring the Emoji Model. Once a word embedding for the tweet corpus is generated, it can be used to query for emoji similarity. As stated earlier, we use the Euclidean norm to derive distances between pairs of emoji. In addition to basic ranking, word vectors also allow for more complex queries such as *"what is like 🎁 but not like 🎄"* (answers according to our model, e.g., are 🎂, *"cakeday,"* or 🎊, i.e., other occasions for presents that are not Christmas). For more information on vector compositionality in word embeddings trained with word2vec, such as in the example above, see Mikolov et al. [2013]. In this article, we do not further explore such multi-emoji relationships but concentrate on pairwise similarity.

However, to get a more general idea of whether the model indeed learned relationships between emoji, a visualization of clusters is useful. As the model is high dimensional, it cannot be plotted directly. We use an implementation [Pedregosa et al. 2011] of *t-distributed Stochastic Neighbor Embedding* (t-SNE) [van der Maaten and Hinton 2008] to reduce the number of dimensions from 300 down to 2 dimensions. Compared to other dimensionality reduction methods, t-SNE performs well at capturing local structure and making clusters in the data visually distinguishable.

The resulting visualization (see Figure 13) identifies several distinct clusters. On the right side, e.g., we can see all clock face emoji close to each other, as well as clusters of vehicles and writing-related objects. In the lower left, food and animals form individual clusters. Here, subclusters around more specific themes can be seen as well. For example, animals living in the water form a distinct group, as do sweets or fruit. In the upper left all smileys can be found. A closer look reveals child clusters for *loving*, *happy*, and *unhappy* faces. In general, we see some emoji that cluster together tightly and others that are more loosely coupled to others. Examples of close clusters are astrological signs (e.g., ♑), moon phases (e.g., 🌑), blood types (e.g., 🆎), or buildings (e.g., 🏢). However, other emoji do not seem to connect as closely, e.g., 🎲 is further away from others. But for some emoji this just shows that they are associated with multiple clusters or less tightly bound, compared to others in the cluster. For example, the 🐉 (dragon) is not as tightly bound to the animal cluster as other animal emoji. Note that while this visualization is useful in identifying groups of closely related emoji, it does not show any global similarity. Thus, two emoji that are on opposite ends of the visualization are not necessarily less similar than two emoji only half the width apart.

Instead of showing relatedness for all emoji, we can also have a look at smaller groupings. Particularly interesting is which emoji are most related to a given one. As we saw earlier, the annotation-based similarity is limited in that it can only capture a category-centric notion of similarity. Instead, our emoji model allows us to query for similarity due to related use of two emoji. In Table IV, we show a selection of emoji with their respective 10 most related emoji. As can be seen, deeper connections are revealed. For example, the 🗽 is detected as being closely related to 🇺🇸, while they share no common Unicode annotations. Similarly, 😢 and 💔 also share no tags, yet are closely related in our model. This is a first indication that our semantic model can indeed capture more detailed relationships between emoji. We confirm this later on in a crowdsourced experiment.

Another measure of model quality is whether it captures emoji diversity. It would, e.g., be problematic if all emoji were slightly similar, with no strong differences between pairs. However, Figure 14 shows that the similarity between emoji is actually heavily skewed. From all possible emoji pairings, only some are closely related, with the majority of pairings exhibiting medium similarity. This is a reassuring result as we indeed would expect the vast number of emoji pairs to only be slightly similar. For example, we would expect emoji such as 🐢 to only exhibit close relationships to a few other animal or ocean-related emoji. With a larger range of other emoji, like 🏧, ☃️, or 🙅, we would not expect much of an overlap in actual use and thus no large similarity.

We also wanted to make sure that similarity is indeed specific to the individual emoji. A naive approach could, e.g., always rank 😃 high, as this common emoji often occurs

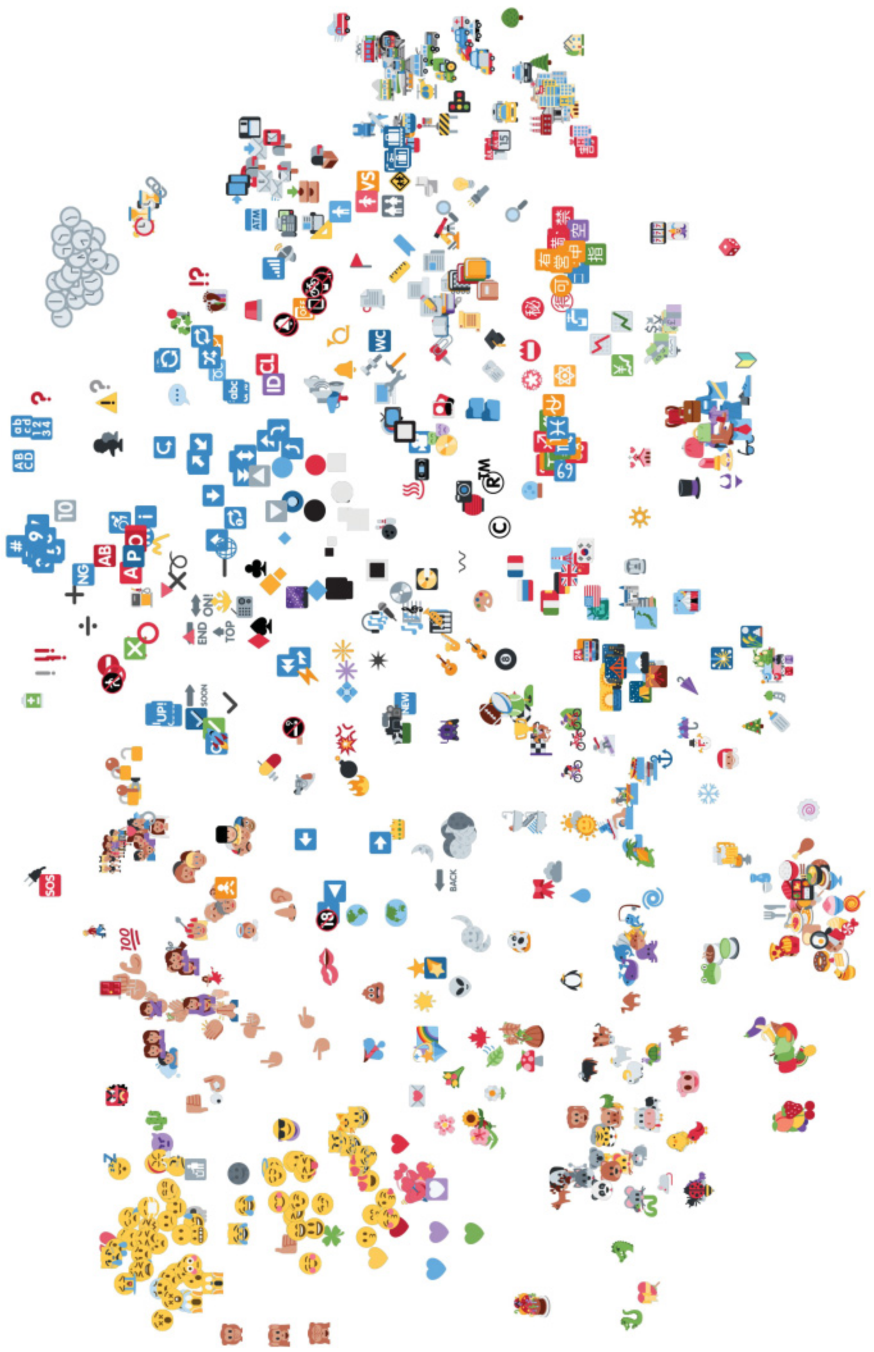


Fig. 13. From tweet data, we train a high-dimensional word vector model. However, this model cannot be directly visualized. Here, we show a two-dimensional embedding of the model, generated via *t-distributed Stochastic Neighbor Embedding* after an initial *principal component analysis* reduction. While the model contains vectors for every word in the training corpus, we show only the emoji here. Semantically similar emoji are clustered close together in this visualization.

Table IV. Top 10 Most Related Emoji for Several Example Emoji

Emoji	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
🐩	🐕	🐶	🐗	🐹	🐺	🐕	🐇	🐾	🐨	🐆
💖	💗	💝	💕	💘	💞	😘	💟	😽	💋	😇
⚽	🎾	🏆	🏉	⚾	⛳	🏁	🏀	🏇	🇬🇧	🏈
🗽	🇺🇸	🏙	🌉	🛄	🏯	🌇	🌃	⛲	🗾	🚄
😢	😔	😞	😥	💔	😿	😓	😟	😣	😪	😭
🍬	🍪	🍫	🍦	🍳	🍭	🍰	🍛	🍧	🍩	🍢
🎉	🎊	🎈	🎁	🎂	🎇	🎋	🎆	🍻	💃	🙌
🐙	🐡	🐠	🐟	🐚	🐬	🐋	🐳	🐊	🐢	🐇

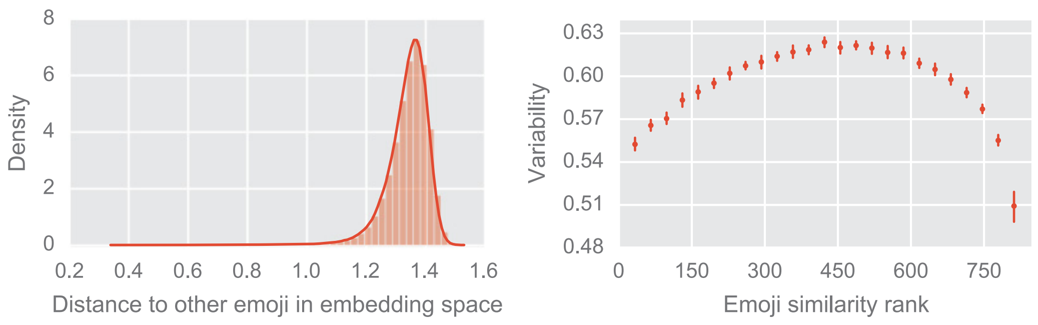


Fig. 14. After building the emoji similarity model, we can investigate how emoji relate to each other. On the left, we can see that similarity between emoji follows a rather skewed distribution. More closely related emoji pairs are found in the long left tail, while most pairs bunch together with medium distance. On the right, we explore whether there are some emoji that dominate the emoji similarity. For each of the 845 emoji, we sort the remaining 844 emoji according to the pairwise similarity. We can then, e.g., look at the set of all emoji ranked most similar. The variability (for each of the 844 possible list positions) describes how diverse such a set is. We aggregate several ranks here to better show the change in variability.

together with others, biasing the model toward frequent emoji. Hence, we looked at whether it was always the same emoji being highly ranked (e.g., 😃 again), or whether there was variability. Here, variability is defined as the size of the rank set (e.g., the set of all emoji ranked second most similar) over the number of emoji (845 in our case). For example, if each emoji is ranked most similar once, the variability would be 1.0. Figure 14 shows how variability varies over the range of ranks. There is no strong bias, and many different emoji are ranked first. For example, the most highly ranked emoji exhibit a variability of about 0.56, meaning that more than half the emoji appear at that rank (keep in mind this is always in relation to a second emoji; thus there are 845 individual rankings).

Finally, Figure 15 shows a hierarchical view of the emoji model. Where Figure 13 shows a global view of similarity, here related emoji are found in nearby nodes of the

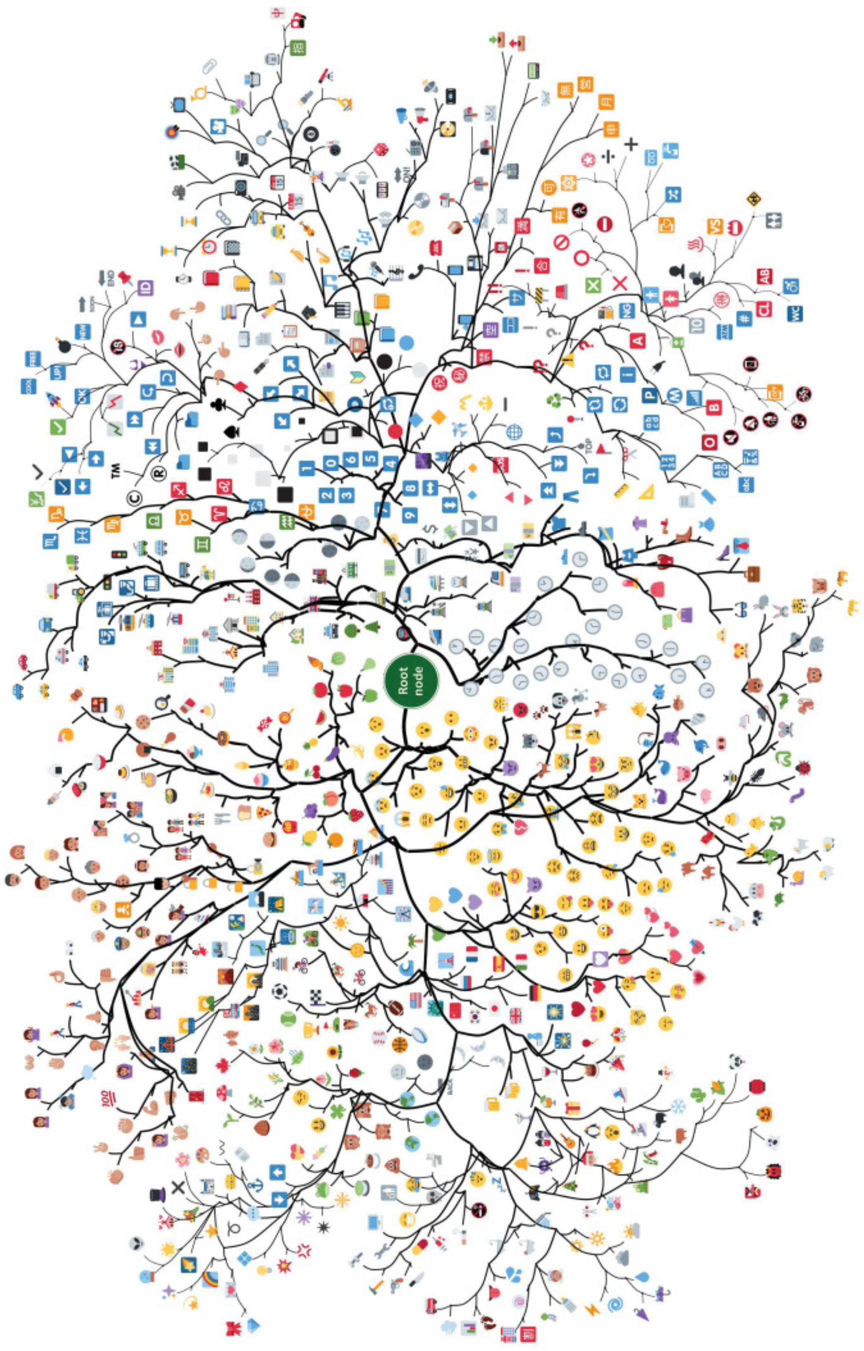
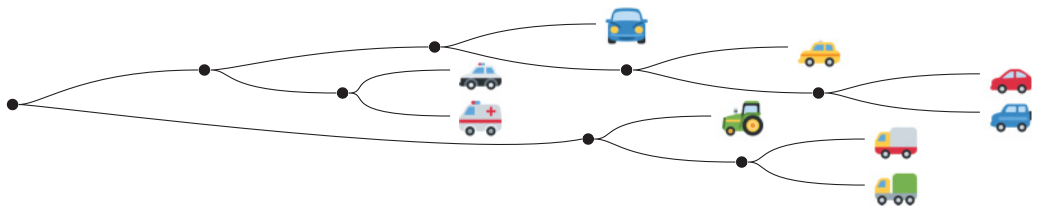


Fig. 15. We use agglomerative clustering to organize emoji into a relatedness-hierarchy. Here, we find related emoji close together in individual branches.

Table V. Top Five Most Related Tokens and Emoji for Several Example Tokens

Token	1st	2nd	3rd	4th	5th	1st	2nd	3rd	4th	5th
Birthday	bday	birthdayyyy	birthdayyyyy	18th	birfday	🎂	🎉	🎁	🎊	🎈
Sad	upset	depressed	upsetting	saddening	heartbreaking	😢	💔	😔	😞	😥
Vacation	vacay	vaca	holiday	vacations	holidays	🚤	🚣	🌞	🏡	🗻
Drunk	tipsy	stoned	intoxicated	sober	hammered	🍻	🍺	🍹	🌚	🍷
Love	adore	loooove	luv	loveee	loveeee	😚	💕	❤️	💞	💌

hierarchy. To generate this view, we use agglomerative clustering wherein nodes are successively merged (based on minimal distance) until a root node has been determined. We can look closer at one branch of the hierarchy:



Here, we see that vehicles cluster together, with emergency and heavy vehicles forming their own subclusters. However, such a rigid clustering also creates artifacts due to the merge criteria (we, e.g., only find 🚔 in a nearby branch). We use *Ward's criterion* here, which tries to minimize the variance of merged clusters. A different approach would be to minimize the maximum distance in a cluster, more heavily penalizing outliers in a cluster.

However, not all odd locations are due to the clustering. For example, note that 🗼 (showing the *Tokyo Tower*) is shown as very related to 🇫🇷—more so than to 🇯🇵. Thus, people apparently interpret it erroneously as the *Eiffel Tower*. Such interpretations can be challenging: Should keyboards arrange emoji as intended or as interpreted? And should the layout for Japan (where users probably recognize the tower) be different than elsewhere?

7.2.2. Exploring the Full Model. While we concentrate on the emoji-only model in this article, we actually have a more complete model for all tokens we used in training the emoji-only model. Here, we take a short look on how those other tokens relate to emoji and each other. Note that because we only trained on tweets, the word model has lower quality coverage than other models trained through word2vec. The usual approach is to include a larger secondary corpus to build a more in-depth model of word relationships. A common method, e.g., is to also use the English Wikipedia corpus during training.

Table V shows rankings for five example tokens. We show ranking for other word tokens and emoji separately, because other words are commonly much more related than emoji. Unsurprisingly, it can be seen there is strong semantic connection to different spellings of the same word. While we do not apply word stemming, this could be used to prune the vocabulary before building the model, in order to omit tokens such as *"birthdayyyyy."* But we can also see that the closest matching emoji for the given query tokens capture the given token quite well. For example, all birthday-related emoji match the theme and could be used in birthday messages.

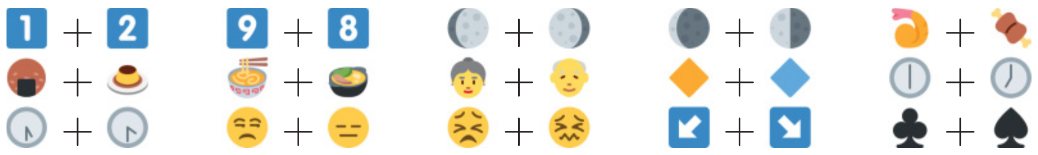
7.3. Evaluating Model Performance

So far, we have shown overviews of the models and examples that demonstrate the quality of the results. However, we wanted to confirm the similarity predictions with data from a second source. Hence, we gathered human ratings of emoji similarity to compare with our computed similarity data from both the tag-based and the semantic model.

We ran our crowdsourcing study on *CrowdFlower*,[31] which, in turn, recruits contributors from a larger range of channels.[32] In each task, we showed contributors 10 emoji pairs and for each asked them: *"How related are these emoji?"* In the task instructions, we further specified that: *"Two emoji could be related because they describe a similar concept, or because they are commonly used together. If you are unsure, think about how likely you are to use the two emoji together in a message."*

Contributors gave their response on a seven-point Likert scale ranging from *"unrelated"* to *"very related."*

To ensure rating quality, we designated a set of test questions for our task. We drew test questions from the emoji pairs where our semantic model indicates strongest similarity. After manually checking appropriateness and quality of those pairs, we selected the 15 most closely related ones as test questions. The selected pairs were



In each task, one of the questions was automatically included from this test question set. If contributors did not answer this question correctly (indicating some level of relationship), they were excluded from the task as they are deemed unreliable.

The rest of the emoji pairs to rate were drawn from two different groups: (1) the 5% most related emoji pairs (per our semantic model), and (2) the remainder of emoji pairs. We excluded emoji pairs already used as test questions from this selection. The total number of emoji pairs is thus given as: $\binom{845}{2} - 15 = 356,575$. From each of those groups, we randomly selected 45 pairs for inclusion in the crowdsourcing study. Hence, we use 45 out of the 17,829 pairs in the top 5% and 45 out of the 338,746 remaining pairs. For each of those 90 pairs, we collected 20 human ratings for a total of 1,800 trusted judgments. Those 1,800 judgments are trusted in the sense that CrowdFlower deems the raters reliable. Contributors are flagged as unreliable if they do not correctly responded to the test question in their task. Furthermore, before being served an actual tasks, contributors had to take a quiz composed of just test questions which, when failed, immediately flags them as unreliable.

In total, we had 52 participants contribute trusted judgments in our study (all ratings are available as supplemental material to this article). The median (and maximum) number of judgments per participants was 60. Contributors hailed from a wide selection of places, but most of them were from India (6), Serbia (5), Venezuela (4), Malaysia (4), and Turkey (3). Overall, trusted contributors provided 2525 judgments (if we include the 98 rejected participants, who failed the quiz or are untrusted, the total number of judgments is 3,265). However, 725 of those were test questions and only used to determine contributor reliability. This leaves us with a final 1,800 judgments. Agreement of raters, per Krippendorff's alpha, is 0.39 (95% bootstrapped confidence interval:

[31]https://www.crowdflower.com/.
[32]https://www.crowdflower.com/labor-channels/.

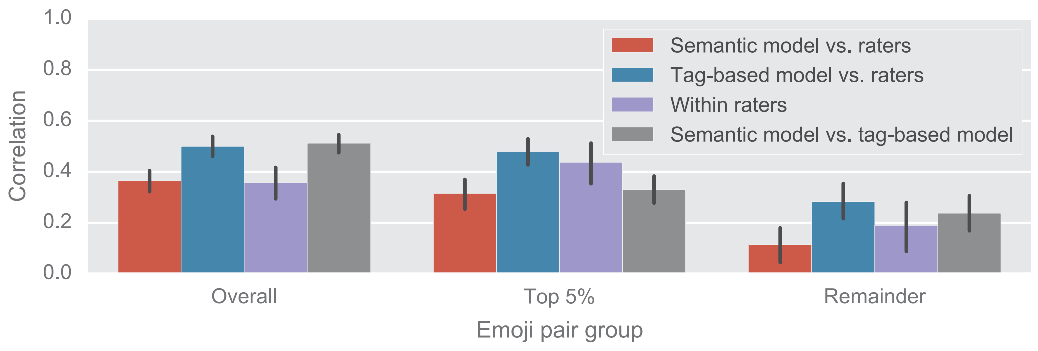


Fig. 16. Spearman's rank correlation coefficient for comparisons of human raters, the semantic, and tag-based models. For comparison, we also show how human raters correlate among themselves (random split between workers). All correlations are significant. Error bars show bootstrapped 95% confidence intervals.

0.30–0.48). This shows rating relationships between emoji was no easy task for raters. We can thus expect some noise when relating this back to our model predictions.

7.3.1. Results. In our analysis, we only consider ratings for the *top 5%* and *remainder* groups but exclude the test questions. Emoji pairs in the top 5% percentile, per our semantic model, received a median rating of 4 (higher equals more related) in the 900 crowdsourced ratings. On the other hand, the 900 emoji pairs from the remainder of the dataset only received a median rating of 2. We used Mann–Whitney's U test to compare the two groups and found a significant effect; $U = 544,873.0$, $p < 0.0001$. Thus, emoji pairs considered the most related by the semantic model are also seen as significantly more related by human raters.

Contributors also had an easier time rating emoji pairs from the top 5% group than from the remainder group. Krippendorff's alpha for the former is 0.42 (95% bootstrapped confidence interval: 0.32–0.52), while the ratings for the remainder group only show an alpha value of 0.20 (95% bootstrapped confidence interval: 0.07–0.35) This is likely due to the top 5% group having more within-group similarity (per our semantic model), while the remainder group contains many emoji pairs with no clear connection. In such a case, some raters might see connections where others see none. For example, the relationship between 💈 (barber pole) and 💇 is only apparent in some cultures and would likely lead to a very different rating by Americans and Germans.

So far, we have shown that our model's predictions are accurate on the level of distinguishing the top 5% most related emoji from the rest. However, we also wanted to see whether similarity estimation holds on the individual emoji pair level. For this, we check whether there is a significant correlation between the similarity scores given by the human raters and by the two models. All correlations in this section are given via Spearman's rank correlation coefficient r_s. See Figure 16, for an overview of how correlation varies for different groups and comparisons. Figure 16 also shows bootstrapped 95% confidence intervals for correlation, while we here only report averages. Over all emoji pairs with human ratings, we found that similarity predictions of the tag-based model and the human raters were correlated; $r_s = 0.50$, $p < 0.0001$. The distance predictions of the semantic model and the human similarity ratings showed similar correlation; $r_s = -0.37$, $p < 0.0001$. Correlation between random subsets of human raters is also at about the same level; $r_s = 0.39$, $p < 0.0001$.

Comparing human ratings and the tag-based model for the top 5% group shows significant correlation; $r_s = 0.48$, $p < 0.0001$. The same holds for the semantic model; $r_s = -0.31$, $p < 0.0001$. Similarly, human ratings and tag-based similarity predictions for the remainder group are significantly correlated; $r_s = 0.28$, $p < 0.0001$. Finally, a significant correlation is also observable when comparing human raters and semantic model predictions; $r_s = -0.11$, $p < 0.001$. Within the human raters, the top 5% group shows higher correlation than the remainder group; $r_s = 0.41$, $p < 0.0001$ and $r_s = 0.27$, $p < 0.0001$.

7.3.2. Discussion. In our crowdsourcing study, we saw significant correlation between human raters and both the tag-based and the semantic models. This shows that both models indeed capture differences in emoji similarity. However, there are differences between the two models. While both align with human raters, as we have seen, they sometimes disagree on the similarity of emoji. This is due to them focusing on different aspects of similarity: The tag-based model denoting whether two emoji show the same or a similar *thing*, while the semantic model captures a more fuzzy notion of relatedness, based on co-occurrence in or similar use. We hence need to further explore the differences between the two models.

7.4. Comparing Tag-Based and Semantic Emoji Models

As we have seen, both models make similarity predictions that align with human raters. Yet, their predictions are not the same and they both capture different aspects of emoji similarity. Here, we take a closer look at where the two models agree the most and the least.

In this section, we define agreement between the two models as the amount of overlap in the emoji similarity ranking they generate. For example, we could ask both models which 10 emoji are most similar to 🐶, giving us two sets 📜_A and 📜_B. The agreement is then given as follows:

$$\frac{|\text{📜}_A \cap \text{📜}_B|}{|\text{📜}_A \cup \text{📜}_B|}, \tag{3}$$

which is equivalent to the Jaccard similarity coefficient from Equation (1). Note that we do not consider rank position in this definition of agreement, but only whether it is included in the other result set or not. After all, for reasonably small result set sizes it does not matter much which rank position an emoji holds. For example, whether an emoji is 3rd or 4th in an eight item result set has no strong impact on retrieval time for that emoji if all items are shown at once (e.g., in the autocorrect bar).

Figure 17 shows agreement between the two models for varying sizes of rankings. For example, when we only consider the top-ranked emoji, the two models agree about 20% of the time. Naturally, increasing the size of the respective result sets increases agreement (at size 844 every other emoji would be included and agreement would be a guaranteed 100%). Overall, there is substantial disagreement between the two methods. However, the agreement is not equal over all emoji but differs depending on which emoji are considered. Figure 17 thus also shows the five emoji where the two models agree and disagree the most. We consider agreement over different ranking sizes when computing this average per-emoji agreement.

The two models have high agreement for emoji with a straightforward interpretation but disagree where interpretation is more open. For example, 🚄 is likely to allow for less flexibility in interpretation than 🔥. The later could be used to describe actual *fires*, trends that are *hot*, food that is *spicy*, people that are *attractive*, stores that are *busy*, and possibly many more scenarios. On the other hand, trains do not lend themselves to equal levels of ambiguity in interpretation.

We can take a closer look at these 10 extreme examples of agreement and disagreement between the two emoji models. For each of those emoji, Table VI shows the 10 most similar ranked emoji by each of the models. As can be seen, where there is agreement

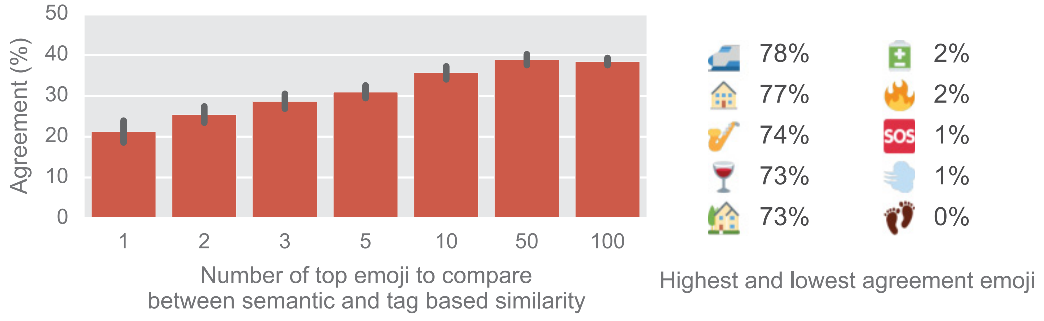


Fig. 17. We derive agreement between the semantic and the tag-based model from the overlap in the top-ranked emoji. For example, the two models, on average, have 30% of the top five emoji in common. Agreement increases as more ranks are considered. Error bars show bootstrapped 95% confidence intervals. On the right, we highlight the five emoji with the respective highest and lowest agreement (averaged over different rank sizes).

the sets largely overlap. Even the ranking within the two results is close. However, when there is strong disagreement, the two result sets are very dissimilar. Yet, the way the results differ can tell us a lot about the strong suit of each of the two models.

A good example of model differences is their ranking for the 🔋 emoji. The semantic model makes several predictions that use *battery* as a stand-in for general *energy level*. For example, drinking a ☕ to wake up, or the quality of a wireless connection 📶. We can also see a direct link between the battery and the phone it is often in, with emoji like 📵 or 📴. Likewise, 😵 could be used to indicate a "dead" phone but could also indicate that a user is very tired. While the presence of the 🐸 emoji might be puzzling at first, there is a strong connection between 🐸 and ☕. As an emoji combination, 🐸☕ is sometimes used to reference a meme on Twitter.[33] On the other hand, the tag-based model cannot make good predictions for the 🔋 emoji. This is primarily due to the fact that this emoji is only annotated with two tags: *"battery"* and *"object."* Yet the *"battery"* tag is not shared with any other emoji and the ranking thus degenerates to a random selection of other objects. One could add additional tags to emoji, but another option might be to find connections between tags. For example, there is also the *"electricity"* tag, which is used by ⚡ and 🔌, but, surprisingly, not by 🔋 itself. The same holds for the *"electric"* tag, which is used for five emoji, yet also excludes 🔋.

Poor tag coverage is a fundamental problem of the tag-based model. Overall, 1,175 different tags are used to describe the 845 emoji we investigated. Yet, while 🇬🇧 has 15 tags, 12 other emoji only are described by two tags (e.g., 👱, or 💃). These are the extremes though and the median number of tags per emoji is 5. The different tags are very unevenly distributed though. While 274 emoji are tagged as *"object,"* 215 as *"symbol,"* 199 as *"nature,"* and 184 as *"person,"* use count quickly goes down and 719 tags are only used by a single emoji. For tag-based similarity, these single-use tags are detrimental and do not help with establishing relationships between emoji.

An emoji with better tag coverage is 👣, which is annotated as *"body," "clothing," "footprint," "person,"* and *"print."* Hence, the tag-based model matches this emoji with other body parts and clothing items. This is an emoji the semantic model does more poorly with. However, it does identify a connection between 👣 and walking/movement-related emoji 💨 (dash symbol) and 🏃. In fact, this connection to walking is not identified by the tag-based model. Checking back with the emoji clusters, as

[33]http://blog.getemoji.com/post/134792876960/what-does-the-frog-and-teacup-emoji.

Table VI. The Semantic and The Tag-Based Model Differ in What They Capture About Emoji. Here, We Explore the Differences for the Five Emoji Where Agreement (As per Figure 17) Between the Two Models is Lowest and The Five Where it is Highest. Note That in This Set Only 🔥 Sees a High Frequency of Use (It Was The 23rd Most Common Emoji in Our Twitter Dataset)

	Emoji	Model	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
Least agreement	👣	Semantic	📊	♻️	📳	💨	🏃	🚡	🚋	⁉️	3️⃣	🚠
		Tag-based	✋	👅	👂	👃	👄	👌	👐	👔	👗	👘
	💨	Semantic	🚒	🏃	🚑	📊	🚂	🚃	🚙	🚋	⛽	🚈
		Tag-based	💦	💭	💬	💥	🎀	💫	💧	💝	💢	💍
	🆘	Semantic	🔻	😫	😣	🌟	🐾	⏰	🏠	😓	😾	😩
		Tag-based	🈚	🈵	🆒	🆕	®	🉐	🆗	©	🆖	🈹
	🔥	Semantic	😵	🐂	💣	💽	♨️	‼️	💥	🔉	👿	🈴
		Tag-based	🔨	🔬	🔧	🔭	✂️	🔩	🔍	🔎	🔪	🔦
	🔋	Semantic	🔌	😟	📶	☕	💿	📵	😬	🐸	😵	📴
		Tag-based	📓	✏️	🔗	🔬	🔧	💣	🔖	⛲	🔔	💮
Most agreement	🚅	Semantic	🚅	🚆	🚈	🚉	🚊	🚝	🚂	🚇	🚃	🚠
		Tag-based	🚅	🚆	🚉	🚈	🚂	🚠	🚃	🚞	🚝	🚢
	🏠	Semantic	🏡	🏢	🏨	🏛	🏣	⛪	🏬	🏤	🏥	🏫
		Tag-based	🏡	🏢	🏨	🏛	🏭	🏬	🏰	🏫	🏩	🏪
	🎷	Semantic	🎺	🎻	🎹	🎼	🎸	🎶	🎤	🀄	📯	🎴
		Tag-based	🎺	🎻	🎸	🎹	🎼	🎵	🎶	🎧	🎤	🎇
	🍷	Semantic	🍸	🍹	🍻	🍶	🍺	🍴	💊	🍝	🍵	🍚
		Tag-based	🍸	🍹	🍶	🍺	🍻	🍵	🍫	☕	🍣	🍞
	🏡	Semantic	🏠	🏨	🏢	🏛	🏤	🏣	🏥	🚗	🏬	👪
		Tag-based	🏠	🏨	🏢	🏭	🏛	🏩	🏫	🏯	🏰	🏪

shown in Figure 13, we can see that 👣 does not connect well to larger agglomerations of related emoji. Additional training data might help strengthen some connections: We only collected 14,609 tweets containing 👣.

Overall, the closer look at these emoji strengthens the impression that *object-related* emoji allow for more consensus in similarity estimation than emoji allowing for more abstract interpretations. However, it is precisely these emoji that add to the vibrancy of emoji use in messaging. Adding a ⛺ when going camping makes the message more colorful and playful. Yet, the ⛺ does not allow for a range of expression as wide as, e.g., the 🔥 emoji.

8. CONCLUSION

In this article, we have outlined and explored the space of emoji text entry. Starting from an introduction to emoji, we first looked at how emoji are used and how well current input methods support them. As we showed in a study of the state of the art, arrangement of emoji into categories is a weak spot of current emoji keyboards. Such organizational problems were also described in the study of *EmojiZoom* [Pohl et al. 2016]. Participants preferred the two-dimensional layout per the Unicode sorting to the ordering of the Google keyboard (which slightly deviates from the Unicode one). Hence, we identify improvements to emoji ordering as an important direction for research on emoji.

The large number of emoji makes it hard to manually optimize emoji ordering though. We have motivated building an emoji similarity model as a way to allow automatic optimization of or reasoning about emoji layouts. This would also scale to different cultural context, e.g., by only considering German language tweets for a German emoji model. We have presented two candidate models to establish emoji similarity: (1) based on emoji annotations and (2) based on semantic information on emoji. In a crowdsourcing study, we have shown that both models correlate with human raters. Yet, what each model captures is quite different. Direct comparison of the two models indicates that semantic similarity is able to capture more nuanced relationships of emoji. However, this is prone to more noise than tag-based comparison. On the other hand, tag-based approaches have poor coverage for many emoji. As the semantic similarity approach does not rely on manual annotation, it scales better with the large number of possible emoji pairings.

Emoji have seen strong uptake and cultural influence. The fact that they are part of the Unicode standard also gives these characters a likely higher permanence than application-specific smileys or stickers. The *Unicode Character Encoding Stability Policies*[34] explicitly state that *"Once a character is encoded, it will not be moved or removed."* Hence, emoji are here to stay.

But, as we discussed, entering emoji is quite different than entering text. Emoji have some unique characteristics, such as their lack of a clear phonetic interpretation and their visual nature. Existing keyboard layout optimization also does not translate well to emoji entry. Instead of a layout optimized for entering character sequences, emoji require input methods that optimize for search and exploration. Our emoji similarity model can inform this process. As we have shown, this model can be used to structure the emoji space and thus, e.g., inform category assignment. It could also be used to retrieve emoji fitting a current context (text or other emoji). Yet, while making emoji available via a retrieval method might work for a few users, it is likely inadequate for a large number of users. Exploration of available emoji is a critical aspect in emoji entry. While familiarity with all letters of Latin script is assumed in other keyboards, emoji keyboards cannot make the same assumption.

Entering emoji still means entering text, not uploading an image. Compared to entering Latin script though, appearance plays a crucial role. For example, while 📓, 📔, 📕, 📗, 📘, and 📙 all show closed books, they each have a distinct color. While there might be a clear mapping when users actually want to describe, e.g., a red book, most of the time the choice of book emoji comes down to taste. The word *book* itself does not pose the same trouble. It is only once the *book* becomes a *notebook* or *novel* that the meaning changes. As the choice of book emoji thus depends on taste and might be different depending on mood, it is not sufficient to just show one of them. Exploring the

[34]http://unicode.org/policies/stability_policy.html.

U+0CA0, U+5F, U+0CA0 ⇒ ಠ_ಠ

U+25E1, U+203F, U+25E1, U+273F ⇒ ◡‿◡✿

U+28, U+256F, U+B0, U+25A1, U+B0, U+29,
U+256F, U+FE35, U+253B, U+2501, U+253B ⇒ (╯°□°)╯︵┻━┻

U+295, U+2022, U+1D25, U+2022, U+294 ⇒ ʕ•ᴥ•ʔ

U+28, U+2310, U+25A0, U+5F, U+25A0, U+29 ⇒ (⌐■_■)

U+25D5, U+203F, U+25D5 ⇒ ◕‿◕

Fig. 18. Unicode characters can be combined to build emoticons more diverse and expressive than those restricted to only ASCII characters. Shown here are *kaomoji*: the Japanese form of emoticons.

available emoji and picking the right one for the current situation are critical aspects of emoji entry.

This visual variability of emoji relates back to where we initially started: Emoji can add playfulness to messaging and allow users to express themselves on a new level. The most frequently used emoji in our Twitter dataset all add to a text on the emotional level. Whether a message is followed by 😍 or 😒 can dramatically change the tone. For example, consider receiving either *"Susan's coming over later* 😍*,"* or *"Susan's coming over later* 😒*."* Emoji are a first to make access to this kind of expression easy and ubiquitous.

Yet, the emoji entry we explored in this article is but one method to add such level of expression to messaging. As already noted in Section 1, emoticons are the classical example of this kind of content. However, there are many possible extensions here and other means to give expressive power to users. We take a closer look at those methods, "beyond emoji entry," that are possible paths to support the underlying goal: allow users to communicate in a playful and casual way.

8.1. Outlook: Beyond Emoji

So far, we have focused specifically on emoji, yet the Unicode standard actually enables a much wider range of textual expressions. As shown in Section 1, several of the characters encoded in Unicode can also be combined to form emoticons (*emotion icons*). Other characters can be appropriated to stylize text.

Where classic emoticons like `:)`, `O_o`, or `:D` only make use of ASCII characters, a much wider variety of emoticons is possible when including more exotic characters (for some examples see Figure 18). One of the more common ones, e.g., is the *look of disapproval*[35]:ಠ_ಠ. This emoticon repurposes the ಠ (*ha*) letter from the Kannada alphabet to symbolize an eye. However, entering such emoticons is not an easy task. Some soft keyboards make emoticons available (e.g., the Windows Phone 8.1 keyboard has a tab for this which contains emoticons such as ♡.♡), but the selection is small. A large number of third party apps try to fill this gap and include emoticons ready for use via copy and paste—an approach also supported by dedicated websites, such as http://www.disapprovallook.com disapprovallook.com. The problem with all these methods is that it only allows users to use what is already given. However, it makes entering novel emoticons restrictively hard and relegates experimenting with textual expression to the desktop. While users could install a range of keyboard languages to access more characters, such keyboards are tailored to text entry in that language, not repurposing of individual characters—some keyboards also use input methods not familiar to regular QWERTY users. As keyboards on mobile devices have moved to have

[35]http://knowyourmeme.com/memes/ಠ_ಠ-look-of-disapproval.

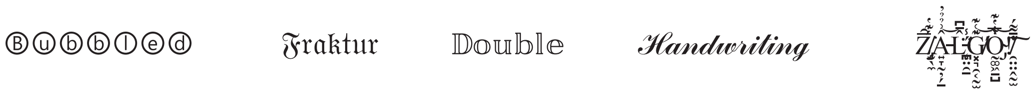


Fig. 19. Unicode characters can be used to stylize text. The first four examples use characters from the *enclosed* and *mathematical alphanumerics* blocks, while the zalgo text uses *combining marks*.

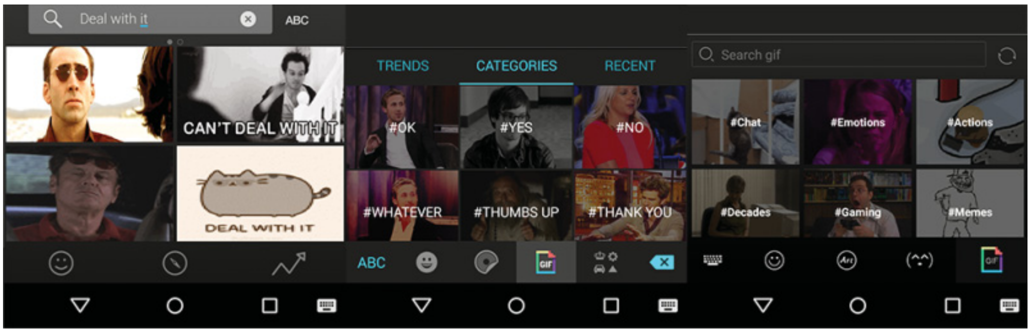


Fig. 20. Some third-party keyboards allow GIF entry. Shown here are (1) Fleksy, (2) Ginger Keyboard, and (3) EmojiKeyboard Pro.

dedicated emoji entry modes, we can envision keyboards moving beyond this state of the art and making an even larger character set available.

Unicode characters can also be used to stylize Latin script (see Figure 19). Here, instead of composition, appearance is changed by using stand-in characters. For example, instead of `U+6B`, one can use `U+1D528` when entering the letter *k* to get the fraktur version 𝔨. The Unicode *mathematical alphanumerics* block, e.g., contains several stylized variants of Latin letters, which allows us to effectively control the font of messages where no font information can be transmitted. Finally, Unicode also contains combining marks that are designed to modify other characters. While this can, e.g., be used as a way to add accents to characters (such as in times.ttfa), it also allows us to scramble text. One variant of this is *zalgo*[36] text (Figure 19, right), which strives to give text an appearance alluding to insanity. As with Unicode emoticons, there is currently no convenient way to add combining marks to characters on mobile devices. However, some keyboards relegate characters with diacritics to menus attached to the respective keys (e.g., the German keyboard on Windows Phone 8.1 offers selection of a or from a popup, shown after long pressing the a key). Such a mechanism could potentially be extended to larger sets of alternative characters and diacritical marks.

But Unicode characters, as emoji or emoticons, are the only one way expression is added to messages. Another popular approach is to add images, e.g., in the form of memes or animated GIFs. Such content was just recently shown by Bakhshi et al. [2016] to be significantly more engaging than just text. The Unicode consortium itself notes that, in the long run, applications should support arbitrary images in text:

> *The longer-term goal for implementations should be to support embedded graphics, in addition to the emoji characters. Embedded graphics allow arbitrary emoji symbols, and are not dependent on additional Unicode encoding.*[37]

[36]http://knowyourmeme.com/memes/zalgo.
[37]http://www.unicode.org/reports/tr51/#Longer_Term.

Today, several third party keyboards such as *Flesky*,[38] *Ginger,*[39] or *EmojiKeyboard Pro*[40] already also allow *"typing"* images. Such images are either found by browsing tags or searching with a phrase (see Figure 20 for several interface examples). One common use of such images is in the form of a *reaction gif*,[41] where an animated gif is used to represent approval, happiness, shrugging, or eye rolling.

With emoticons, emoji, and images, mobile text entry faces two challenges: (1) how to make the large set of existing content available, and (2) how to allow users to create their own content for personalized expression. For emoji, we only need to solve (1), yet if arbitrary images and emoticons are allowed, creation becomes a crucial aspect. However, while there are several web-based tools and browser extensions for styling text or creating memes, many of those are not easily accessible on mobile devices. Yet being mobile need not mean that users are necessarily restricted to existing content. In fact, as demonstrated by *12Pixels*, custom-tailored applications are able to allow creative making on mobile device much more restricted than today's smartphones [Willis and Poupyrev 2010]. As text can often be insufficient as a means to communicate emotions, this kind of visual language can fill in the gaps and offer an additional means of expression. If we restrict input to just text or make it hard to express emotions, we cannot communicate with each other as effectively as possible.

REFERENCES

Sho Aoki and Osamu Uchida. 2011. A method for automatically generating the emotional vectors of emoticons using weblog articles. In *Proceedings of the 10th WSEAS International Conference on Applied Computer and Applied Computational Science (ACACOS'11)*. 132–136.

Gilles Bailly, Antti Oulasvirta, Duncan P. Brumby, and Andrew Howes. 2014. Model of visual search and selection time in linear menus. In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems (CHI'14)*. ACM, New York, NY, 3865–3874. DOI:http://dx.doi.org/10.1145/2556288.2557093

Saeideh Bakhshi, David A. Shamma, Lyndon Kennedy, Yale Song, Paloma de Juan, and Joseph 'Jofish' Kaye. 2016. Fast, cheap, and good: Why animated GIFs engage us. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI'16)*. ACM Press, New York, NY, 575–586. DOI:http://dx.doi.org/10.1145/2858036.2858532

Xiaojun Bi, Barton A. Smith, and Shumin Zhai. 2012. Multilingual touchscreen keyboard design and optimization. *Human Computer Interaction* 27, 4 (2012), 352–382. DOI:http://dx.doi.org/10.1080/07370024.2012.678241

Matthias Böhmer, Christian Lander, Sven Gehring, Duncan P. Brumby, and Antonio Krüger. 2014. Interrupted by a phone call: Exploring designs for lowering the impact of call notifications for smartphone users. In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems (CHI'14)*. ACM Press, New York, NY, 3045–3054. DOI:http://dx.doi.org/10.1145/2556288.2557066

Spencer Cappallo, Thomas Mensink, and Cees G. M. Snoek. 2015a. Image2Emoji: Zero-shot emoji prediction for visual media. In *Proceedings of the 23rd ACM International Conference on Multimedia*. 1311–1314. DOI:http://dx.doi.org/10.1145/2733373.2806335

Spencer Cappallo, Thomas Mensink, and Cees G. M. Snoek. 2015b. Query-by-Emoji video search. In *Proceedings of the 23rd ACM International Conference on Multimedia*. 735–736. DOI:http://dx.doi.org/10.1145/2733373.2807961

Heeryon Cho and Toru Ishida. 2011. Exploring cultural differences in pictogram interpretations. In *The Language Grid*, Toru Ishida (Ed.). Springer, Berlin, Heidelberg, 133–148. DOI:http://dx.doi.org/10.1007/978-3-642-21178-2_9

Karen Church, Denzil Ferreira, Nikola Banovic, and Kent Lyons. 2015. Understanding the challenges of mobile phone usage data. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI'15)*. ACM Press, New York, NY, 504–514. DOI:http://dx.doi.org/10.1145/2785830.2785891

[38]http://fleksy.com.

[39]https://play.google.com/store/apps/details?id=com.gingersoftware.android.keyboard.

[40]https://play.google.com/store/apps/details?id=com.emoji.coolkeyboard.

[41]See, e.g., http://giphy.com/categories/reactions/.

Mark Dunlop and John Levine. 2012. Multidimensional Pareto optimization of touchscreen keyboards for speed, familiarity and improved spell checking. In *Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems (CHI'12)*. ACM Press, New York, NY, 2669–2678. DOI:http://dx.doi.org/10.1145/2207676.2208659

Thomas B. Fitzpatrick. 1988. The validity and practicality of sun-reactive skin types I through VI. *Archives of Dermatology* 124, 6 (1988), 869–871. DOI:http://dx.doi.org/10.1001/archderm.1988.01670060015008

Jeffrey T. Hancock, Christopher Landrigan, and Courtney Silver. 2007. Expressing emotion in text-based communication. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'07)*. ACM Press, New York, NY, 929–932. DOI:http://dx.doi.org/10.1145/1240624.1240764

Niels Henze, Enrico Rukzio, and Susanne Boll. 2012. Observational and experimental investigation of typing behaviour using virtual keyboards for mobile devices. In *Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems (CHI'12)*. ACM, 2659–2668. DOI:http://dx.doi.org/10.1145/2208636.2208658

Joris H. Janssen, Wijnand A. Ijsselsteijn, and Joyce H. D. M. Westerink. 2014. How affective technologies can influence intimate interactions and improve social connectedness. *International Journal of Human Computer Studies* 72, 1 (2014), 33–43. DOI:http://dx.doi.org/10.1016/j.ijhcs.2013.09.007

Andreas Karrenbauer and Antti Oulasvirta. 2014. Improvements to keyboard optimization with integer programming. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST'14)*. ACM Press, New York, NY, 621–626. DOI:http://dx.doi.org/10.1145/2642918.2647382

David E. Kieras and Anthony J. Hornof. 2014. Towards accurate and practical predictive models of active-vision-based visual search. In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems (CHI'14)*. ACM, New York, NY, 3875–3884. DOI:http://dx.doi.org/10.1145/2556288.2557324

Per Ola Kristensson and Keith Vertanen. 2014. The inviscid text entry rate and its application as a grand goal for mobile text entry. In *Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices & Services (MobileHCI'14)*. ACM Press, New York, NY, 335–338. DOI:http://dx.doi.org/10.1145/2628363.2628405

Henry Kučera and W. Nelson Francis. 1967. *Computational Analysis of Present-Day American English*. Brown University Press, Providence, RI.

Lisa Lebduska. 2015. Emoji, Emoji, What for art thou? *Harlot* 12 (2015). http://harlotofthearts.org/index.php/harlot/article/view/186/157.

Ying Liu and Qiqun Wang. 2007. Chinese pinyin phrasal input on mobile phone: Usability and developing trends. In *Proceedings of the 4th International Conference on Mobile Technology, Applications, and Systems and the 1st International Symposium on Computer Human Interaction in Mobile Technology (Mobility'07)*, Vol. 07. ACM Press, New York, NY, 540–546. DOI:http://dx.doi.org/10.1145/1378063.1378151

Daniel P. Lopresti and Andrew Tomkins. 1993. Approximate matching of hand-drawn pictograms. In *Proceedings of the Third International Workshop on Frontiers in Handwriting Recognition*. 102–111.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*.

Hannah Miller, Jacob Thebault-Spieker, Shuo Chang, Isaac Johnson, Loren Terveen, and Brent Hecht. 2016. Blissfully happy or ready to fight: Varying interpretations of emoji. In *Proceedings of International AAAI Conference on Web and Social Media*. 259–268.

Marketta Niemelä and Jukka Saarinen. 2000. Visual search for grouped versus ungrouped icons in a computer interface. *Human Factors: The Journal of the Human Factors and Ergonomics Society* 42, 4 (2000), 630–635. DOI:http://dx.doi.org/10.1518/001872000779697999

Petra Kralj Novak, Jasmina Smailovic, Borut Sluban, and Igor Mozetic. 2015. Sentiment of emojis. *PLoS ONE* 10, 12 (2015), 1–19. DOI:http://dx.doi.org/10.1371/journal.pone.0144296

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesna. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

Henning Pohl and Roderick Murray-Smith. 2013. Focused and casual interactions: Allowing users to vary their level of engagement. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'13)*. ACM, New York, NY, 2223–2232. DOI:http://dx.doi.org/10.1145/2470654.2481307

Henning Pohl, Dennis Stanke, and Michael Rohs. 2016. EmojiZoom: Emoji entry via large overview maps. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services Companion (MobileHCI'16)*. DOI:http://dx.doi.org/10.1145/2935334.2935382

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valetta, Malta, 45–50.

Jared Suttles and Nancy Ide. 2013. Distant supervision for emotion classification with discrete binary values. In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing'13)*, Vol. 7817 LNCS. 121–136. DOI:http://dx.doi.org/10.1007/978-3-642-37256-8_11

C. C. Tappert, C. Y. Suen, and T. Wakahara. 1990. The state of the art in online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12, 8 (1990), 787–808. DOI:http://dx.doi.org/10.1109/34.57669

Chad C. Tossell, Philip Kortum, Clayton Shepard, Laura H. Barg-Walkow, Ahmad Rahmati, and Lin Zhong. 2012. A longitudinal study of emoticon use in text messaging from smartphones. *Computers in Human Behavior* 28, 2 (2012), 659–663. DOI:http://dx.doi.org/10.1016/j.chb.2011.11.012

Yuki Urabe, Rafal Rzepka, and Kenji Araki. 2013. Emoticon recommendation system for effective communication. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM'13)*. ACM, New York, NY, 1460–1461. DOI:http://dx.doi.org/10.1145/2492517.2492594

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9 (2008), 2579–2605. DOI:http://dx.doi.org/10.1007/s10479-011-0841-3

Leticia Vidal, Gastón Ares, and Sara R. Jaeger. 2015. Use of emoticon and emoji in tweets for food-related emotional expression. *Food Quality and Preference* 49 (2015), 119–128. DOI:http://dx.doi.org/10.1016/j.foodqual.2015.12.002

Daryl Weir, Henning Pohl, Simon Rogers, Keith Vertanen, and Per Ola Kristensson. 2014. uncertain text entry on mobile devices. In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems (CHI'14)*. ACM, New York, NY, 2307–2316. DOI:http://dx.doi.org/10.1145/2556288.2557412

Karl D. D. Willis and Ivan Poupyrev. 2010. 12Pixels: Exploring social drawing on mobile phones. In *Proceedings of the 8th International Conference on Pervasive Computing (Pervasive'10)*. Springer, Berlin, 391–408. DOI:http://dx.doi.org/10.1007/978-3-642-12654-3_23

Shumin Zhai, Michael Hunter, and Barton A. Smith. 2000. The metropolis keyboard—An exploration of quantitative techniques for virtual keyboard design. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology (UIST'00)*. ACM, New York, NY, 119–128. DOI:http://dx.doi.org/10.1145/354401.354424

Received January 2016; revised December 2016; accepted December 2016