Assessing Emoji Use in Modern Text Processing Tools 👨🏽‍🔬

Abu Awal Md Shoeb
Dept. of Computer Science
Rutgers University
New Brunswick, NJ, USA
abu.shoeb@rutgers.edu

Gerard de Melo
Hasso Plattner Institute /
University of Potsdam
Potsdam, Germany
gdm@demelo.org

Abstract

Emojis have become ubiquitous in digital communication, due to their visual appeal as well as their ability to vividly convey human emotion, among other factors. This also leads to an increased need for systems and tools to operate on text containing emojis. In this study, we assess this support by considering test sets of tweets with emojis, based on which we perform a series of experiments investigating the ability of prominent NLP and text processing tools to adequately process them. In particular, we consider tokenization, part-of-speech tagging, dependency parsing, as well as sentiment analysis. Our findings show that many systems still have notable shortcomings when operating on text containing emojis.

1 Introduction

In our modern digital era, interpersonal communication often takes place via online channels such as instant messaging, email, social media, etc. This entails an increasing need for tools that operate on the resulting digital data. For instance, online conversations can be invaluable sources of insights that reveal fine-grained consumer preferences with regard to products, services, or businesses (Dong and de Melo, 2018).

However, the shifts in modality and medium also shape the way we express ourselves, making it increasingly natural for us to embed emojis, images, hashtags into our conversations. In this paper, we focus specifically on emojis, which have recently become fairly ubiquitous in digital communication, with a 2017 study reporting 5 billion emojis being sent daily just on Facebook Messenger (Burge, 2017). Emojis are textual elements that are encoded as characters but rendered as small digital images or icons that can be used to express an idea or emotion.

Goals. Due to their increasing prominence, there is a growing need to properly handle emojis whenever one deals with text. We consider a set of popular NLP tools and empirically assess to what extent they support emojis across a set of standard tasks, encompassing tokenization, part-of-speech tagging, dependency parsing, and sentiment analysis.

Although emojis can be encoded as Unicode characters, there are unique properties of emoji encoding that merit special consideration, such as skin tone modifiers and composite emoji incorporating multiple basic emojis. Moreover, text harboring emojis may adhere to subtly different conventions than more traditional forms of text, e.g., with regard to token and sentence boundaries. Emojis can take the place of words with different parts-of-speech and assume different grammatical roles. Finally, emojis may of course also alter the semantics of the text, which in turn may, for instance, affect its sentiment polarity.

Overview. For our analysis, we draw primarily on social media and study diverse forms of emoji use. We run a series of experiments on such data evaluating each NLP tool to observe its behaviour at different stages in the processing pipeline. The results show that current tools have notable deficiencies in coping with modern emoji use in text.

2 Related Work

While emoji characters have a long history, they have substantially grown in popularity since their incorporation into Unicode 6.0 in 2010 followed by increasing support for them on mobile devices. Accordingly, numerous studies have sought to explain how the broad availability of emojis has affected human communication, considering grammatical, semantic, as well as pragmatic aspects (Kaye et al., 2017; McCulloch, 2019). Only few studies have specifically considered some of the more advanced technical possibilities that the Unicode standard affords, such as zero width joiners to express more

complex concepts. For instance, with regard to emoji skin tone modifiers, Robertson et al. (2020) study in depth how the use of such modifiers varies on social media, including cases of users modulating their skintone, i.e., using a different tone than the one they usually pick.

Given the widespread use of emojis in everyday communication, there is an increasing need for NLP tools that can handle them. Prominent NLP toolkits such as Stanford's Stanza (Qi et al., 2020) and NLTK (Bird et al., 2009) power a wide range of user-facing applications. A number of reports compare the pros and cons of popular NLP libraries (Wolff, 2020; Kozaczko, 2018; Choudhury, 2019; Bilyk, 2020), but these primarily consider the features and popularity of the tools, as well as their performance. There have not been studies assessing them with regard to their ability to cope with modern emoji-laden text. Since emojis are becoming increasingly ubiquitous, it is crucial for developers and institutions deploying such software to know whether it can properly handle the kinds of text that nowadays may quite likely arrive as input data. In many real-world settings, applications and services are expected to operate on text containing emojis, and thus it is important to investigate these capabilities.

Many academic studies present new models for particular NLP tasks relating to emojis. For instance, Felbo et al. (2017) developed an emoji prediction model for tweets. Weerasooriya et al. (2016) discussed how to extract essential keywords from a tweet using NLP tools. Cohn et al. (2019) attempted to understand the use of emojis from a grammatical perspective, seeking to determine the parts-of-speech of emoji occurrences in a sentence or tweet. Owoputi et al. (2013) proposed an improved part-of-speech tagging model for online conversational text based on word clusters. Proisl (2018) developed a part-of-speech tagger for German social media and Kong et al. (2014) developed a dependency parser for English tweets. However, such work mostly targets just one specific task and is typically not well-integrated with common open source toolkits, which we focus on in our study.

3 Experimental Data

As we wish to assess the support of emojis provided by different text processing tools, we first consider some of the different cases of emoji use that one may encounter, in order to compile relevant data.

3.1 Emoji Use in Text

Emojis can appear in a sentence or tweet in different circumstances. They may show up at the beginning or at the end of a tweet. Likewise, they may appear as part of a series of emojis separated by spaces, or can be clustered within a text without any interleaved spacing. Based on observations on a collection of tweets crawled from Twitter (Shoeb et al., 2019), we defined a series of cases distinguishing different aspects of emoji use, including the number of emojis (i.e., single emojis vs. multiple emojis), position of emojis, the use of skin tone modifiers, and so on.

For skin tone emojis, the Unicode standard adopts the Fitzpatrick Scale (Fitzpatrick, 1975), according to which the skin tone for selected emojis can be modulated with five different color settings: 🏻 Light Skin Tone (e.g., 👍🏻), 🏼 Medium-Light Skin Tone (e.g., 👍🏼), 🏽 Medium Skin Tone (e.g., 👍🏽), 🏾 Medium-Dark Skin Tone (e.g., 👍🏾), and 🏿 Dark Skin Tone (e.g., 👍🏿). Internally, an Emoji Modifier Sequence is assumed when a modifier character follows a supported base emoji character, resulting in a single emoji with skin tone.

Some characters now classified emojis are encoded in Plane 0, the Basic Multilingual Plane, where 16 bits suffice to encode individual characters. However, the majority of emojis reside in Plane 1, the Supplementary Multilingual Plane, which in the past had mainly been reserved for rare historic scripts. When including the latter, individual characters can no longer be encoded directly within just 16 bits. Hence, we consider whether a tool handles both non-BMP and BMP emojis.

Emojis with Zero Width Joiner (ZWJ) join two or more other characters together in sequence to compose a new one. Popular emoji ZWJ sequences include group ones such as the family emoji 👨‍👧‍👧, consisting in this case of Man, Woman, Girl, Boy emojis, and encoded by combining 👨 Man, the U+200D ZWJ code, 👩 Woman, U+200D again, 👧 Girl, U+200D, and finally 👦 Boy. These are rendered as a single emoji on supported platforms.

3.2 Tweet Selection

Given the different cases of emoji use discussed above, we searched for relevant examples in a collection of tweets that we compiled earlier from Twitter (Shoeb et al., 2019). The purpose of this endeavor was to assemble a collection of tweets based on a set of most frequently used emojis so that ev-

Tweets	Count	%
Total	22.3 M	100
Unique	21.4 M	95.84
No more than 5 tweets from one user	20.8 M	93.27
Only single emoji	5.67 M	25.38
Multiple emojis	16.48 M	73.77
Skin tone modifiers emojis	1.31 M	5.85
Light skin tone emojis	382 K	1.71
Medium light skin tone emojis	386 K	1.73
Medium skin tone emojis	337 K	1.51
Medium dark skin Tone emojis	274 K	1.23
Dark skin tone emojis	53 K	0.24
Zero Width Joiner (ZWJ) emojis	97 K	0.43

Table 1: Emoji Centric Twitter Corpus statistics – the distribution of emojis over the ~22 million tweets with regard to the considered emoji use in text

ery single tweet contains at least one emoji. The popularity of the emojis was determined using Novak et al. (2015) and Emoji Tracker[1], a website that monitors the use of emojis on Twitter in real time. In total, we obtained a set of 22.3 million tweets over a span of one year. This collection, named as EmoTag, is readily available online[2]. Table 1 provides corresponding statistics of our collection, showing that even rare phenomena do occur in substantial numbers of tweets.

Next, we chose representative samples for each case. We restricted our search to English language tweets and ensured that not all tweets simply consisted of URLs or mentions. The latter are fairly common on social media, and since it would not be very uncommon for a text processing tool to encounter them in tweets, we did also incorporate a few such tweets along with tweets containing genuine text. Ultimately, we obtain a diverse collection of short input texts, including different skintones, ZWJ emojis, and other cases mentioned in Section 3.1 and Table 1.

We drew upon the compiled input texts for assessments with regard to different NLP tasks. The following sections describe each of the considered tasks, i.e., Tokenization (Section 4), Part-of-Speech Tagging (Section 5), Dependency Parsing (Section 6), and Sentiment Analysis (Section 7) separately. The full dataset for the following experiments can be found at `http://emoji.nlproc.org`.

4 Tokenization

Tokenization is the act of breaking up a sequence of strings into a sequence of basic pieces such as words, keywords, phrases, symbols, and other elements, referred to as tokens. In the process of tokenization, some characters such as punctuation marks may be discarded. It is important for a tokenizer to generate meaningful results, as the output of this step becomes the input for subsequent processing steps such as parsing and text mining in the pipeline. In our study, we expect a tokenizer to segment a text into tokens such as words, emojis, and other special characters.

4.1 Task Setup

While tokenizing a sentence, or a tweet with emojis, in particular, we focus on the position and type of emojis presented earlier in Section 3. An emoji can accompany a word with both leading and trailing spaces, or it can be attached to words without any separating whitespace. We typically expect a tokenizer to distinguish an emoji from a word even in the absence of a space delimiter if it appears to constitute a separate concept. The same principle should be followed for emoji clusters, i.e., if multiple emojis occur in a sequence such as "🎂🥂🎈", they are expected to be recognized as individual tokens.

Another aspect of successful tokenization is adequately handling emoji skin tone modifiers. As emojis can have five different skin tone modifiers, we ensure that our test data contains the same number of tweets from all skin tones. An ideal tokenizer should not split skin tone emoji into two individual characters. For example, the *Waving Hand Light Skin Tone* 👋🏻 emoji should not be split into a regular *Waving Hand* emoji 👋 and a tone modifier 🏻.

We also test the abilities of tools in terms of handling ZWJ emoji sequences. We randomly pick a small set of tweets containing ZWJ sequences for this purpose. For example, an ideal tokenizer should not split up a Family Emoji as four individual emojis such as Man, Woman, Girl, Boy, as the emoji is meant to be rendered as a single one.

Note that some tokenizers discard punctuation during the tokenization process, while others retain them as tokens. For example, *Gensim* removes all punctuation, including all emojis. Furthermore, the *NLTK Tweet Tokenizer* does not split up a hashtag as "#" followed by a word, but rather keeps it intact, as hashtags usually convey meaningful information in tweets. Thus, to generalize the tokenization process across tools, we apply certain post-processing techniques before comparing the list of tokens with

[1] `http://emojitracker.com/`
[2] `https://github.com/abushoeb/emotag`

	Task - Tokenization					
Tools	**SE**	**ME**	**STE**	**BMP**	**NB**	**ZWJ**
Gensim	0	0	0	0	0	0
NLTK	70	0	68	70	80	70
NLTK-TT	100	100	0	100	100	0
PyNLPl	90	0	68	60	80	70
SpaCy	100	100	0	100	100	0
SpaCyMoji	100	100	92	100	100	10
Stanza	80	10	70	80	100	40
TextBlob	70	0	68	70	80	70

Table 2: Tokenization accuracy (%) of tools for different test set subsets. SE: single emoji, ME: multiple, STE: skin tone emojis, BMP: Basic Multilingual Plane, NB: Non-BMP, ZWJ: zero width joiner emojis.

the expected list. One such technique is to discard all punctuation from the list of tokens, while for #hashtag occurrences, we treat both "hashtag" and "#hashtag" as valid options.

Tools. In total, we consider 8 libraries for our experiments. These are the regular English tokenizer of the Natural Language Toolkit (NLTK) by Bird et al. (2009), the NLTK Tweet Tokenizer (i.e., its Twitter-aware tokenizer), the Stanford NLP Group's Stanza (formerly known as StanfordNLP) (Qi et al., 2020), SpaCy and SpaCyMoji, PyNLPl (the Python library for Natural Language Processing, pronounced as *pineapple*), Gensim (Řehůřek and Sojka, 2010), TextBlob, and AllenNLP (Gardner et al., 2018).[3]

4.2 Results

Table 2 presents the results of tokenizing the given case-specific test data, based on an overall set of 100 input texts. We partitioned this test data with regard to different cases of emoji use for a more fine-grained analysis.

For single emoji (SE), intended to be the simplest case, where each input cannot contain more than one emoji, we observe that most tools except for *Gensim* obtain acceptable results. Since *Gensim* discards emoji characters, it also fails all other test cases. In contrast, both *SpaCy* and *SpaCyMoji* achieve 100% accuracy. Other tools may fail to segment off emojis that have been attached to words without whitespace.

The multiple emojis (ME) case considers inputs with more than a single emoji, including clusters of emojis. Some tools, such as *NLTK* and *PyNLPl*, failed for this part despite having done well on single emoji utterances. Apart from separating off emojis from words, tools here differ mostly based on whether they split up groups of emojis.

For skin tone emojis, there are 50 test cases with skin tones. Note that these can have single or multiple emojis, but it is ensured that they bear at least one skin tone emoji. In some cases, the problems are the same as for regular emojis, e.g., splitting off emojis from words. However, some tools generally split off skin tone modifiers from the emojis they are intended to modify. *Stanza* only breaks a color tone emoji into the base emojis and tone modifiers when it is concatenated with text. Otherwise it can handle a skin tone emoji without splitting it. *SpaCyMoji* obtains a near-perfect result but still does not manage to preserve all skintone emojis.

The next test is designed to assess Basic Multilingual Plane (BMP) and non-BMP emojis, respectively. For each of these cases, a distinct set of 10 tweets was used to assess the performance. Interestingly, non-BMP emojis appear to be better-supported, presumably because they include the most popular emojis.

Finally, we consider emojis with zero width joiners (ZJW), where each tweet contains no more than two emojis with at least one ZWJ emoji. The tools that fail in this case, such as *NLTK-TT*, instead of preserving a ZJW emoji such as 👫, produce multiple separate tokens, including the Unicode zero-width joiners as individual tokens, e.g., 👨, U+200D, 👩, U+200D, 👧, U+200D, and 👦. In fact, none of the tools could achieve 100% accuracy across all ZWJ emojis. This is because they may fail when a regular emoji and a ZWJ one appear together. For example, one of the inputs contains the emojis 🔒👫, which *NLTK* treats as a single token, although it successfully handles other ZWJ emojis when they are space-separated. In contrast, *NLTK-TT* appears to be the best option for dealing with emoji clusters, but when it comes to ZWJ emojis, it separates all emojis and joiners.

5 Part-of-Speech Tagging

Part-of-Speech (POS) tagging is the process of assigning each token a label that reflects its word class. This may be with respect to traditional parts of speech, such as noun, verb, adjective, etc., or using a more fine-grained inventory of classes.

[3] We rely on Python 3.8 along with the latest version of all tools (Gensim 3.8.3, NLTK 3.4.5, PyNLPl 1.2.9, SpaCy 2.2.4, SpaCyMoji 2.0.0, Stanza 1.1.1, TextBlob 0.15.3) available until November 2020.

Tools	*Task - Parts-of-Speech (POS) Tagging*						
	Noun 26%	**Adjective 22%**	**Verb ~17.3%**	**Adverb ~17.3%**	**Punctuation ~17.3%**	**Average 100%**	**Modified Tokenizer**
NLTK	100.0	0	0	0	0	26.1	26.1
NLTK-TT	83.3	100	100	0	0	60.9	60.9
SpaCy	66.7	0	100	0	0	34.8	34.8
SpaCyMoji	66.7	0	100	0	0	34.8	34.8
Stanza	83.3	20	100	25	0	47.8	↑ 52.2
TextBlob	83.3	20	100	0	0	43.5	↑ 60.9

Table 3: The percentage of success of tools at labeling emojis with different parts-of-speech. The last column reports the average percentage of success when a modified tokenizer is used.

Tools	Tweets	Target Emoji	Expected POS	Default Tokenizer	Modified Tokenizer
Stanza	She kept her 🐕 dog but had to sell her 🐱....	🐕	Noun	🐕 (ADJ) 🐱.... (.)	🐕 (ADJ) 🐱 (NN)
Stanza	**MODIFIED:** She kept her 🐕 ~~dog~~ but had to sell her 🐱....	🐕	Noun	🐕 (Noun) 🐱.... (.)	🐕 (Noun) 🐱 (Noun)
Stanza	I MADE A PICTURE ‼️ ‼️ What do you think ❓✨😱	‼️	Punctuation	‼️ ‼️ (.) ❓✨ (NNP)	‼️ (NN) ‼️ (.)
TextBlob	I MADE A PICTURE ‼️ ‼️ What do you think ❓✨😱	❓	Punctuation	❓✨😱 (NNS)	❓ (NNP) ✨ (NNP) 😱 (NN)
TextBlob	Yes, she is😎and I like it	😎	Adjective	is😎and (Verb)	😎 (Adj)

Table 4: Examples of tweets in which an emoji assumes the role of different parts-of-speech. The last column reports how the tagging accuracy can be improved by utilizing a unified tweet-aware tokenizer across all tools.

5.1 Task Setup

To understand how different POS taggers handle emojis in a sentence, we evaluate all tools for a subset of inputs covering the majority of emoji scenarios mentioned in Section 3.

For evaluation, we compiled a set of 23 real tweets, in which emojis are used as different parts-of-speech, namely as nouns, adjectives, verbs, adverbs, or as punctuation. We mapped the original part-of-speech tags to these coarse-grained categories and then checked for correctness with regard to human annotations obtained for our tweets. Only the part-of-speech tags assigned to the emojis were considered, while the tagging of all other non-emoji tokens was deemed irrelevant for the purposes of this experiment. Note also that this test suite is limited to clear-cut cases of emojis used within sentences and we do not claim that every potential use of an emoji has an obvious well-defined part-of-speech tag.

Tools. For this task, we evaluated all tools except *Gensim* and *PyNLPl*, as they do not directly offer any POS tagging functionality. Since tokenization is a prerequisite for POS tagging, a tool is likely to fail to correctly tag a word or emoji if the emoji is not properly tokenized in the preceding step. However, for a more extensive evaluation, we considered two setups. First, we conducted the POS tagging experiment based on the output of the integrated tokenizer of the respective tool. Thus, if a tool was unable to tokenize "Emojis😀are" as three separate tokens "Emojis", "😀", and "are", we still proceeded with the task treating it as one token for the respective tool's POS tagger. Subsequently, we conducted the POS tagging experiment while considering a unified ground truth tokenization as input for all tools. For example, in the case of "Emojis😀are", the tagger could expect to receive them as separate tokens "Emojis", "😀", and "are".

5.2 Results

Table 3 reports the results of our part-of-speech tagging experiments. The final two columns summarize the results with the original tokenizer and the modified tokenizer. None of the tools in our experiment could handle the case of emojis acting as adverbs or as punctuation. For instance, "My Credit Score Went 🔼 7 Points 🙏" is one such

example where the *Upwards Button* emoji 🔼 assumes an adverbial role, which none of the taggers recognize, despite the emoji being space-delimited.

Similarly, occurrences of the question mark emoji ❓ or double exclamation mark emoji ‼️ used as punctuation are labeled as nouns by all considered tools.

Interestingly, we obtained a 100% success rate for handling verb emojis, except with *NLTK*. Although the latter is the only tool that passes all test cases for noun emojis, it fails for all other cases. Overall, *NLTK-TT* and *Stanza* obtain the highest success rates as reported in the penultimate column of the table.

When considering the harmonized ground truth tokenization, as reported in the final column of Table 3, the results for *TextBlob* are boosted significantly and for *Stanza* a more modest gain is observed. *TextBlob* and *Stanza* for instance may fail when emojis are not separated by whitespace from regular words (e.g., "love❤️") or from another emoji (e.g., "🤔🤓"). Rectifying the tokenization in such cases improves the results of both tools.

The first example in Table 4 shows the interesting phenomenon of redundancy causing incorrect predictions. In this tweet, both the dog emoji 🐕 and the cat emoji 😺 are expected to be tagged as nouns, but Stanza assumes the former to be an adjective due to the additional presence of the regular word "dog". To examine this further, we also considered several modifications of the original tweet. First, we considered the tweet without the additional word "dog" word after the dog emoji 🐕, in which case Stanza can easily identify it as a noun. This is reported in the second row of Table 4. We also tried replacing the dog emoji with the word "dog" to see if *Stanza* can cope with erroneous word reduplication, and it turned out that Stanza could correctly identify both occurrences as nouns. Finally, we considered replacing the word "dog" with another 🐕 emoji. In this case, the tool marked the first 🐕 as a noun and the second 🐕 as punctuation.

6 Dependency Parsing

In dependency grammar, the syntactic structure of a sentence is described as a tree capturing relationships between *head* words and *dependent* words. Given that emojis can have different grammatical roles within a sentence, we thus assessed to what extent popular dependency parsers are affected by the presence of emojis in the input.

6.1 Task Setup

We rely on the English Web Treebank (EWT), one of the around 200 treebanks in the Universal Dependency (UD) collection[4], which seeks to define a consistent annotation of grammar (including parts of speech, morphological features, and syntactic dependencies) across over 100 languages. The English Web Treebank UD corpus provides gold standard Universal Dependency annotations , built over the source material of the English Web Treebank (Bies et al., 2012). We randomly pick a set of sentences from EWT and then replace certain obvious words with matching emojis in both the plain text sentences and their corresponding dependency trees to obtain a ground truth set. Examples of such word–emoji replacements include fire 🔥, death 💀, salad 🥗, etc. To further examine the robustness of the tools, we also incorporate multi-emojis, skin tone emojis, and ZWJ emojis in the input. For instance, one of the EWT sentences includes "Chicken salad salad is great too." for which we embed the 🥗 emoji as "Chicken 🥗🥗 is great too.". The purpose of this approach is to assess how well a dependency parser can handle such forms of emoji use. Again, our test suite is limited to clear-cut instances and we do not make the assumption that any possible emoji use will have an unambiguous well-defined ground truth annotation.

Tools. Not all of the previously considered tools provide their own dependency parser. For this evaluation, we thus considered only Stanford's *CoreNLP*, *SpaCy*, and *Stanza*.

6.2 Results

Table 5 reports both Labeled Attachment Score (LAS) and Unlabeled Attachment Score (UAS) results, where the latter consider just the location of the edge, i.e., just the structure of the tree, while the former as well mandate that the edge labels be identified correctly. The first two columns report the average attachment score based on the entire dependency tree for the given set of inputs. The next two columns (i.e. Single Emoji Sub-tree) consider only the parent and child nodes of emojis in the tree, i.e., an emoji-centered sub-tree (or forest). Finally, the last two columns report LAS and UAS

[4]Version 2.7 https://universaldependencies.org/#download

	Single Emoji		*Single Emoji Sub-tree*		*Multi Emoji*	
Tools	**LAS**	**UAS**	**LAS**	**UAS**	**LAS**	**UAS**
CoreNLP	0.729	0.766	0.631	0.715	0.625	0.655
SpaCy	0.359	0.459	0.211	0.256	0.311	0.401
Stanza	0.821	0.836	0.758	0.796	0.725	0.733

Table 5: Labeled Attachment Score (LAS) and Unlabeled Attachment Score (UAS) of Dependency Parser based on the English Web Treebanks (EWT)

for complex emojis including multi-emojis, skin tone emojis, and ZWJ emojis in the given testcases.

The results show a clear degradation of both the tree structure (UAS) and the dependency labels (LAS) when it comes to tackling edges in the graph connecting other tokens to emojis. This becomes more evident with the presence of complex emojis in the tree. In general, Stanford *CoreNLP* and *Stanza* appear to be more robust than *SpaCy*.

7 Sentiment Analysis

Although the word "emoji" is not etymologically related to the word "emotion", several studies show how emojis can help to express emotions (Shoeb and de Melo, 2020) and sentiment in textual communication (Novak et al., 2015). Keeping this in mind, we further assessed how well NLP tools fare at the task of predicting the sentiment polarity of a text harboring emojis. Table 6 shows examples of texts with different emojis. While the text alone may be ambiguous with respect to its sentiment polarity, the emoji appears to eliminate much of the ambiguity if it is appended to the end of the text. The goal of this endeavor is to examine if the sentiment polarity is predicted correctly when a high-intensity emoji is incorporated into a neutral sentence.

7.1 Task Setup

For this task, we leverage a set of custom sentences and tweets from the Sentiment140 dataset (Go et al., 2009). We considered a set of texts with neutral or ambiguous sentiment. The sentiment label was verified by multiple tools before considering them in our experiment. A sentence as well as a tweet were only considered when their sentiment labels were consistent across multiple tools. Although the specific sentiment score may vary from one tool to another, we ensured that the sentiment label remained consistent. Each example was then modified with both positive and negative emojis appended to the end, giving us the opportunity to observe whether the predicted polarity of the original sentence changes in accordance with the polarity of the emojis. For example, *I'll explain it later* is a neutral sentence that is modified either with a positive emoji 😊 or with a negative one such as 👎. We use different sets of positive and negative emojis to modify the sentiment of the text, covering a broad spectrum of the sentiment polarity of emojis. The sentiment of emojis was determined based on the data by Novak et al. (2015).

Tools. Although many tools could be trained on a labeled set of tweets, we sought to assess pre-existing systems as they are often used out-of-the-box without additional training or fine-tuning. Hence, this study considers *NLTK* and *TextBlob*, as they can readily be used on the fly without requiring new labeled data. Note that *TextBlob*'s sentiment module contains two sentiment analyzers, PatternAnalyzer and NaiveBayesAnalyzer, the latter trained on movie reviews. For *NLTK*, we use VADER (Valence Aware Dictionary and sEntiment Reasoner), a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiment as expressed in social media. Additionally, we evaluate the standalone VADER library directly as it is meant to support emoji sentiment (Hutto and Gilbert, 2014).

7.2 Results

The results are given in Table 7. In the sentiment prediction task for a given tweet with emojis, neither NLTK nor the TextBlob models appear to be able to consider the emojis as part of their sentiment polarity prediction. Only the stand-alone VADER library is able to discern any difference when positive or negative emojis are provided with the sentence, as reflected in the final row of Table 7. The discrepancies between NLTK's VADER component and the stand-alone VADER stem from differences in the lexicon used by the tools. The stand-alone VADER includes a dedicated emoji lexicon that is omitted in the NLTK version. Some studies (Jain et al., 2019) show that an emoji can moderate the sentiment of a given tweet if the sentiment of an emoji is considered during training.

Sentences	*Modifiers*		*Sentiment Predictions*		
	+ve Emoji	–ve Emoji	Only Text	Text with +ve Emoji	Text with –ve Emoji
They decided to release it	😍	👎			
Let's go for it	💃	😩	Expected	Expected	Expected
I'll explain it later	😊	👎	Neutral	+ve	–ve
There is a book on the desk	😁	😒	Observed	Observed	Observed
This is the end	😍	😩	Neutral	Neutral	Neutral
My passport is expired by little over a month	😁	😒	–ve	No	No
It's good that they have a direct flight now	😍	😩	+ve	Change	Change

Table 6: Example sentences with relatively high polarity emojis that could moderate the overall sentiment of the given sentences – NLP tools, in general, fail to capture the combined (text+emoji) sentiment

Tools	Model	NT	*Emojis* +ve	-ve
NLTK	VADER	100.0	0.0	0.0
TextBlob	PatternAnalyzer	100.0	0.0	0.0
TextBlob	NaiveBayesAnalyzer	100.0	0.0	0.0
VADER	VADER	100.0	57.1	50.0

Table 7: Accuracy (in %) of different tools at predicting sentiment scores of neutral text alone (NT) or neutral text along with positive (+ve) or negative (-ve) emojis

Emoji	Nearest Neighbour Emojis
Clapping Hands (Regular) 👏	👌 👍 🙌 👊 👐
Clapping Hands (Light) 👏🏻	🙌🏻 👌🏻 👍🏻 💪🏻 👊🏻
Clapping Hands (Medium Light) 👏🏼	🙌🏼 👌🏼 👍🏼 [👏🏽] 👊🏼
Clapping Hands (Medium) 👏🏽	🙌🏽 👌🏽 👍🏽 [👏🏽] 💪🏽
Clapping Hands (Medium Dark) 👏🏾	🙌🏾 👌🏾 👍🏾 💪🏾 ✊🏾
Clapping Hands (Dark) 👏🏿	🙌🏿 👌🏿 👍🏿 💪🏿 👊🏿
ZWJ Family (Man, Woman, Girl, Boy) 👨‍👩‍👧‍👦	👨‍👩‍👧‍👦 👨‍👩‍👧‍👦 👨‍👩‍👧‍👦 👨‍👩‍👧‍👦 👨‍👩‍👧‍👦

Table 8: Nearest neighbour emojis for the Clapping Hands and Family emojis. All nearest neighbours follow mostly the same color tone of the respective emojis except some indicated with [ ].

Clearly, systems trained on emoji-bearing data can learn to consider them during prediction if their tokenization is handled properly and they are not discarded during preprocessing. However, given the importance of emojis in conveying sentiment, it appears that most out-of-the-box tools ought to consider emojis as well.

8 Discussion

Overall, based on Table 9, we can see that none of the considered tools perfectly handles all evaluated tasks with emojis. Indeed, many text preprocessing pipelines, especially deep learning ones with a limited vocabulary, routinely discard emojis along with punctuation characters as non-standard characters. Gensim by default follows this common approach, which is likely suboptimal for emojis. *NLTK-TT* as well as *Stanza* help keep track of hashtags as they retain them with the "#" sign intact, whereas other tools split them up as two individual tokens or remove the "#". *NLTK*, *Stanza*, and *TextBlob* fail to tokenize emojis if emojis are tied up with other words, while *SpaCy*, *SpaCyMoji*, and *NLTK-TT* handle such cases. Note that accurate tokenization, e.g., splitting off emojis attached to words, can also be a prerequisite for many downstream tasks, such as enabling higher-quality text classification and information retrieval.

For POS tagging, somewhat surprisingly, almost all tools did well with verbs, while they all struggled with punctuation emojis as well as adverbs. The results for adjectives were as well quite mixed. Overall, *NLTK-TT* and *TextBlob* achieved the highest success rate for POS tagging, although both still struggle with adverbs and punctuation, which can also lead to adverse effects in downstream tasks such as syntactic parsing. Moreover, *TextBlob* requires the use of a modified tokenizer. For dependency parsing, we found Stanford CoreNLP and Stanza to be the most robust in correctly assessing emojis. SpaCy, in contrast, does not appear to generalize well enough to lexical items such as emojis that may be lacking in the training data. In general, there is a need for dependency parsers to be trained on more diverse data.

Thus, in practice one may wish to consider a mix-and-match approach, using a tokenizer from one library, a tagger from another, and a dependency parser from yet another library.

In our POS tagging and dependency parsing evaluations, we sought to study clear-cut cases to observe whether tools have basic support for emojis. Further discussion is necessary on recommended annotation schemes for more diverse forms of emoji

Tools	SE	GE	STE	BMP	ZWJ
Gensim	✗	✗	✗	✗	✗
NLTK PyNLPl Stanza TextBlob	✓	✗	✓	✓	✓
AllenNLP NLTK-TT SpaCy	✓	✓	✗	✓	✗
SpaCyMoji	✓	✓	✓	✓	✗

Table 9: An overview of popular text processing NLP tools and their emoji support. SE: single emoji, GE: groups of emojis, STE: skin tone emojis, BMP: Basic Multilingual Plane, ZWJ: zero width joiner emojis.

use for which the ground truth may not be as obvious. Some researchers argue that the default tagging of emoji should be as adverbials, interjections, or punctuation (Grosz et al., 2021). Similarly, emojis are syntactically comparable to free adjuncts, which constrains the set of valid parse trees. Hence, further work is necessary to devise broader-coverage benchmarks for the tasks considered in our study.

Semantic Associations. Finally, we also inspected semantic associations for particular kinds of emojis. We considered a 300-dimensional word2vec SGNS model trained on the EmoTag (Shoeb et al., 2019) dataset, and generated a set of nearest neighbours for selected target emojis.

Table 8 reports the nearest emoji neighbours for different skin tone variants of the *Clapping Hand* emoji. Most of the top 5 neighbours for each emoji bear the same skin tone color except one each for *Medium Light* and *Medium* tone emojis reported in Rows 4 and 5, respectively. We conjecture that speakers who use skin tone modifiers frequently also use additional emojis that support such modification and that they naturally tend to use the respective modifier fairly consistently.

The last row of the same table shows the nearest neighbours for a ZWJ family emoji. All of the nearest neighbours of this ZWJ emoji contain a ZWJ sequence as well, suggesting that they occur in similar contexts.

9 Conclusion

Emojis have become an integral part of modern interpersonal communication and text encountered in chat messages, social media, or emails is often laden with emojis. Hence, it is important to endow NLP tools with emoji support not only to obtain a deeper understanding of this wealth of data but also to properly preserve and process them correctly.

In this study, we assessed how well prominent NLP tools cope with text containing emoji characters. To this end, we evaluated a set of tools on three different tasks across a range of challenging test sets capturing particular phenomena and encodings. Our study demonstrates that there are notable shortcomings in widely used NLP tools. Although many tools are partially capable of operating on emojis, none of them proved fully equipped to tackle the full set of aspects considered in our study. Hence, special care needs to be taken when developing applications that may encounter emojis.

References

Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012. English web treebank.

Volodymyr Bilyk. 2020. Natural language processing tools and libraries 2020.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly, Beijing.

Jeremy Burge. 2017. 5 billion emojis sent daily on messenger.

Ambika Choudhury. 2019. Top 10 python NLP libraries for 2019.

Neil Cohn, Jan Engelen, and Joost Schilperoord. 2019. The grammar of emoji? constraints on communicative pictorial sequencing. *Cognitive research: principles and implications*, 4(1):33.

Xin Dong and Gerard de Melo. 2018. Cross-lingual propagation for deep sentiment analysis. In *Proceedings of AAAI 2018*. AAAI Press.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625, Copenhagen, Denmark. Association for Computational Linguistics.

Thomas B. Fitzpatrick. 1975. Fitzpatrick scale - wikipedia.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages

1–6, Melbourne, Australia. Association for Computational Linguistics.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009.

Patrick G Grosz, Elsi Kaiser, and Francesco Pierini. 2021. Discourse anaphoricity and first-person indexicality in emoji resolution.

Clayton Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 8.

Akansha Jain, Ishita Aggarwal, and Ankit Singh. 2019. ParallelDots at SemEval-2019 task 3: Domain adaptation with feature embeddings for contextual emotion analysis. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 185–189, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Linda K. Kaye, Stephanie A. Malone, and Helen J. Wall. 2017. Emojis: Insights, affordances, and possibilities for psychological science. *Trends in Cognitive Sciences*, 21(2):66 – 68.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. 2014. A dependency parser for tweets. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1001–1012, Doha, Qatar. Association for Computational Linguistics.

Dominik Kozaczko. 2018. 8 best python natural language processing (NLP) libraries.

Gretchen McCulloch. 2019. *Because Internet: Understanding the New Rules of Language*. Penguin Publishing Group.

Petra Kralj Novak, Jasmina Smailović, Borut Sluban, and Igor Mozetič. 2015. Sentiment of emojis. *Plos One*, 10(12).

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–390, Atlanta, Georgia. Association for Computational Linguistics.

Thomas Proisl. 2018. Someweta: A part-of-speech tagger for german social media and web texts. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

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*, pages 45–50, Valletta, Malta. ELRA.

Alexander Robertson, Walid Magdy, and Sharon Goldwater. 2020. Emoji skin tone modifiers: Analyzing variation in usage on social media. *Trans. Soc. Comput.*, 3(2).

Abu Awal Md Shoeb and Gerard de Melo. 2020. Emotag1200: Understanding the association between emojis and emotions. In *Proceedings of EMNLP 2020*.

Abu Awal Md Shoeb, Shahab Raji, and Gerard de Melo. 2019. EmoTag – Towards an emotion-based analysis of emojis. In *Proceedings of RANLP 2019*, pages 1094–1103.

T. Weerasooriya, N. Perera, and S. R. Liyanage. 2016. A method to extract essential keywords from a tweet using NLP tools. In *2016 Sixteenth International Conference on Advances in ICT for Emerging Regions (ICTer)*, pages 29–34.

Rachel Wolff. 2020. 10 natural language processing tools – saas & open-source.