Trash or Treasure: How to Utilize Emojis in Social Media Sentiment Classification

DURF Research Report

Bale (Bingsen) Chen
Researcher
bale.chen@nyu.edu

Mathieu Laurière
Faculty Mentor
mathieu.lauriere@nyu.edu

Abstract

Sentiment analysis has become a popular topic in the field of Natural Language Processing these years. It is helpful to monitor social trends and collective emotions by analyzing social media data with sentiment analysis models. Besides, emojis have also been prevalent in recent years on social media, which is an efficient way to embed emotional cues in written text. However, most studies treat emojis as unstructured and useless tokens to be cleaned out in Social Media Sentiment Analysis. Therefore, this study probes into possible ways to qualify and include emojis in the sentiment analysis process. Also, we investigated the emoji-compatibility of currently trending models like BERT, RoBERTa, BERTweet, etc. 5 methods of incorporating emojis were proposed and experimented with 6 emoji-supportive encoder models. The study fills in the literature gap about encoding emojis with BERT-based encoders, provides strong evidence that emojis significantly improve the models' performance, and proposes a feasible and promising future methodology for social media sentiment analysis.

Keywords Social Media Sentiment Analysis · Emojis · Encoder · Data Preprocessing

1 Introduction

As social media has become an essential part of people's lives, the content that people share on the Internet is highly valuable to many parties. Many modern natural language processing (NLP) techniques were experimented and deployed to understand the general public's social media posts. Sentiment Analysis, is one of the most popular and critical NLP topics that focuses on analyzing opinions, sentiments, emotions, or attitudes toward entities in written texts computationally [15]. Social media sentiment analysis (SMSA) is thus a field of understanding and building representations for the sentiments expressed in short social media posts. One of the usual approaches that simplifies and structures such representation learning task is sentiment classification [3], where each piece of text is classified as connoting positive, neutral, or negative sentiment. This research also takes on sentiment classification task to experiment different sentiment analysis methods.

Emojis, invented by Shegetaka Kurita in 1990s, are graphic symbols with predefined names, IDs, and code (Unicode), which include not only representations of facial expressions (e.g. 😀), abstract concepts (e.g. 🛂), and emotions/feelings (e.g. ❤️), but also animals (e.g. 🐕), plants (e.g. 🌲) activities (e.g. ⚽), gestures/body parts (e.g. 👍), and objects (e.g. 🕶️) [28]. These in-text graphics have gained ground in social media. According to Emojipedia's statistics in 2021 [1], a famous emoji reference site, over one-fifth of the tweets now contains emojis (21.54%), while over half of the comments on Instagram include emojis [8]. Emojis are handy and concise ways to express emotions and convey meanings, which gives them great popularity.

[1]https://emojipedia.org/stats/

However ubiquitous emojis are in network communications, they are not favored by the research field of NLP and SMSA. In the stage of preprocessing data, emojis are usually removed alongside with other unstructured information like URLs, stop words, unique characters, and pictures [4]. While some researchers have started to study the potential of including emojis in SMSA in recent years, it remains a niche approach and awaits further research. Our research aims to join the scholarly conversation by exploring different methods of incorporating emojis in SMSA to improve the performance of current models and examining the emoji-compatibility of existing models.

2 Related Works

2.1 Sentiment Analysis

Sentiment analysis is usually conducted by estimating the sentiment polarity of some written texts. As there are different types of texts, usually sentiment analysis is categorized into document-level, sentence-level, and aspect-level [20]. On document-level, the task is to estimate the overall sentiment of multiple sentences. It disregards the individuality of each sentence but summarizes the sentiment of the whole document with a score. For sentence-level, models usually gives each individual sentence a sentiment label or score. Lastly, on aspect-level, the sentiment or attitude towards a specific target is identified, which usually involves two steps – extracting the aspect and estimating sentiment.

The field has surged since 2004 and continues to bloom [22]. Before deep learning and neural network have become popular, researchers mainly used lexicon-based or traditional machine learning methods. Then, deep neural networks and recurrent neural networks were introduced and gained ground. In recent years, importing and fine-tuning large pretrained models such as BERT and RoBERTa, has become an unavoidable trend.

Lexicon-based Methods SentiWordNet [9] is one of the most used document-level lexicon. Based on WordNet database, it takes into account every word in a document and computes the sentiment score by aggregating the positivity and negativity of each term. VADER [14], developed by Hutto and Gilbert, is a sentence-level lexicon that computes sentiment polarity of sentence based on both sentiment intensity of each word and grammatical features of a sentence.

Word Embeddings and Machine learning Speaking of machine learning models, text is not a valid input. Usually, texts are vectorized before being fed into a model. Google's `word2vec` [21] was a breakthrough of representation learning for textual data. It is based on continuous bag-of-words and skip-gram methods to learn the vector representation of a word by its context. Pennington et al. developed GloVe word embeddings [27] that learns word representations based on co-occurrences of different words, which performs well on identifying word analogies and similarity.

With those predefined embeddings, we can then transform textual data into numerical data. Word vectors can be passed into machine learning models such as Random Forest and Support Vector Machine to output the sentiment score.

Neural Networks Due to the sequential nature of human language, recurrent neural networks are more favored in sentiment analysis than other models. With a many-to-one architecture, the recurrent neural networks can take in the word embeddings in a sequence and output the last hidden state. The last hidden state can be then passed into a feed-forward fully-connected layer to generate a final sentiment score.

Large Pretrained Models In recent years, with the development of Transformer models, many previous state-of-the-art models were superseded by large pretrained transformer models. Developers can easily load and deploy the pretrained models thanks to the Hugging Face `transformers` package [2]. Bidirectional Encoder Representations for Transformers (BERT), developed in 2019, significantly pushed the state of the art of multiple benchmark datasets and NLP tasks. We now see more variations like RoBERTa [17], BERTweet [25], DeBERTa [13], etc. BERT-based models as mentioned above are encoders that learn excellent word representations for many NLP tasks such as sentiment analysis.

Twitter, as one of the biggest social media platforms where people can post short microblogs (tweets), is a perfect resource for SMSA. For its popularity across the globe and easy accessibility, the sentiment level of tweets can well represent the sentiment of the mass, and thus it is a great place to monitor social sentiment or

[2]https://huggingface.co/models

trends. In the field of TSC, the state of the art is held by pretrained BERTweet [25] model as they achieved a new highest F1-score and accuracy on both SemEval-17 Task4A [29] and SemEval-18 Task3 [32]. Our SMSA is also based on Twitter data, which means the datasets are real tweets from Twitter users.

2.2 Emojis and SMSA

Although the mainstream approaches in SMSA has not taken emojis as a resource, there have been studies who use emojis in SMSA in a variety of ways. Rather than using emojis to analyze tweet sentiment, researchers developed a SMSA task called emoji prediction to evaluate SMSA models [1]. Given a text message that originally includes an emoji, the emoji prediction task requires the model to predict predict the emoji based merely on textual content of the message. Similarly, DeepMoji [10] model was pretrained with the task of emoji prediction as well. The model learned any-domain representations for written text and had achieved state-of-the-art performance in many sentiment analysis tasks back in 2017. The ELSA [6] model also incorporate the emoji prediction task to better learn sentence representations.

In recent years, we started to see some researches on emoji-powered sentiment analysis. In 2015, Novak et al. created a sentiment lexicon that assigned each emoji with a position, positivity, neurality, negativity, and sentiment score based on its occurrences [26]. Then, vectorized embeddings emerges such as `emoji2vec` [7] and "What does emoji mean" [2]. In the following years, a number of researchers starts to utilize emojis in SMSA studies by training their own embedding models with their own annotated data [16] [18] [5]. Methodology-wise, Singh et al. investigated three ways of incorporating emojis: treating them as regular tokens, using `emoji2vec` embeddings, and replacing emojis with their textual description. They found that using textual description achieves a better performance in SMSA tasks. Liu et al. also experimented three ways of including emojis in SMSA: converting them to tag words, to the set of sentiment words translated from emojis manually, to emoji usage (1 for positive usage, 0 for neural usage, and -1 for negative usage) [16].

Few researches have looked into the coordination between the trending transformer encoders and the emojis, not to mention their combinations with different data preprocessing methods. Our research serves as an introductory investigation of leveraging emojis in Twitter sentiment classification in the era of transformer models.

3 Methodology

In this section, we will introduce the datasets, models structure, and design of our experiment. Our research questions are as follow:

- RQ1: Since current SA studies usually remove emojis from the textual data to keep the data clean, would emojis decrease or increase the model performance if being included?
- RQ2: How should we combine emojis and textual data as input in the SMSA models?
- RQ3: What current encoder models would perform better with the help of emojis in SMSA tasks?

The framework of our research is depicted in Fig. 1 and is presented in detail in the rest of this section.

3.1 Datasets

To better compare the results from our proposed methodologies and previous literature, we originally planed to use popular benchmark SMSA datasets. However, due to limited amount of interest in emoji-inclusive SMSA studies thus far, we only found 2 datasets that meet our needs. Here below is a non-exhaustive table of datasets we consulted (See Table 1).

After a round of data collection, only the emoji2vec dataset and SemEval-15 Task 11 dataset are ready to use. For emoji2vec dataset, the tweets are stored in a pickle file and splitted into train and test sets. There are in total 64599 tweets (51679 in the training set, 12920 in the test set). Each of the tweets is labeled as positive(1), neutral(0), or negative(-1). For SemEval-15 Task 11 dataset, there are 8000 tweets with figurative language in total with a float number label $\lambda \in [-5, 5]$. λ is the sentiment level ranging from very negative (-5) to very positive (5).

[3]"No" does not necessarily mean the original dataset does not have emojis. One possibility is that oftentimes datasets are kept in `*.csv` format, which does not support emojis, so that emojis are lost in the process of storing data. Or, the dataset collectors have already done the job of data cleaning, in which process the emojis are removed.

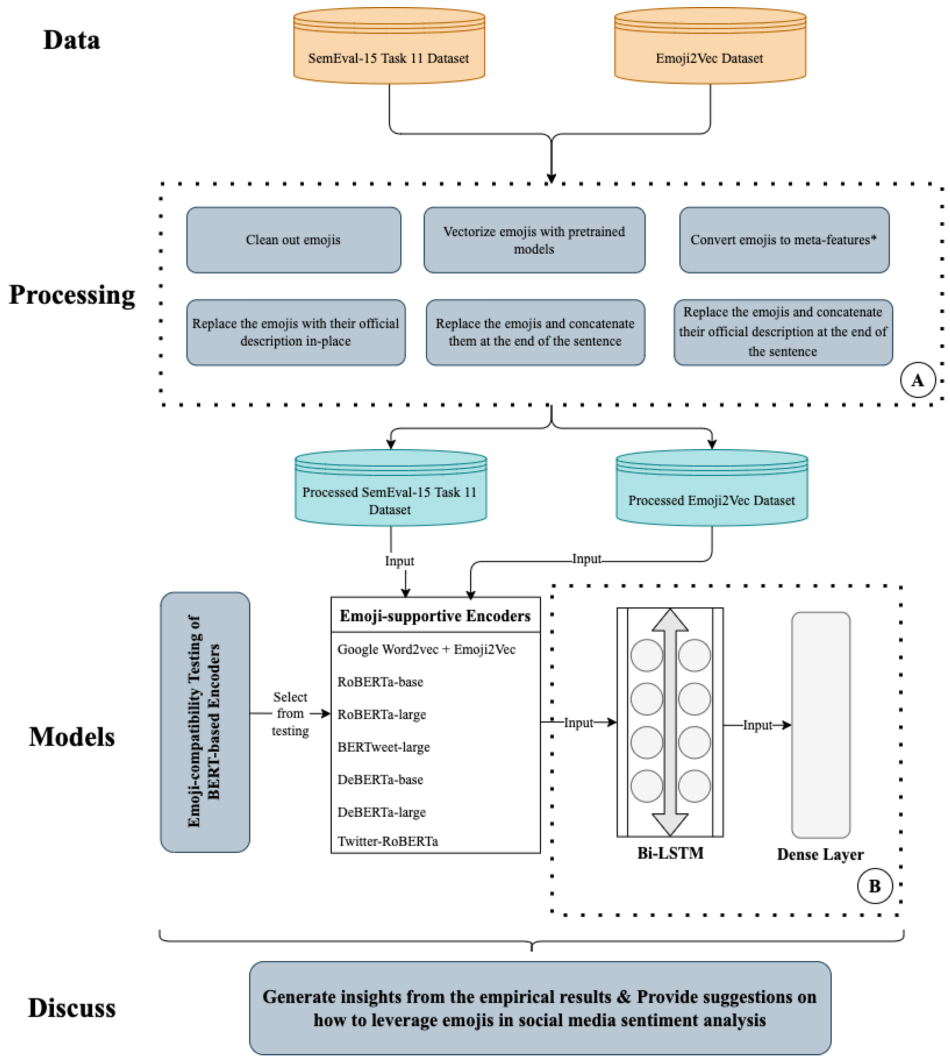

Data
SemEval-15 Task 11 Dataset
Emoji2Vec Dataset
Processing
Clean out emojis
Vectorize emojis with pretrained models
Convert emojis to meta-features*
Replace the emojis with their official description in-place
Replace the emojis and concatenate them at the end of the sentence
Replace the emojis and concatenate their official description at the end of the sentence
A
Processed SemEval-15 Task 11 Dataset
Processed Emoji2Vec Dataset
Input
Input
Models
Emoji-compatibility Testing of BERT-based Encoders
Select from testing
Emoji-supportive Encoders
Google Word2vec + Emoji2Vec
RoBERTa-base
RoBERTa-large
BERTweet-large
DeBERTa-base
DeBERTa-large
Twitter-RoBERTa
Input
Bi-LSTM
Input
Dense Layer
B
Discuss
Generate insights from the empirical results & Provide suggestions on how to leverage emojis in social media sentiment analysis

Figure 1: Research Framework Flowchart. Box A displays the 6 proposed methods to combine emojis and text in the input. Box B is the decoder that is trained through back-propagation. The training process is explained in Section 3.3.2

Name	Domain	Contains emojis? [3]
IMDb Reviews [19]	Movie Reviews	No
SemEval-17 Task 4 [29]	Generic	No
SemEval-16 Task 4 [23]	Generic	No
SemEval-13 Task 2 [24]	Generic	Perished*
Sentiment140 [12]	Generic	No
SemEval-15 Task 11 [11]	Figurative/Sarcasm	Yes
emoji2vec [7]	Generic	Yes
SentiStrength [31]	Generic	No

Table 1: Benchmark datasets consulted

* "Perished" means either the tweet has been deleted or the user account does not exist anymore so that we cannot access the tweet with Tweet IDs.

3.2 Model

We adopted the encoder-decoder architecture in our model. The encoder will embed words in a sentence into a sequence of vectors, which represents a reading and comprehending process of the sentence. The decoder then takes the sequence for word embeddings as input and outputs the predicted sentiment label. A diagram of our model can be found in Fig. 2.

Encoder We mainly selected the most popular Transformer-based encoders and `emoji2vec` pre-defined word embeddings [7] that is supplementary to the Google `word2vec`. The `emoji2vec` embeddings were defined and trained in the following way,

For each emoji i, we define a trainable vector of float numbers x_i and a vector of float numbers v_i that represents its official textual description. Also, we match each emoji with its official description that is expressed as a sequence of vectorized word embeddings $w_1, ..., w_n$ from Google `word2vec`,

$$v_i = \sum_{k=1}^{n} w_k$$

To train the vector representation x_i for emoji i, we generate emoji-description pairs that are either a true match (the emoji i matches the description j) or a false match (the emoji i does not match the description j, but is paired with a description of another emoji). Then, we train the vectors x_i's with loglikelihood loss,

$$\mathcal{L}(i, j, y_{ij}) = -\log(\sigma(y_{ij} x_i^\top v_j - (1 - y_{ij}) x_i^\top v_j))$$

where $y_{ij} = 1$ if emoji i matches description j and 0 otherwise.

For transformer encoders, the most important feature is the self-attention mechanism [33]. It allows the computation of each word embedding to attend on the context automatically through a relatively simple mathematical process that we now describe.

Let us denote by B the batch size, S the sequence length, and D the dimension of initial word vectors. Given the input $X \in \mathbb{R}^{B \times S \times D}$, and trainable weight matrices $W_Q, W_K \in \mathbb{R}^{D \times d_k}$ and $W_V \in \mathbb{R}^{D \times d_v}$, we compute the query (Q), key (K), and value (V) representations of the word sequences,

$$Q = XW_Q, \qquad K = XW_K, \qquad V = XW_V$$

Attention-based word embeddings (X_{att}) can then be calculated by the scaled dot product attention formula [33],

$$X_{att} = Attention(Q, K, V) = softmax\left(\frac{QK^\top}{\sqrt{d_k}}\right) V \in R^{B \times S \times d_v}$$

Decoder To compare the performance of different encoders and data processing methods, we hold the decoder part fixed across all experiments. To capture the sequential structure of the input and map the high-dimensional vectors to class labels (-1, 0, 1), we design the decoder to be a combination of Bi-LSTM layer and feed forward fully connected layer as shown in box B of Fig. 2. The Bi-LSTM layer consists of 2 LSTM models, where the 2 models scan the sequence in opposite directions. This architecture made the decoder layer capable of extracting more information from the sequence.

The box B in Fig. 1 is consistent with and elaborated in Fig. 2. It encloses the decoder part that is trained through back-propagation in the experiment while parts other than box B remain frozen.

3.3 Experiment

3.3.1 Emoji-compatibility Testing

To screen out the existing encoders that does not support emojis, we designed a emoji-compatibility testing experiment. In particular, supporting emojis means having unique tokens in the encoder's corresponding tokenizer, which enables the encoders to generate different embedding vectors for different emoji tokens. In contrast, the encoders that does not support emojis at all would tokenize all emojis as unknown tokens (e.g. `<UNK>`). Whether or not a encoder is compatible with emojis purely depends on the corpus used for pre-training. As consulting the entire corpora of different encoders is inefficient in terms of memory usage and data unavailability, we designed an experiment shown in steps below:

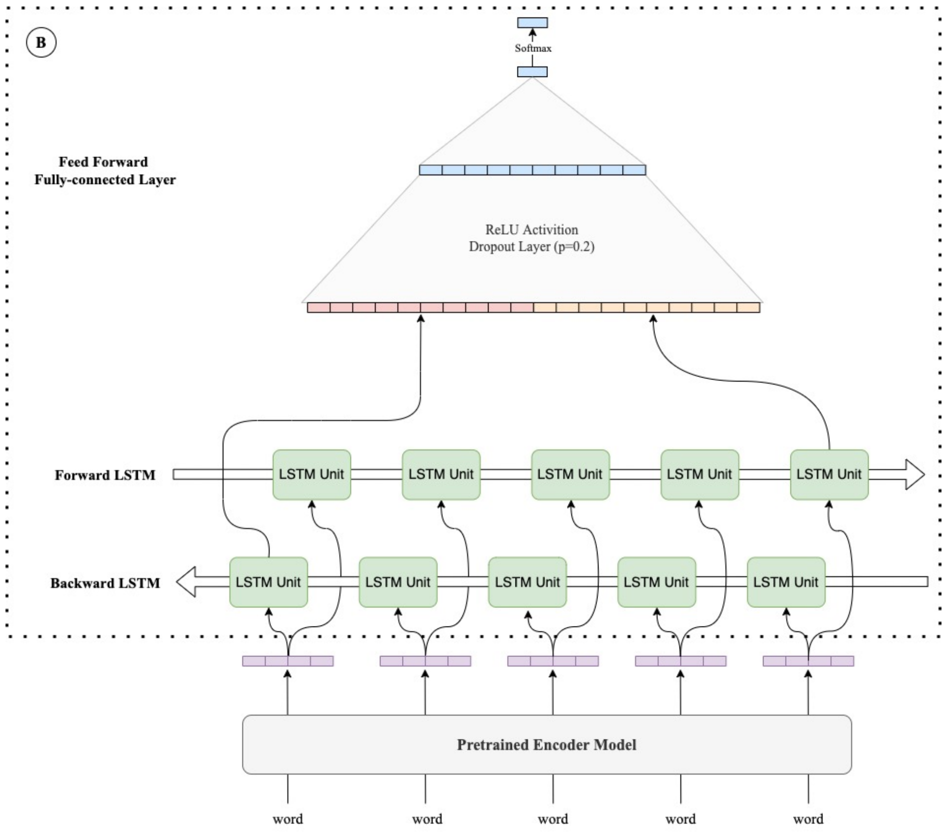


Figure 2: Model diagram

1. Import the tokenizer using Pytorch from the `transformer` package by Hugging Face.
2. Generate a list of all emojis from Emoji List v14.0 by Unicode emoji[4]. Each emoji can be regarded as a emoji token.
3. Use `tokenizer.encode` to convert each emoji token into a token ID.
4. Use `tokenizer.decode` to convert each token ID from last step back to the corresponding token by consulting the tokenizer vocabulary.
5. Compare each original emoji token (o_i) with the returned token (r_i) from step 4. Compute the emoji-compatibility μ for this tokenizer by

$$\mu = \frac{\sum_{i=1}^{n} \mathbb{1}_{\{o_i = r_i\}}}{n} \times 100\%$$

The rationale behind is that tokenizers will encode all out-of-vocabulary tokens into a single token ID, which will be decoded as the unknown token. If the emoji token is recognized by the tokenizer, the decode function will return the exact same emoji token as inputted.

3.3.2 Proposed Data Preprocessing Methods

In our experiment, we proposed 5 data preprocessing methods to incorporate emoji information into the SMSA model as opposed to removing emojis (rm) from the original tweets.

Directly Encode (dir) Use the pretrained encoders or predefined word embeddings that support emojis to directly encode and vectorize the whole sentence. Emojis will be treated as a normal word token. This is a very straightforward and intuitive way to including the emojis..

[4]https://www.unicode.org/emoji/charts/emoji-list.html

Replacing emojis with descriptions [30] (emoji2desc) Currently encoder models are not specifically trained with emoji-embedded text, so they might not have satisfactory vector representations for emojis. Converting emojis back to its official textual description could help the encoder to generate a more accurate vector representation for a certain emoji. Also, as new emojis are designed from time to time, converting emojis to textual description prevented the out-of-vocabulary issue.

Concatenate emojis (concat-emoji) Firstly, the emojis are extracted from the sentences and concatenated at the end of the sentence. We can then encode the whole sentence. Since emojis are not parts of the grammatical structure of the sentence, reposition them at the end can distinguish the written text and emoji information. Note that whether we concatenate the emojis at the end or at the beginning should not make a significant difference since Bi-LSTM model decode the sequence from both ways.

Concatenate description (concat-desc) Besides repositioning the emojis, we also replace them with their textual descriptions for better vector representations and preventing out-of-vocabulary issues.

Meta-feature (meta) Instead of treating emojis as part of the sentence, we can also regard them as high-level features. For each emoji, we match its position, positivity, neutrality, negativity, and sentiment score features from Emoji Sentiment Ranking [26]. The emojis are removed from the sentence when passing through the encoder and Bi-LSTM but encoded separately by the encoder model instead. The emoji vector representations and Emoji Sentiment Ranking features are concatenated with the last hidden states of Bi-LSTM model before being fed into the feed-forward fully connected layer. This method attempts to understand the role of emojis, or more specifically, whether or not emojis should be treated as part-of-speech tokens in SMSA tasks.

3.4 Training and Evaluation

In the experiment, we firstly merged all the dataset consists of the emoji2vec dataset and SemEval-15 Task 11 Dataset. In each trial of the experiment an encoder model is selected with a data preprocessing method that we mentioned above. The process is illustrated in box A of Fig. 1. For each trial, we randomly split the dataset 5 times and train the decoder for 5 rounds respectively, which means we will generate 5 results for each combination of models and methods. Repeating the experiment for 5 times can help us understand the mean accuracy and robustness of the model, along with calculating the 95% confidence intervals.

Training process is conducted on NYU Greene High Performance Computing Cluster with an 8-core CPU and a v100 GPU. Besides, all the process is programmed and automated using Python and PyTorch package. The project is open sourced with an Apache License 2.0. [5]

The best-performing model on the validation set is selected after each round of training and be evaluated on the test set. We use percentage accuracy (%Acc) as the evaluation metrics to measure the performance of different models and proposed methods. To better see the improvement made by emojis in SMSA, the improvement (δ) of incorporating the emojis is calculated and presented in an intuitive way:

$$\delta = \frac{\%\text{Acc}_{\text{method that includes emoji}} - \%\text{Acc}_{\text{method that removes emoji}}}{\%\text{Acc}_{\text{method that removes emoji}}}.$$

4 Results

4.1 Emoji-compatibility Test

In the emoji-compatibility testing, we chose a set of 13 popular BERT-based encoder models. The emoji-compatibility μ of each encoder can be found in Table 2.

We observe that RoBERTa (both base and large version), DeBERTa (both base and large version), BERTweet-large, and Twitter-RoBERTa have unique tokens for all the emojis. However, common encoders like BERT (both base and large version), DistilBERT, and ALBERT nearly do not support any emoji. Note that BERTweet-base cannot support emojis but BERTweet-large can, while XLMRoBERTa (both base and large version) can only recognize about one-fourth of all the emojis. At the end, we selected the encoders that are compatible with all emojis and screened out the others.

[5]https://github.com/BaleChen/emoji-setiment-analysis

Model Name	μ
BERT-base	0.00%
BERT-large	0.19%
RoBERTa-base	100.00%
RoBERTa-large	100.00%
XLMRoBERTa-base	26.61%
XLMRoBERTa-large	26.61%
DeBERTa-base	100.00%
DeBERTa-large	100.00%
BERTweet-base	0.26%
BERTweet-large	100.00%
Twitter-RoBERTa	100.00%
DistilBERT	0.19%
ALBERT	0.00%

Table 2: Emoji-compatibility testing results

4.2 Experiment Results

Firstly, we recorded the mean accuracy across 5 training rounds and presented the results in the following 2-dimensional table (See Table 3). Comparing among different models, we found that using Twitter-RoBERTa

	concat-desc	**concat-emoji**	**dir**	**emoji2desc**	**meta**	**rm**
BERTweet-Large	69.570%	69.560%	69.353%	69.404%	69.399%	68.601%
DeBERTa-base	67.978%	67.991%	68.212%	68.029%	67.791%	66.947%
DeBERTa-large	69.259%	69.314%	69.342%	69.163%	68.610%	68.235%
RoBERTa-base	68.074%	67.946%	68.034%	67.974%	67.977%	67.144%
RoBERTa-large	68.661%	68.785%	68.947%	68.642%	68.664%	68.345%
Twitter-RoBERTa	70.034%	70.525%	***70.687%**	70.045%	70.303%	69.421%
emoji2vec	65.724%	64.474%	64.399%	65.413%	65.086%	64.299%

Table 3: Full Results

model to directly encode emojis stands out among all other combinations with an accuracy of 70.687%, while the Twitter-RoBERTa model is the only one that has accuracy higher than 70%. The model performs notably better than other models. However, `emoji2vec` has the lowest score whichever data preprocessing method we use. All the BERT-based encoders significantly outperforms the pre-defined embedding model `emoji2vec` which had been a useful tool in encoding emojis. The leading 2 models are Twitter-RoBERTa and BERTweet, inferring that the domain similarity contributes to a higher accuracy.

Plotting the confident intervals and averaging across models, we can see the contrast in Fig. 3 The confidence intervals are relatively small, meaning that emoji2vec is inferior to other transformer models with a high statistical confidence, while Twitter-RoBERTa model is significantly performing better than other models in our SMSA task.

Seeing from another perspective, removing all emojis is the worst method among the 6 proposed methods no matter what encoder we use. It lowers the accuracy by 1.202% on average across different models. This result can be justified more rigorously by Fig. 4. The confidence interval of method rm is evidently lower than other methods, having less overlap ranges. For the other methods that includes emojis in the process, the overlapped confidence intervals indicates a relatively blurry distinction.

Seeing from Fig. 4 that shows the improvement index δ, we found that, firstly, all the indice are positive, which strongly justifies the usefulness of emojis in SMSA. Including emojis in the data would improve the

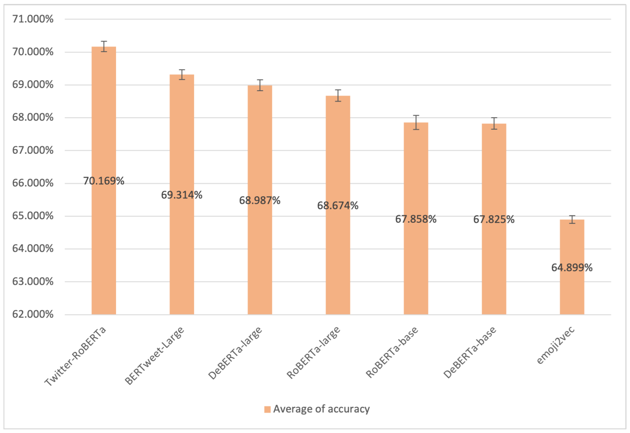


Figure 3: Bar Chart with Confidence Intervals for the Performance of Different Encoder Models

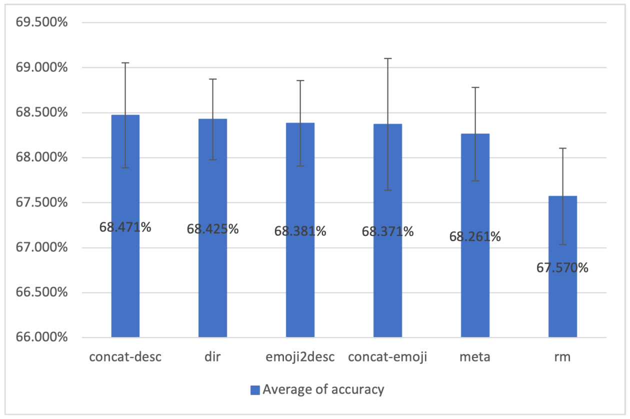


Figure 4: Bar Chart with Confidence Intervals for the Performance of Different Data Preprocessing Methods

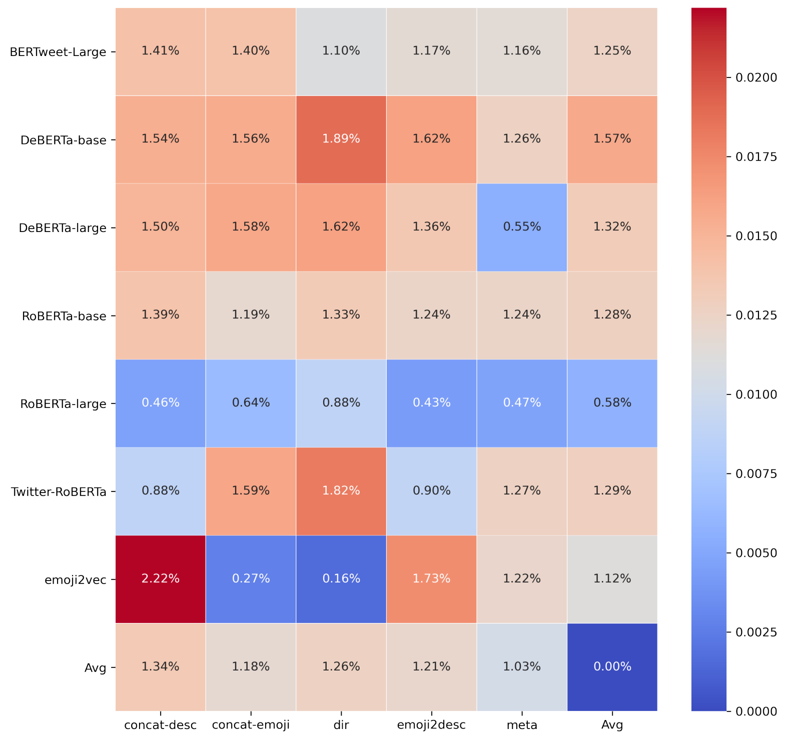
model performance. The maximum and minimum increase appears on the same encoder model – `emoji2vec`. This result indicates that emoji2vec is more sensitive to the data preprocessing methods than other encoders. It is likely to be attributed to the relatively worse vector representations of `emoji2vec` because the δ is notably higher when emojis are converted to their textual descriptions or considered as meta-features. Using `emoji2vec` embeddings to vectorize emoji tokens in this sense is making a trivial betterment than removing the emojis.

Also, we observe from Figure 5 that, averaging across different encoder models, the difference in δ is insignificant among the 5 methods. So, in our experiment, not enough empirical evidence can indicate which method is better for including emojis in SMSA. On the other hand, averaging across different methods, we see that RoBERTa-large encoder does not gain as much edge from emojis. This peculiar result might be explained by the fact that RoBERTa-large's architecture might be more suitable for understanding pure text, but it still awaits a rigorous justification.

5 Discussion & Conclusion

In this study, we mainly investigated the compatibility and coordination between existing encoder models and emojis, aiming to seek out scientific evidence that shows the usefulness of emojis in SMSA tasks. At the same time, we also experimented different methods of incorporating emojis, examining the role of emojis in terms of expressing sentiment in social media posts. Accordingly, we structured our research and experiments to having both the emoji-compatibility experiment and model training experiments on an actual dataset.

From the emoji-compatibility testing, we found that among the 13 popular BERT-based encoders, half of them cannot recognize and produce unique vector representations for emojis. This finding makes future



	concat-desc	concat-emoji	dir	emoji2desc	meta	Avg
BERTweet-Large	1.41%	1.40%	1.10%	1.17%	1.16%	1.25%
DeBERTa-base	1.54%	1.56%	1.89%	1.62%	1.26%	1.57%
DeBERTa-large	1.50%	1.58%	1.62%	1.36%	0.55%	1.32%
RoBERTa-base	1.39%	1.19%	1.33%	1.24%	1.24%	1.28%
RoBERTa-large	0.46%	0.64%	0.88%	0.43%	0.47%	0.58%
Twitter-RoBERTa	0.88%	1.59%	1.82%	0.90%	1.27%	1.29%
emoji2vec	2.22%	0.27%	0.16%	1.73%	1.22%	1.12%
Avg	1.34%	1.18%	1.26%	1.21%	1.03%	0.00%

Figure 5: Improvement index δ

research in emoji-powered SMSA easier for model selection, and is also useful for industrial usage since people will know what encoders to use when it comes to SMSA.

One of the most significant insights is that including emojis, no matter how you include them, enhances the performance of SMSA models. Some encoders like `emoji2vec`, which was developed in 2015 and prior to the boom of transformer models, hold relatively poor representations of emojis under the standards of this time. These encoders might benefit more from emojis if we convert the emojis to textual descriptions. This finding is consistent with the results from Singh et al. [30]. We also found that currently trending RoBERTa-large model does not benefit as much from the emojis as other transformer-based encoders.

Our research has the following limitations: 1) There's limited number of Twitter datasets available with emojis, our results are not comparable to any benchmarks. 2) Due to the computational power limit and time constraints, 5 rounds of training was done instead of >20 rounds to increase statistical robustness. 3) Aside from percentage accuracy, evaluation metrics like F1-score, recall, precision, etc., might provide more comprehensive insights. The future of emoji-powered SMSA researches can pay more attention to where the improvement comes from to increase intepretablity of our results. Also, since emojis are relatively consistent and largely used across different languages, it is also stimulating to see how the emojis would help with multi-lingual SMSA tasks.

Acknowledgement

I would like to extend my warmest gratitude to my research supervisor and mentor Professor Mathieu Laurière. He provides me with scholarly and resourceful advice, guides me through this summer research. It is my great honor and pleasure to finish this study with him and receive his email greeting on my birthday. Besides, I genuinely appreciate NYUSH for offering me the DURF research opportunity. Thanks to all my friends and family who helped me throughout this summer. The research would not have been possible without any of you ❤️.

References

[1] F. Barbieri, J. Camacho-Collados, F. Ronzano, L. Espinosa Anke, M. Ballesteros, V. Basile, V. Patti, and H. Saggion. Semeval 2018 task 2: Multilingual emoji prediction. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, page 24–33, New Orleans, Louisiana, 2018. Association for Computational Linguistics.

[2] F. Barbieri, L. Espinosa-Anke, and H. Saggion. Revealing Patterns of Twitter Emoji Usage in Barcelona and Madrid. page 6.

[3] S. Barreto, R. Moura, J. Carvalho, A. Paes, and A. Plastino. Sentiment analysis in tweets: an assessment study from classical to modern text representation models. *CoRR*, abs/2105.14373, 2021.

[4] P. Chakriswaran, D. R. Vincent, K. Srinivasan, V. Sharma, C.-Y. Chang, and D. G. Reina. Emotion AI-Driven Sentiment Analysis: A Survey, Future Research Directions, and Open Issues. *Applied Sciences*, 9(24):5462, Dec. 2019.

[5] Y. Chen, J. Yuan, Q. You, and J. Luo. Twitter Sentiment Analysis via Bi-sense Emoji Embedding and Attention-based LSTM. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 117–125, Oct. 2018. arXiv:1807.07961 [cs].

[6] Z. Chen, S. Shen, Z. Hu, X. Lu, Q. Mei, and X. Liu. Emoji-Powered Representation Learning for Cross-Lingual Sentiment Classification. In *The World Wide Web Conference on - WWW '19*, pages 251–262, San Francisco, CA, USA, 2019. ACM Press.

[7] B. Eisner, T. Rocktäschel, I. Augenstein, M. Bošnjak, and S. Riedel. emoji2vec: Learning Emoji Representations from their Description, Nov. 2016. Number: arXiv:1609.08359 arXiv:1609.08359 [cs].

[8] Emojipedia. Emoji statistics, 2022.

[9] A. Esuli and F. Sebastiani. SENTIWORDNET: A publicly available lexical resource for opinion mining. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy, May 2006. European Language Resources Association (ELRA).

[10] B. Felbo, A. Mislove, A. Søgaard, I. Rahwan, and S. Lehmann. 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, 2017. arXiv:1708.00524 [cs, stat].

[11] A. Ghosh, G. Li, T. Veale, P. Rosso, E. Shutova, J. Barnden, and A. Reyes. SemEval-2015 Task 11: Sentiment Analysis of Figurative Language in Twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 470–478, Denver, Colorado, 2015. Association for Computational Linguistics.

[12] A. Go. Sentiment classification using distant supervision. 2009.

[13] P. He, X. Liu, J. Gao, and W. Chen. Deberta: Decoding-enhanced bert with disentangled attention. In *2021 International Conference on Learning Representations*, May 2021.

[14] C. J. Hutto and E. Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *ICWSM*, 2014.

[15] B. Liu. *Sentiment analysis: Mining opinions, sentiments, and emotions*. January 2015.

[16] C. Liu, F. Fang, X. Lin, T. Cai, X. Tan, J. Liu, and X. Lu. Improving sentiment analysis accuracy with emoji embedding. *Journal of Safety Science and Resilience*, 2(4):246–252, Dec. 2021.

[17] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[18] Y. Lou, Y. Zhang, F. Li, T. Qian, and D. Ji. Emoji-Based Sentiment Analysis Using Attention Networks. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 19(5):64:1–64:13, 2020.

[19] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[20] W. Medhat, A. Hassan, and H. Korashy. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, 5(4):1093–1113, 2014.

[21] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space, Sept. 2013. Number: arXiv:1301.3781 arXiv:1301.3781 [cs].

[22] M. V. Mäntylä, D. Graziotin, and M. Kuutila. The evolution of sentiment analysis—a review of research topics, venues, and top cited papers. *Computer Science Review*, 27:16–32, 2018.

[23] P. Nakov, A. Ritter, S. Rosenthal, F. Sebastiani, and V. Stoyanov. SemEval-2016 task 4: Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1–18, San Diego, California, June 2016. Association for Computational Linguistics.

[24] P. Nakov, S. Rosenthal, Z. Kozareva, V. Stoyanov, A. Ritter, and T. Wilson. SemEval-2013 task 2: Sentiment analysis in Twitter. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 312–320, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics.

[25] D. Q. Nguyen, T. Vu, and A. T. Nguyen. BERTweet: A pre-trained language model for English Tweets, Oct. 2020. arXiv:2005.10200 [cs].

[26] P. K. Novak, J. Smailović, B. Sluban, and I. Mozetič. Sentiment of Emojis. *PLOS ONE*, 10(12):e0144296, Dec. 2015. arXiv:1509.07761 [cs].

[27] J. Pennington, R. Socher, and C. Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, Oct. 2014. Association for Computational Linguistics.

[28] D. Rodrigues, M. Prada, R. Gaspar, M. V. Garrido, and D. Lopes. Lisbon emoji and emoticon database (leed): Norms for emoji and emoticons in seven evaluative dimensions. *Behavior Research Methods*, 50(1):392–405, Feb 2018.

[29] S. Rosenthal, N. Farra, and P. Nakov. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518, Vancouver, Canada, Aug. 2017. Association for Computational Linguistics.

[30] A. Singh, E. Blanco, and W. Jin. Incorporating Emoji Descriptions Improves Tweet Classification. In *Proceedings of the 2019 Conference of the North*, pages 2096–2101, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.

[31] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 61:2544–2558, 12 2010.

[32] C. Van Hee, E. Lefever, and V. Hoste. SemEval-2018 task 3: Irony detection in English tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 39–50, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[33] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.