Data_Preprocessing_NLP_Removing_emojis_emoticons_Flags

By: Venkatesh_Mungi venkateshmungi1247@gmail.com

Text Preprocessing: Handle Emoji & Emoticon

Text pre-processing step is a very crucial stage when you work with Natural Language Processing (NLP). There are many text pre-processing methods we need to conduct in text cleaning stage such as handle stop words, special characters, emoji, emoticon, punctuations, spelling correction, URL, etc.

Nowadays in a day to day life, people often use emoji and emoticon in a sentence to express the feeling or describe object instead of writing a word in a social media platform.

Both emoji and emoticon convey emotional expression in a text message. The main difference between emoji and emoticon: an emoji is a small actual image that is used to express emotions or idea in text messages. where an emoticon is a facial expression representation using keyboard characters and punctuations. For example, ? is an emoji and ‘:)’ is an emoticon that represents a happy face.

For text analysis, we might need to handle it. In this tutorial, you will get to know about how to handle emoji/emoticon in a text with examples. You can remove the emojis from the text for text analysis. Sometimes, emoticons give strong information about a text such as feeling expression.

For example, if we are working with product reviews sentiment analysis data where we have to identify each review whether it is positive or negative. People may write the emoji in a review to express the feeling towards the product how much he/she is satisfied with the product. In that case, we may lose valuable information if we remove the emojis. In this case, a better approach is to convert emoji to word format so that it preserves the emoji information.

In [1]:
```
try:
    import cPickle as pickle
except ImportError:
    import pickle
import re

with open('Emoji_Dict.p', 'rb') as fp:
    Emoji_Dict = pickle.load(fp)
Emoji_Dict = {v: k for k, v in Emoji_Dict.items()}

def convert_emojis_to_word(text):
    for emot in Emoji_Dict:
        text = re.sub(r'('+emot+')', "_".join(Emoji_Dict[emot].replace(",","").replace(":","").split()), text)
    return text

text = "I won ? in ?"
convert_emojis_to_word(text)
```

Out[1]:
```
'I won ? in ?'
```

In [2]:
```
convert_emojis_to_word("I like to eat ?")
```

Out[2]:
```
'I like to eat ?'
```

Handle Emoticon

In [3]:
```
import re
try:
    import cPickle as pickle
except ImportError:
    import pickle

with open('Emoticon_Dict.p', 'rb') as fp:
    Emoticon_Dict = pickle.load(fp)

def remove_emoticons(text):
    emoticon_pattern = re.compile(u'(' + u'|'.join(k for k in Emoticon_Dict) + u')')
    return emoticon_pattern.sub(r'', text)

remove_emoticons("Good Morning :-)")
```

Out[3]:
```
'Good Morning '
```

Removing Emojis Using demoji library

In [4]:
```
import demoji # Remove all kind of emojis with the “demoji” package
```

In [5]:
```
def remove_em(text):
    dem = demoji.findall(text)
    for item in dem.keys():
        text = text.replace(item, "")
    return text

text = "H!😂I am very 🤣very 🤣 Happy🤣 to say 😘that, we will👍🎉 do ✨⭐✨this🤠 great job."
remove_em(text)
```

Out[5]:
```
'H!I am very very  Happy to say that, we will do this great job.'
```

demoji exports several text-related functions for find-and-replace functionality with emojis:

In [6]:
```
tweet = """\#startspreadingthenews yankees win great start by 🎅 going 5strong innings with 5k’s🔥 🐂 solo homerun 🌋🌋 with 2 solo homeruns and
demoji.findall(tweet)
```

Out[6]:
```
{'🎅': 'Santa Claus: medium-dark skin tone',
 '🌋': 'volcano',
 '🇳🇮': 'flag: Nicaragua',
 '👹': 'ogre',
 '🇲🇽': 'flag: Mexico',
 '🚣🏼': 'person rowing boat: medium-light skin tone',
 '🔥': 'fire',
 '🐂': 'ox',
 '👨🏽\u200d⚖️': 'man judge: medium skin tone',
 '🤡': 'clown face'}
```

Footnote: Emoji Sequences

Numerous emojis that look like single Unicode characters are actually multi-character sequences. Examples:

> The keycap 2️⃣ is actually 3 characters, U+0032 (the ASCII digit 2), U+FE0F (variation selector), and U+20E3 (combining enclosing keycap).
> The flag of Scotland 7 component characters, b'\\U0001f3f4\\U000e0067\\U000e0062\\U000e0073\\U000e0063\\U000e0074\\U000e007f' in full esap
> ed notation.

(You can see any of these through s.encode("unicode-escape").)

demoji is careful to handle this and should find the full sequences rather than their incomplete subcomponents.

The way it does this it to sort emoji codes by their length, and then compile a concatenated regular expression that will greedily search for longer emojis first, falling back to shorter ones if not found. This is not by any means a super-optimized way of searching as it has O(N2) properties, but the focus is on accuracy and completeness.

In [7]:
```
from pprint import pprint
seq = """\I bet you didn't know that 🙋, 🤷, and 🙋‍♀️ are three different emojis."""
pprint(seq.encode('unicode-escape'))
```

```
(b"\\\\I bet you didn't know that \\U0001f64b, \\U0001f64b\\u200d\\u2642\\ufe0"
 b'f, and \\U0001f64b\\u200d\\u2640\\ufe0f are three different emojis.')
```

Removing emoticons,symbols & pictographs, transport & map symbols, flags (iOS), dingbats using regex module

In [8]:
```
import re

text = u'This is a smiley face \U0001f602'
print(text) # with emoji

def deEmojify(text):
    regrex_pattern = re.compile(pattern = "["
        u"\U0001F600-\U0001F64F"  # emoticons
        u"\U0001F300-\U0001F5FF"  # symbols & pictographs
        u"\U0001F680-\U0001F6FF"  # transport & map symbols
        u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                           "]+", flags = re.UNICODE)
    return regrex_pattern.sub(r'',text)

print(deEmojify(text))
```

```
This is a smiley face 😂
This is a smiley face
```

In [9]:
```
import re

data = "I bet you didn't know that 🙆, 🙋, and 🙋 are three different emojis."
def remove_emojis(data):
    emoj = re.compile("["
        u"\U0001F600-\U0001F64F"  # emoticons
        u"\U0001F300-\U0001F5FF"  # symbols & pictographs
        u"\U0001F680-\U0001F6FF"  # transport & map symbols
        u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
        u"\U00002500-\U00002BEF"  # chinese char
        u"\U00002702-\U000027B0"
        u"\U00002702-\U000027B0"
        u"\U000024C2-\U0001F251"
        u"\U0001f926-\U0001f937"
        u"\U00010000-\U0010ffff"
        u"\u2640-\u2642"
        u"\u2600-\u2B55"
        u"\u200d"
        u"\u23cf"
        u"\u23e9"
        u"\u231a"
        u"\ufe0f"  # dingbats
        u"\u3030"
                      "]+", re.UNICODE)
    return re.sub(emoj, '', data)

remove_emojis(data)
```

Out[9]:
```
"I bet you didn't know that , , and  are three different emojis."
```

Working with Emoji Module

In [10]:
```
import emoji
print(emoji.emojize('Python is :thumbs_up:'))
print(emoji.emojize('Python is :thumbsup:', language='alias'))
print(emoji.demojize('Python is 👍'))
print(emoji.emojize("Python is fun :red_heart:"))
print(emoji.emojize("Python is fun :red_heart:", variant="emoji_type"))
print(emoji.is_emoji("👍"))
```

```
Python is 👍
Python is 👍
Python is :thumbs_up:
Python is fun ❤️
Python is fun ❤️
True
```

By default, the language is English (language='en') but also supported languages are:

Spanish ('es'), Portuguese ('pt'), Italian ('it'), French ('fr'), German ('de'), Farsi/Persian ('fa')

In [11]:
```
print(emoji.emojize('Python es :pulgar_hacia_arriba:', language='es'))
print(emoji.demojize('Python es 👍', language='es'))
print(emoji.emojize("Python é :polegar_para_cima:", language='pt'))
print(emoji.demojize("Python é 👍", language='pt'))
```

```
Python es 👍
Python es :pulgar_hacia_arriba:
Python é 👍
Python é :polegar_para_cima:
```

###To Know more about Emojis and emoticons go through following links:

http://www.unicode.org/emoji/charts/full-emoji-list.html (http://www.unicode.org/emoji/charts/full-emoji-list.html)

https://www.webfx.com/tools/emoji-cheat-sheet/ (https://www.webfx.com/tools/emoji-cheat-sheet/)

https://www.geeksforgeeks.org/python-program-to-print-emojis/ (https://www.geeksforgeeks.org/python-program-to-print-emojis/)

In [ ]: