

Data Labelling

All sentiment scores fall within the range $[-1, 1]$. A sentiment score may be less than 0 (negative), equal to zero (neutral) or greater than 0 (positive). Sentiment labels may be equal to -1 (negative), equal to 0 (neutral) or equal to 1 (positive). Tweets having no Human/TextBlob/Vader/Emoji sentiment score/label associated to it is assigned a value of -2.

The labelling strategy adopted is as follows.

- Each tweet text body was assigned a sentiment score by both TextBlob and Vader NLP libraries. The reason for choice of these libraries shall be explained further along in this notebook in detail.
- Each tweet was assigned an emoji sentiment score depending on the emojis (if any) in the tweet.
- All scores range between $[-1, 1]$ with a score value being set to -2 if the tweet was not scored. From these tweet scores, tweet labels (-1 = negative, 0 = neutral, 1 = positive) was obtained by rounding the value of the tweet scores using python's `round()` function.
- 2 Humans labelled 102 tweets by hand. The hand assigned labels were compared against each other and labels assigned by other means mentioned above. It was found that TextBlob's labels were largely inaccurate while those obtained from Vader and emoji scores consistently show apt reflection of text sentiment.
- Thus, the decision was made to accept an average of vader and emoji sentiment scores whenever the emoji sentiment score is available and only the vader score otherwise as the final score to be input into the `round()` function to produce the final sentiment label associated with each emoji.
- Other means of assigning sentiment scores like using "sentiment140" and "NLTK" were explored but these options were later dropped due to lack of evidence of accurate results as was the case with sentiment140 surrounding which documentation was scarce or due to requirement of very tedious text preprocessing that had to be done before possibly questionable sentiment scores could be obtained as was the difficulty with the NLTK library.
- The sentiment labels assigned using the currently adopted method was manually checked against corresponding tweet text bodies by team members and were verified to reflect seemingly correct sentiment in almost all cases.

Imports

```
In [1]: 1 import pandas as pd
        2 import numpy as np
        3 from textblob import TextBlob
        4 import nltk
        5 from nltk.sentiment.vader import SentimentIntensityAnalyzer
        6 import os
```

Utility Functions

```
In [2]: 1 ''' Returns a list of all indices in the given string
2         where the given substring was found.
3         '''
4     def find_all(string, substring):
5         indices = []
6         while True:
7             index = string.find(substring)
8             if index == -1: break
9             indices.append(index)
10            string = string[index+len(substring):]
11        return indices

In [3]: 1 ''' Returns an object containing no. of same and different
2         elements when comparing two given pandas series. A
3         score depicting the no. of same elements out of total no. of
4         elements is also returned. Both series must be of the same length.
5         '''
6     def compare_series_values(series1, series2):
7         if len(series1) != len(series2): raise ValueError("series1 and series2 must be of the same length")
8         comparison = series1 == series2
9         value_counts = comparison.value_counts()
10        n_different = value_counts[0]
11        n_same = value_counts[1]
12        score = "{}/{ {}".format(n_same, len(comparison))
13        return {"same": n_same, "different": n_different, "score": score}
```

Load Tweets Data


```
In [4]: 1 # Load tweets dataset
2     TWEETS_DATASET_PATH = "./data/tweets.csv"
3     TWEETS_DF = pd.read_csv(TWEETS_DATASET_PATH)
4     display(TWEETS_DF.head(3))
5     print(len(TWEETS_DF))
```

	Tweet Body
0	Its #Expo2020 Day
1	We celebrated the National Day of the Slovak R...
2	Vertebral Deformity Measurements on MRI, CT, a...

6613

In [17]:

```
1 # Load hand assigned labels dataset
2 TWEETS_HUMAN_LABELLED_DATASET_PATH = "./data/tweets_human_labelled.csv"
3 TWEETS_HUMAN_LABELLED_DF = pd.read_csv(TWEETS_HUMAN_LABELLED_DATASET_PA
4 display(TWEETS_HUMAN_LABELLED_DF.head(3))
5 print(len(TWEETS_HUMAN_LABELLED_DF))
```

	Tweet Body	TextBlob Sentiment Score	TextBlob Sentiment Label	Vader Sentiment Score	Vader Sentiment Label	Emoji Sentiment Score	Emoji Sentiment Label	Human' Sentimen Labe
0	Meet us at #Expo2020 in Dubai to celebrate Rwa...	0.500000	0	0.7624	1	-2.000000	-2	1.0
1	 "Connecting beauty with sustainability &...	0.333333	0	0.7964	1	0.499813	0	1.0
2	Join us for a week of events and activities as...	0.500000	0	0.9274	1	0.527473	1	1.0

1044

Vader & TextBlob Sentiment Labelling

Text Blob is a text processing tool/library used in the aiding of Natural Language processing. There are multiple uses of this library out of which we have primarily used the "sentiment analysis" portion for this use case. (PyPI, 2021)

Vader or "Valence Aware Dictionary and Sentiment Reasoner" is another tool provided by the nltk package. It is vastly used in sentiment analysis to test how negative, how positive or how neutral a word could be. These values indicate the intensity of the sentiment being conveyed (Some sentences may be more negative than others, or vice versa). (Beri, 2020)

Both approaches are lexicon based approaches, that is, the sentiment is assigned based on a predefined dictionary of words that are mapped to these sentiments. Sentiments are assigned per word after which an overall sentiment is assigned to the sentence as a whole using the individual sentiments. Both approaches also provide their overall score from a range of -1 to 1; -1 being at the extreme negative, +1 being an extreme positive and 0 being neutral. It is noteworthy that these approaches reverse sentiments according to negations as well. eg: "not good" would not be assigned a positive sentiment.

Pros and cons of the lexical approach

(DeLancey, 2020)

- The approach is relatively easy to understand and implement for straightforward tasks.
- This approach makes it easy to analyze large datasets.
- The nature of the analysis forces it to overlook certain words, given spelling errors, which could change the meaning of the sentence.
- The analyzer is put in a difficult spot when having to deal with irony due to its

predefined mapping to sentiments.

Comparing Textblob and Vader

- According to an analysis seen in (White, 2020), it was observed that Vader performed much better with twitter data, that is, text that contain slangs, emojis, and more. Whereas textblob seemed to perform better with more formal text data.
- According to our analysis on the experiment dataset, we can also observe the distribution of sentiments as follows:

Sentiment	TextBlob	Vader
negative	0.3%	1.4%
neutral	91%	56%
positive	8.5%	42%

It is noteworthy that TextBlob assigns most of its sentiments as neutral and vader has a more uniform distribution of the sentiments. TextBlob seemed to have assigned a lot of the tweets to neutral (even tweets that were deemed very positive by the human subjects).

- After comparing the human labelled tweets to the labels assigned by TextBlob and Vader, we came to a conclusion that Vader was able to mimic humans in understanding the sentiments a little better than TextBlob was able to.

```
In [5]: 1 # Load vader lexicon and sentiment analyzer
        2 nltk.download('vader_lexicon')
        3 sid = SentimentIntensityAnalyzer()
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data] C:\Users\ardileep\AppData\Roaming\nltk_data...
[nltk_data] Package vader_lexicon is already up-to-date!
```

```
In [6]: 1 ''' Returns sentiment scores in the range [-1, 1]
        2 assigned to a given text using TextBlob's and
        3 Vader's sentiment analyzer respectively.
        4 '''
        5 def get_tbv_sentiment_scores(text):
        6     TAnalysis = TextBlob(text)
        7     Vanalysis = sid.polarity_scores(text)
        8     return TAnalysis.sentiment.polarity, Vanalysis["compound"]
```

In [7]:

```
1 # get sentiment of text body of each tweet
2 TBscores=[] # textblob sentiment scores
3 TBalias=[] # textblob sentiment Labels
4 Vscores=[] # vader sentiment scores
5 Valias=[] # vader sentiment Labels
6 for tweet in TWEETS_DF["Tweet Body"]:
7     TBscore, Vscore = get_tbv_sentiment_scores(tweet)
8     TBscores.append(TBscore)
9     TBalias.append(round(TBscore))
10    Vscores.append(Vscore)
11    Valias.append(round(Vscore))
12 TWEETS_DF["TextBlob Sentiment Score"] = TBscores
13 TWEETS_DF["TextBlob Sentiment Label"] = TBalias
14 TWEETS_DF["Vader Sentiment Score"] = Vscores
15 TWEETS_DF["Vader Sentiment Label"] = Valias
16
17 display(TWEETS_DF.head(3))
18 print(len(TWEETS_DF))
```

	Tweet Body	TextBlob Sentiment Score	TextBlob Sentiment Label	Vader Sentiment Score	Vader Sentiment Label
0	Its #Expo2020 Day	0.00000	0	0.0000	0
1	We celebrated the National Day of the Slovak R...	0.61875	1	0.9531	1
2	Vertebral Deformity Measurements on MRI, CT, a...	0.00000	0	0.0000	0

6613

Emoji Sentiment Labelling

Studies show that emojis are more than just a trend. They are universally loved and frequently used by people across age groups due to the ease with which they allow one to express sentiment through text. (M. Dols de Jong, 2019) (A. Atanasova, 2016)

Identifying sentiment from a chat message or tweet is often challenging. Human emotion is complicated. When a chat/tweet text body itself may seem to carry a negative sentiment, emojis present may indicate otherwise and may hint at that piece of text being written within a humorous/less serious context in which case the true sentiment expressed by that text is likely neutral or maybe even positive. (A. Atanasova, 2016)

Since the sole purpose of emojis is to allow users to emote via text, it is highly likely that identifying the type and frequency of emojis in a text body can contribute to gaining an understanding about the sentiment surrounding that piece of text. Thus, during the sentiment labelling process, the emojis in the tweet have been identified and mapped to a sentiment score calculated using data from an emoji lexicon called "Emoji Sentiment Ranking 1.0" that can be found at <https://www.clarin.si/repository/xmlui/handle/11356/1048> (<https://www.clarin.si/repository/xmlui/handle/11356/1048>). The .csv version of this lexicon named "Emoji_Sentiment_Data_v1.0.csv" on the website was downloaded and used. (K. Novak et al, 2015, [dataset])

The "Emoji Sentiment Ranking 1.0" lexicon contains 751 emojis and associated sentiment

data (no. of occurrences of emoji in positive/negative/neutral tweets) gathered from 70000 tweets that were hand labelled by 83 humans across 13 European languages. While it is understood that this source has the limitation that its data remains confined within context of european languages, this dataset was still accepted since body language/facial expression that popular emojis convey are largely universal basic emotion. Also, the decision was made to use this dataset publically available under the creative commons license because it was developed as part of an academic endeavor and uses accredited methods as documented in K. Noval et al's work "Sentiment of Emojis" (K. Novak et al, 2015). Moreover, a large no. of tweets of different languages were analyzed by a sufficiently large no. of people.

For all 751 emojis in the "Emoji Sentiment Ranking 1.0" dataset, the following cell calculates a sentiment score in the range [-1, 1] (where a score less than zero indicates negative sentiment, equal to zero indicates neutral and more than zero indicates positive sentiment) from the no. of occurrences of each emoji in each category of tweets (negative sentiment tweet, positive sentiment tweet, neutral sentiment tweet). The logic used to calculate the score is as follows.

- The frequency distribution values of each emoji across the 3 tweet sentiment categories were normalized.
- Then the normalized negative sentiment value was multiplied by -1, the neutral sentiment value by 0 and the positive sentiment value was multiplied by 1 before summing up all the values to produce the final sentiment score.
- The neutral sentiment value was multiplied by 0 and thus effectively ignored since if a score is to be neutral, the positive and negative scores would simply cancel each other out.

In [8]:

```
1 # get emoji lexicon
2 EMOJI_CORPUS_PATH = "./data/emoji_sentiment_data.csv"
3 EMOJI_DF = pd.read_csv(EMOJI_CORPUS_PATH)
4 EMOJI_DF = EMOJI_DF[["Emoji", "Occurrences", "Negative", "Neutral", "Po
5
6 # calculate sentiment score associated with each emoji from no. of
7 # positive, neutral and negative occurrences of that emoji
8 EMOJI_DF["Sentiment"] = [
9     (-1*(neg/o)) + (0*(neu/o)) + (1*(pos/o))
10     for o, neg, neu, pos
11     in zip(EMOJI_DF["Occurrences"], EMOJI_DF["Negative"], EMOJI_DF["Neu
12 ]
13
14 display(EMOJI_DF.head(3))
15 print(len(EMOJI_DF))
```

	Emoji	Occurrences	Negative	Neutral	Positive	Sentiment
0	😄	14622	3614	4163	6845	0.220968
1	❤	8050	355	1334	6361	0.746087
2	♥	7144	252	1942	4950	0.657615

```
In [9]: 1 # store useful data from emoji lexicon in a list and dictionary for eas
2 EMOJI_LIST = EMOJI_DF["Emoji"].to_list()
3 print("EMOJI_LIST = {} ...".format(EMOJI_LIST[:5]))
4
5 EMOJI_SENTIMENT_MAP = {emoji:sentiment for emoji, sentiment in zip(EMOJ
6 print("EMOJI_SENTIMENT_MAP = {} ... ".format(list(EMOJI_SENTIMENT_MAP.i

EMOJI_LIST = ['😄', '❤️', '♥️', '😍', '🤔'] ...
EMOJI_SENTIMENT_MAP = [('😄', 0.22096840377513338), ('❤️', 0.7460869565217
392)] ...
```

```
In [10]: 1 ''' Returns a list of emojis in a given sentence. '''
2 def get_emojis_in_text(sentence):
3     global EMOJI_LIST
4     emojis = []
5     for emoji in EMOJI_LIST:
6         for i in range(len(find_all(sentence, emoji))): emojis.append(e
7     return emojis
```

```
In [11]: 1 # get emoji sentiment of text body of each tweet
2 tweet_emoji_sentiment_scores = [] # emoji sentiment scores
3 tweet_emoji_sentiment_labels = [] # emoji sentiment labels
4 for index, row in TWEETS_DF.iterrows():
5     text = row["Tweet Body"]
6     tweet_emoji_list = get_emojis_in_text(text)
7     num_emojis = len(tweet_emoji_list)
8
9     # if the tweet has no emojis then set score to -2.
10    if num_emojis == 0:
11        tweet_emoji_sentiment_scores.append(-2)
12        tweet_emoji_sentiment_labels.append(-2)
13        continue
14
15    # if the tweet has emojis, calculate average score of all emojis.
16    score = 0
17    for emoji in tweet_emoji_list: score += EMOJI_SENTIMENT_MAP[emoji]
18    score = score/num_emojis
19    tweet_emoji_sentiment_scores.append(score)
20    tweet_emoji_sentiment_labels.append(round(score))
21
22    # NOTE: if a tweet has no emojis it is assigned an emoji sentiment scor
23 TWEETS_DF["Emoji Sentiment Score"] = tweet_emoji_sentiment_scores
24 TWEETS_DF["Emoji Sentiment Label"] = tweet_emoji_sentiment_labels
25
26 display(TWEETS_DF.head(3))
```

	Tweet Body	TextBlob Sentiment Score	TextBlob Sentiment Label	Vader Sentiment Score	Vader Sentiment Label	Emoji Sentiment Score	Emoji Sentiment Label
0	Its #Expo2020 Day	0.00000	0	0.0000	0	-2.0	-2
1	We celebrated the National Day of the Slovak R...	0.61875	1	0.9531	1	-2.0	-2
2	Vertebral Deformity Measurements on MRI, CT, a...	0.00000	0	0.0000	0	-2.0	-2

Add Hand Assigned Labels

Some of the tweets were assigned labels by humans in order to test how close the calculated results would come to a sentiment assigned by a human (2 team members took part in this). But these labels were not chosen as the final labels because the labels assigned by the 2 humans varied by about 25%. This was expected as humans could have a lot of factors that affect their judgement of a sentiment, one of them being biases. Doing this experiment grounded why using human assigned labels might not be the most reliable source when only few humans are available for the labelling process. In order for hand assigned labels to be a true reflection of the general sentiment of a tweet with less individual bias, several humans (ideally at least 50 or more) would have to label all the tweets after which the final label of the tweet would be based on the mean of all the hand assigned labels. Given, the large no of tweets to be labelled and the requirement for a large no of humans. The decision was made to use existing tested/proven tools like Vader to obtain labels for the tweets instead of hand labelling them all as this would be the most reliable and practical way to label a large no. of tweets (>2000) within the given timeframe.

In [193]:

```
1 ''' Aids in hand labelling tweets by displaying tweet
2     and prompting to input a sentiment label,
3     's' for skip or 'q' for quit. Entered labels are
4     added to the tweets_human_labelled.csv file as
5     a new column.
6 '''
7 def hand_assign_sentiment_labels():
8     global TWEETS_HUMAN_LABELLED_DF
9
10    print(
11        "Please assign a sentiment label -1, 0 or 1 (-1 = negative, 0 =
12        + " to each tweet given below. Enter 's' to skip a tweet or 'q'
13    )
14
15    tweets = TWEETS_HUMAN_LABELLED_DF["Tweet Body"].to_list()
16    n_tweets = len(tweets)
17    new_column_name = "Human" + str(
18        len(find_all(" ".join(TWEETS_HUMAN_LABELLED_DF.columns), "Human
19    ) + " Sentiment Label"
20
21    assigned_labels = [-2]*len(TWEETS_HUMAN_LABELLED_DF)
22    for i in range(n_tweets):
23        while True:
24            input_value = input("{} / {} | {} :".format(i, n_tweets, tweets[i]))
25            if input_value in ['s', 'S']: # if 's' is input, skip this tweet
26                print("index {} skipped".format(i))
27                break
28            if input_value in ['q', 'Q']: # if 'q' is input, quit and save labels
29                TWEETS_HUMAN_LABELLED_DF[new_column_name] = assigned_labels
30                print("Thank You :3")
31                return
32            input_value = int(input_value)
33            if not (input_value in [-1, 0, 1]): # if an invalid label is entered
34                print("Invalid entry. Try Again.")
35            else:
36                print("assigned {} at index {}".format(input_value, i))
37                assigned_labels[i] = input_value # if correct label is entered
38                break
39    # save labels when all tweets have been labelled
40    TWEETS_HUMAN_LABELLED_DF[new_column_name] = assigned_labels
41    print("Thank You :3")
```

```
In [194]: 1 # UNCOMMENT TO LABEL TWEETS BY HAND
          2 hand_assign_sentiment_labels()
```

Please assign a sentiment label -1, 0 or 1 (-1 = negative, 0 = neutral, 1 = positive) to each tweet given below. Enter 's' to skip a tweet or 'q' to quit labelling.

```
assigned 0 at index 0
assigned 1 at index 1
assigned 1 at index 2
assigned 1 at index 3
assigned 1 at index 4
assigned 1 at index 5
assigned 1 at index 6
assigned 1 at index 7
assigned 1 at index 8
assigned 1 at index 9
assigned 1 at index 10
assigned 1 at index 11
assigned 1 at index 12
assigned 0 at index 13
assigned 0 at index 14
assigned 1 at index 15
assigned 1 at index 16
```

```
In [ ]: 1 # NOTE: this cell should only be run once
        2 TWEETS_DF = TWEETS_DF.join(TWEETS_HUMAN_LABELLED_DF.drop(["Tweet Body"]))
        3 display(TWEETS_DF.head(3))
```

Save Labelled Tweets Dataset

```
In [16]: 1 TWEETS_LABELLED_DATASET_PATH = "./data/tweets_labelled.csv"
          2 if os.path.exists(TWEETS_LABELLED_DATASET_PATH):
          3     input_value = input("{} already exists, replace? (y/n)".format(TWEETS_LABELLED_DATASET_PATH))
          4     if input_value in ["y", "Y"]:
          5         print("saving {} ...".format(TWEETS_LABELLED_DATASET_PATH))
          6         TWEETS_DF.to_csv(TWEETS_LABELLED_DATASET_PATH, index=False)
          7         print("done! :3")
          8     else:
          9         TWEETS_DF.to_csv(TWEETS_LABELLED_DATASET_PATH, index=False)
         10         print("done! :3")
```

done! :3

Compare Labels

In [17]:

```
1 LABELLED_TWEETS_DF = pd.read_csv(TWEETS_LABELLED_DATASET_PATH)
2 display(LABELLED_TWEETS_DF.head(3))
3 print(len(LABELLED_TWEETS_DF))
```

	Tweet Body	TextBlob Sentiment Score	TextBlob Sentiment Label	Vader Sentiment Score	Vader Sentiment Label	Emoji Sentiment Score	Emoji Sentiment Label
0	Its #Expo2020 Day	0.00000	0	0.0000	0	-2.0	-2
1	We celebrated the National Day of the Slovak R...	0.61875	1	0.9531	1	-2.0	-2
2	Vertebral Deformity Measurements on MRI, CT, a...	0.00000	0	0.0000	0	-2.0	-2

6613

In [198]:

```
1 compare_series_values(
2     LABELLED_TWEETS_DF["Human1 Sentiment Label"][:102],
3     LABELLED_TWEETS_DF["Human2 Sentiment Label"][:102]
4 )
```

Out[198]: {'same': 77, 'different': 25, 'score': '77/102'}

Calculate overall average of the Vader, TextBlob and the Emoji score

In [18]:

```
1 TWEETS_LABELLED_DATASET_PATH = "./data/tweets_labelled.csv"
2 LABELLED_TWEETS_DF = pd.read_csv(TWEETS_LABELLED_DATASET_PATH)
3 display(LABELLED_TWEETS_DF.head(3))
```

	Tweet Body	TextBlob Sentiment Score	TextBlob Sentiment Label	Vader Sentiment Score	Vader Sentiment Label	Emoji Sentiment Score	Emoji Sentiment Label
0	Its #Expo2020 Day	0.00000	0	0.0000	0	-2.0	-2
1	We celebrated the National Day of the Slovak R...	0.61875	1	0.9531	1	-2.0	-2
2	Vertebral Deformity Measurements on MRI, CT, a...	0.00000	0	0.0000	0	-2.0	-2

```
In [19]: 1 average_scores=[]
2 average_alias=[]
3 #take the sum of vader and emoji scores if emojis exist in the tweet(em
4 for i in range(len(LABELLED_TWEETS_DF)):
5     if (LABELLED_TWEETS_DF["Emoji Sentiment Score"][i]!=-2):
6         average=(LABELLED_TWEETS_DF["Vader Sentiment Score"][i]
7         +LABELLED_TWEETS_DF["Emoji Sentiment Score"][i])/2
8         average_scores.append(average)
9         average_alias.append(round(average))
10 #if emojis dont exist in the tweet, take only the vader score
11     else:
12         average=LABELLED_TWEETS_DF["Vader Sentiment Score"][i]
13         average_scores.append(average)
14         average_alias.append(round(average))
```

Adding to dataframe and comparing values to find the best suited

```
In [22]: 1 LABELLED_TWEETS_DF["Average Sentiment Score"] = average_scores
2 LABELLED_TWEETS_DF["Average Sentiment Label"] = average_alias
3
4 print("COMPARISON OF AVERAGE WITH VADER:",
5 compare_series_values(
6     LABELLED_TWEETS_DF["Vader Sentiment Label"],
7     LABELLED_TWEETS_DF["Average Sentiment Label"]
8 ))
```

COMPARISON OF AVERAGE WITH VADER: {'same': 6438, 'different': 175, 'score': '6438/6613'}

General Comparison

In [30]:

```

1 TWEETS_HUMAN_LABELLED_DF=pd.read_csv("data/tweets_human_labelled.csv")
2
3
4 print("COMPARISON OF VADER WITH HUMAN1:",
5 compare_series_values(
6     TWEETS_HUMAN_LABELLED_DF["Vader Sentiment Label"][:102],
7     TWEETS_HUMAN_LABELLED_DF["Human1 Sentiment Label"][:102]
8 ))
9
10 print("COMPARISON OF TEXTBLOB WITH HUMAN1:",
11 compare_series_values(
12     TWEETS_HUMAN_LABELLED_DF["TextBlob Sentiment Label"][:102],
13     TWEETS_HUMAN_LABELLED_DF["Human1 Sentiment Label"][:102]
14 ))
15
16 print("COMPARISON OF VADER WITH HUMAN1:",
17 compare_series_values(
18     TWEETS_HUMAN_LABELLED_DF["Vader Sentiment Label"][:102],
19     TWEETS_HUMAN_LABELLED_DF["Human2 Sentiment Label"][:102]
20 ))
21
22 print("COMPARISON OF TEXTBLOB WITH HUMAN1:",
23 compare_series_values(
24     TWEETS_HUMAN_LABELLED_DF["TextBlob Sentiment Label"][:102],
25     TWEETS_HUMAN_LABELLED_DF["Human2 Sentiment Label"][:102]
26 ))

```

COMPARISON OF VADER WITH HUMAN1: {'same': 75, 'different': 27, 'score': '5/102'}

COMPARISON OF TEXTBLOB WITH HUMAN1: {'same': 42, 'different': 60, 'score': '42/102'}

COMPARISON OF VADER WITH HUMAN1: {'same': 70, 'different': 32, 'score': '70/102'}

COMPARISON OF TEXTBLOB WITH HUMAN1: {'same': 41, 'different': 61, 'score': '41/102'}

In [23]:

```
1 LABELLED_TWEETS_DF.head()
```

Out[23]:

	Tweet Body	TextBlob Sentiment Score	TextBlob Sentiment Label	Vader Sentiment Score	Vader Sentiment Label	Emoji Sentiment Score	Emoji Sentiment Label	A'
0	Its #Expo2020 Day	0.00000	0	0.0000	0	-2.000000	-2	0
1	We celebrated the National Day of the Slovak R...	0.61875	1	0.9531	1	-2.000000	-2	0
2	Vertebral Deformity Measurements on MRI, CT, a...	0.00000	0	0.0000	0	-2.000000	-2	0
3	🚨 #BREAKING An important statement by the 🇿🇦 Y...	0.15000	0	0.2023	0	0.646261	1	0
4	@Tourism_gov_za @LindiweSisuluSA @expo2020duba...	-0.17500	0	-0.7712	-1	-2.000000	-2	-0

Saving to file

```
In [24]: 1 LABELLED_TWEETS_DF.to_csv("./data/tweets_labelled.csv", index=False)
```

Analyzing overall distribution of sentiments for Vader and TextBlob

```
In [25]: 1 TWEETS_HUMAN_LABELLED_DF=pd.read_csv("data/tweets_human_labelled.csv")
2 print(TWEETS_HUMAN_LABELLED_DF['Vader Sentiment Label'].value_counts())
3 print(TWEETS_HUMAN_LABELLED_DF['TextBlob Sentiment Label'].value_counts())
```

```
0    587
1    442
-1     15
Name: Vader Sentiment Label, dtype: int64
0    951
1     89
-1      4
Name: TextBlob Sentiment Label, dtype: int64
```

It can be observed that the tweets are unevenly distributed in general. This must be handled in upcoming stages of the project.