Sentiment & Emotion Analysis

BAMP 2022 - MCT 4 - Ante Jelavic, Franziskus Perkhofer, Manuel Mencher, Melissa Ewering, Tim Ritzheimer

Short description: This script is based on the data (tweets) collected in the script "DataCollectionTweetsPerUser". The purpose of this script is to analyze the sentiments as well as the emotions of these tweets using Transformer based Deep Neutral Networks. First, the input data is checked and cleaned - the same is done with the output data after the analysis to prepare it for further analysis in the script "VisualizationOfResults".

Since the data is not labeled the analysis is based on pre-trained deep neural network transformer models from Huggingface. More concretely, two models have been used:

1. siebert/sentiment-roberta-large-english (Heitmann et al. 2020)

- Paper link: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3489963
- Huggingface link: https://huggingface.co/siebert/sentiment-roberta-large-english

> @article{heitmann2020, title={More than a feeling: Benchmarks for sentiment analysis accuracy}, author={Heitmann, Mark and Siebert, Christian and Hartmann, Jochen and Schamp, Christina}, journal={Available at SSRN 3489963}, year={2020}
> }

2. j-hartmann/emotion-english-roberta-large (Hartmann, 2022)

- Reference: Jochen Hartmann, "Emotion English DistilRoBERTa-base". https://huggingface.co/j-hartmann/emotion-english-distilroberta-base/, 2022.

> @misc{hartmann2022emotionenglish, author={Hartmann, Jochen}, title={Emotion English DistilRoBERTa-base}, year={2022}, howpublished = {\url{https://huggingface.co/j-hartmann/emotion-english-distilroberta-base/}}, }

Both Transformer models are fine-tuned checkpoints of RoBERTa-large (Liu et al. 2019) - Paper link: https://arxiv.org/pdf/1907.11692.pdf. The respective script sections are largely based on the documentation and scripts provided by the authors.

For performance reasons, this script was created and executed on Google Colab and then saved as an .ipynb and .pdf file. Therefore, all input and output files are stored on the connected Google Drive account and were afterwards transferred. In order to be able to run this script locally, it may be necessary to make adjustments to the dependencies / loaded packages.

```
# Loading packages & dependencies

# For dealing with json responses we receive from the API
```

```
import json
# For displaying the data after
import pandas as pd
# For saving the response data in CSV format
import csv
# For parsing the dates received from twitter in readable formats
import datetime
import dateutil.parser
import unicodedata
#To add wait time between requests
import time
#enable downloading output files from colab environment
from google.colab import files
```

Data Preparation before Analysis

This part will load the data (tweets) and perform a simple analysis and cleaning activities to prepare the dataset for sentiment & emotion analysis.

```
#Import data from csv and xls files stored on Google Drive

from google.colab import drive
drive.mount('/content/drive')



file_name_1 = "/content/drive/MyDrive/Colab Notebooks/Tweets.csv"
file_name_2 = "/content/drive/MyDrive/Colab Notebooks/Demographics.xlsx"

rawTweets = pd.read_csv(file_name_1) # Tweets.csv (see Output_Data)
demographics = pd.read_excel(file_name_2) #Demographics.xlsx (see Input_Data)
```

```
Mounted at /content/drive
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882:
  exec(code_obj, self.user_global_ns, self.user_ns)
```

```
#Check the number of twets per user and per language and save to csv
pd.set_option('display.max_rows', 102)
pd.set_option('display.max_columns', 50)
crosstab = pd.crosstab(rawTweets['author_id'], rawTweets['lang'])
crosstab.to_csv("crosstab.csv") # (see Output_Data)
crosstab
```

lang	am	ar	ca	cs	cy	da	de	el	en	es	et	eu	fa	fi	fr	hi	ht
author_id																	
1.294741e+06	0	0	1	1	3	0	2	0	3193	4	0	0	0	0	4	0	3
5.715682e+06	0	0	0	0	0	1	1	0	3190	0	0	0	0	0	1	0	1
6.612402e+06	0	0	2	1	4	1	5	0	3012	9	3	1	0	0	12	0	3
6.705042e+06	0	0	0	0	0	0	3	0	3154	8	0	0	0	0	7	1	1
8.161232e+06	0	0	0	0	0	0	0	0	3223	1	1	0	0	0	2	0	0
9.950972e+06	0	0	0	2	1	0	0	0	764	2	1	1	0	0	0	0	0
1.235483e+07	0	0	4	0	1	0	9	0	2958	3	3	0	0	0	3	0	1
1.245530e+07	0	0	5	0	2	2	28	0	2856	16	7	1	0	0	53	0	2
1.415713e+07	0	0	0	0	0	0	0	0	3230	1	0	0	0	0	1	0	0
1.468060e+07	0	0	0	0	0	1	1	0	1253	15	4	0	0	8	3	0	0
1.514348e+07	0	2	1	0	0	0	3	0	2163	1	0	0	1	0	3	0	0
1.543940e+07	0	0	0	0	0	0	3	0	2131	1	2	0	0	0	2	0	0
1.639995e+07	0	0	1	1	1	2	7	0	3071	4	0	1	0	0	4	0	3
1.668111e+07	0	0	2	0	0	2	3	0	3173	3	0	0	0	0	0	0	0
1.693573e+07	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0
1.702096e+07	0	1	0	0	0	0	0	0	894	0	0	0	0	0	0	0	0
1.706236e+07	0	0	1	0	1	1	0	0	3157	2	0	0	0	0	2	0	0
1.775303e+07	0	0	0	0	0	0	1	0	1238	7	1	1	0	0	7	0	1
1.873097e+07	0	0	0	0	0	0	0	0	218	0	0	0	0	0	0	0	0
1.876484e+07	0	0	6	0	1	0	2	0	2975	1	0	1	0	0	12	0	1
1.888753e+07	0	0	0	0	0	0	0	0	646	0	0	0	0	0	0	0	0
1.895326e+07	0	0	0	0	0	0	1	0	1367	6	2	0	0	0	2	0	1
1.911177e+07	0	0	0	0	1	0	1	0	1038	0	1	0	0	0	1	0	0
1.933538e+07	0	0	0	0	0	0	0	0	78	0	0	0	0	0	0	0	0
1.934644e+07	0	0	0	0	3	1	4	0	3015	2	1	0	0	0	7	0	4
1.953029e+07	0	0	0	0	1	0	1	1	3148	1	1	0	0	0	2	0	0
1.953487e+07	0	0	0	0	2	0	3	0	3081	9	0	0	0	0	6	0	0
1.958838e+07	0	0	0	0	1	0	0	0	299	0	0	0	0	0	1	0	0
1.964459e+07	0	0	0	0	1	0	3	0	3122	7	0	0	0	0	18	0	0
1.966087e+07	0	0	2	0	4	2	1	0	3019	6	2	0	0	0	24	0	1
1.972564e+07	0	0	1	0	0	0	0	0	1644	3	0	0	0	0	0	0	0
1.981119e+07	0	1	3	0	2	2	23	4	3096	21	0	1	0	1	14	0	2

[illegible]	0	1	0	0	2	2	23	4	[illegible]	21	0	1	0	1	14	0	2
1.981849e+07	0	0	0	0	0	0	0	0	409	0	0	0	0	0	0	0	0
1.990271e+07	0	0	0	0	1	0	0	0	3175	0	0	0	0	0	0	0	1
2.001531e+07	0	0	0	0	1	1	2	0	2732	4	0	1	0	0	7	0	2
2.009441e+07	0	0	0	0	0	0	3	0	3157	6	1	1	0	0	5	0	0
2.013762e+07	0	0	0	0	0	0	0	0	818	0	0	0	0	0	0	0	1
2.015565e+07	0	0	0	0	2	0	5	0	843	0	1	0	0	0	4	0	1
2.024488e+07	0	0	0	0	0	0	3	0	1876	2	5	0	0	0	1	1	0
2.063270e+07	0	0	0	0	0	0	0	0	1089	0	4	0	0	0	0	0	0
2.068594e+07	0	0	2	1	1	1	2	0	3046	7	1	1	0	0	9	0	2
2.202198e+07	0	0	0	0	2	0	1	0	3045	2	2	0	0	0	2	0	1
2.281273e+07	0	0	0	0	0	0	0	0	3222	2	1	0	0	0	2	0	0
2.444764e+07	0	0	0	0	1	0	4	0	1144	0	0	0	0	0	3	0	0
2.548925e+07	0	0	0	0	0	0	0	0	1841	0	0	0	0	0	4	0	0
2.565740e+07	0	0	0	0	0	0	0	0	1649	2	0	0	0	0	1	0	0
3.330025e+07	0	0	0	0	0	0	2	0	3174	0	0	0	0	0	0	0	0
3.353797e+07	0	0	1	0	0	0	1	0	3235	2	0	0	0	0	2	0	0
3.355280e+07	0	0	0	0	0	0	0	0	308	1	0	0	0	0	0	0	0
4.410141e+07	0	0	1	0	0	1	3	0	2646	4	0	1	0	0	1	0	3
4.457833e+07	0	0	1	1	0	0	6	0	3154	4	1	0	0	0	3	0	0
4.487440e+07	0	0	2	0	2	0	4	0	3120	7	1	0	0	2	5	0	0
4.534157e+07	0	0	0	0	0	0	1	0	3167	2	1	0	0	0	8	0	0
4.695904e+07	0	0	1	3	1	0	3	0	3071	4	1	3	0	1	9	0	3
6.118357e+07	0	0	0	0	0	1	0	0	3143	1	0	0	0	0	2	0	0
6.504512e+07	0	0	0	0	1	0	0	0	3110	4	0	1	0	0	2	0	2
8.012703e+07	0	0	0	0	0	0	0	0	232	0	1	0	0	0	0	0	0
8.385592e+07	0	0	0	0	0	0	0	0	271	1	1	0	0	0	0	1	0
8.435123e+07	0	0	0	0	0	0	0	0	130	0	0	0	0	0	0	0	0
8.666492e+07	0	0	1	0	1	0	1	0	3176	5	0	0	0	0	9	0	1
9.462830e+07	0	0	0	0	0	0	0	0	586	0	2	0	0	0	2	0	0
9.620528e+07	0	0	0	0	1	0	0	0	3019	4	2	1	0	0	6	0	1
1.173664e+08	0	0	1	0	3	0	10	0	3175	2	0	3	0	1	10	0	1
1.282169e+08	0	0	2	0	0	0	1	0	1853	1	0	0	0	0	1	0	0
1.305575e+08	0	0	0	0	0	1	2	0	3027	7	0	0	0	0	5	1	1

1.506321e+08	0	0	1	0	0	1	1	0	1583	7	0	2	0	0	4	0	0
1.538102e+08	0	0	0	0	0	0	0	0	3205	4	1	0	0	0	7	0	0
1.559280e+08	0	0	0	1	2	4	3	0	2202	27	12	1	0	1	7	0	0
1.602727e+08	0	0	0	0	2	0	0	0	232	2	0	0	0	0	0	0	0
1.609269e+08	0	0	1	0	0	1	2	0	3076	7	2	0	0	0	0	0	3
1.625017e+08	0	0	0	0	1	0	0	0	3016	12	1	0	0	0	13	0	1
1.667679e+08	0	0	0	0	1	1	1	0	1054	4	2	1	0	0	6	0	0
1.903713e+08	0	0	1	1	0	0	1	0	651	4	0	0	0	0	2	0	2
1.937322e+08	0	0	0	0	0	0	0	0	2606	0	0	0	0	0	0	0	0
1.985848e+08	0	0	0	0	1	0	1	0	3190	2	1	0	0	0	3	0	0
2.211600e+08	0	0	0	0	0	0	0	0	334	1	1	0	0	0	0	0	0
2.379633e+08	0	0	0	0	0	0	0	0	263	0	0	0	0	0	0	0	0
2.625025e+08	0	0	0	0	0	0	0	0	547	4	1	0	0	0	29	0	0
2.878346e+08	0	0	3	0	2	1	1	0	2952	7	2	1	0	1	5	0	1
3.261843e+08	0	0	0	1	0	1	0	0	3060	2	1	0	0	0	5	0	0
3.286346e+08	0	0	0	0	0	0	0	0	3214	0	0	0	0	0	1	0	0
3.734162e+08	0	0	0	0	1	0	0	0	507	3	0	0	0	0	5	0	0
3.753830e+08	0	0	0	0	0	0	0	0	286	0	1	0	0	0	0	0	0
3.866545e+08	0	0	0	0	0	0	2	0	2314	5	0	1	0	0	7	0	1
4.108115e+08	0	0	0	0	1	1	5	0	2531	9	5	1	0	0	7	1	2
4.540710e+08	0	0	2	0	1	0	1	0	2989	5	1	1	0	0	13	0	1
4.982048e+08	0	0	0	0	0	0	0	0	239	0	0	0	0	0	1	0	0
5.483845e+08	0	0	1	0	0	2	3	0	2981	4	3	0	0	0	2	0	0
6.029938e+08	0	0	4	0	0	0	0	0	1489	8	4	4	0	0	0	3	1
7.435387e+08	0	0	0	0	0	0	1	0	284	0	0	0	0	0	0	0	0
7.450026e+08	0	0	0	0	0	0	1	0	216	1	0	0	0	0	0	0	0
7.837930e+08	0	2	0	0	1	0	0	0	3043	4	0	0	0	0	1	2	1
9.374992e+08	1	0	0	0	0	0	0	0	328	0	0	0	1	0	3	0	0
1.013399e+09	0	0	0	0	0	0	0	0	157	0	0	0	0	0	0	0	0
1.700542e+09	0	0	1	0	0	0	0	0	1660	0	1	0	0	0	1	0	0

Conclusion: 198430 tweets are in english. Not any profile did not tweet in english. The 3 user with smallest amount of english posts did 13, 78 andf 84 posts. The user with the highest amount of english posts did approx. 3200 posts (or even more as 3200 is the limit of possible tweets which can be retrieved by the twitter API per user). The second most used language is undefined with 10307 tweets and afterwards german with 188 tweets. Based on the low amount of tweets not in english, it was decided to drop all tweets not in english to simplify the following analysis.

```
# drop all tweets not in english
enTweets = rawTweets.drop(rawTweets[rawTweets["lang"] != "en"].index)
len(enTweets)
```

```
    198430
```

Analysis of hashtags (hashtags indicate that a post belongs to a certain topic)

```
# Function to extract the hashtags from text s
def extract_hash_tags(s):
  return set(part[1:] for part in s.split() if part.startswith('#'))
```

```
# Extract all hashtags and save in seperate column
extracted_hashtags = []
for x in range(len(enTweets["text"])):
    hashtags = list(extract_hash_tags(enTweets["text"].iloc[x]))
    extracted_hashtags.append(hashtags)

enTweets["Hashtags"] = extracted_hashtags
enTweets.tail()
```

	Unnamed: 0	referenced_tweets	text	author_id	created_
211238	10	NaN	Loved doing this shoot with @InfrarougeMag!#Wi...	262502487.0	2019- 05T12:36:57.0
211239	11	NaN	Have you been to see it yet? #TheEmperorOfParis...	262502487.0	2019- 04T14:10:05.0
			When you realise the		

```
# add unique ID to dataset to enable bettter over
enTweets.insert(0, 'Unique_ID', range(0, len(enTweets)))
```

```
# save a new table with each hashtags assigned to a Unique_ID (realted to tweets)
hash_df = pd.DataFrame(columns=['Unique_ID', 'Hashtag'])
i = 0
for x in range(len(enTweets["Unique_ID"])):
    for y in range(len(enTweets["Hashtags"].iloc[x])):
        #data = pd.DataFrame({"Unique_ID": x, "Hashtag": enTweets["Hashtags"].iloc[
        to_append = [x, enTweets["Hashtags"].iloc[x][y]]
        hash_df.loc[len(hash_df)] = to_append

hash_df.to_csv("Hashtags.csv") # see Output_Data
```

```
hash_df['Hashtag'].value_counts()[:30] #show top 30 most frequent hashtags
```

```
MakeHumanityGreatAgain    558
COVID19                   519
Ad                        432
TEAMSM                    432
VirginFamily              368
TheApprentice             361
TomorrowsPapersToday      323
JoinIn                    300
SistersInLaw              276
COP26                     224
ESG                       220
100bookshops              199
WayTooEarly               195
AfterLife                 190
ad                        176
Brexit                    161
StopBrexit                152
SuperNature               152
coronavirus               125
BorisJohnson              123
BlackLivesMatter          118
thecroonersessions        105
DOOH                       98
Peston                     97
PMQs                       97
MusicPlayedByHumans        96
100Bookshops               95
celebrityApprentice        95
NHS                        88
5GoldRings                 86
Name: Hashtag, dtype: int64
```

The following topic groups seem to be relevant for a large user group and will be therefore analyzed more deeply to show an example:

- COVID (e.g. hashtags: #COVID19, #coronaviruse, #corona,...)
- Brexit (e.g. hashtags: #Brexit, #StopBrecit, ...)
- ESG (e.g. hashtags: #COP26, #ESG, ...)

In the next steps we will analze the hastags and cluser different wrintings and synonyms to those 3 groups. This is manual work, however, the other use-case (topic-modelling) of this BAMP will show a way how to cover this task automatically.

```
# Function to iterate over the hashtags and check for related searchterms in order
def checkHashtag (hashtag_series, searchterm_series):
  hashtag_found = []
  for hashtag_list in hashtag_series:
    val = 0
    for hashtag in hashtag_list:
      for searchterm in searchterm_series:
        if searchterm.lower() in hashtag.lower():
```

```
        val = 1
      hashtag_found.append(val)
    return hashtag_found


# Search for topic groups by hashtag and save those into the tweets dataset
enTweets['covid_hashtags'] = checkHashtag(enTweets["Hashtags"], ['covid', 'corona',
enTweets['brexit_hashtags'] = checkHashtag(enTweets["Hashtags"], ['brexit'])
enTweets['esg_hashtags'] = checkHashtag(enTweets["Hashtags"], ['esg', 'sustainabili

enTweets.to_csv("enTweetsNew.csv") # see Output_Data
enTweets.head()
```

	Unique_ID	Unnamed: 0	referenced_tweets	text	author_id	created_a
0	0	0	[{'type': 'retweeted', 'id': '1476916508265783...	RT @TheElders: "True peace is never won by dip...	8161232.0	2021-12-31T15:38:31.000
				My thoughts on COVID		2021-12

```
# Download output files to save locally
#files.download('Hashtags.csv')
#files.download('enTweetsNew.csv')
```

Sentiment Analysis

This section will perform the sentiment analysis based on a pretrained transformer model.

```
# Import required packages and transformers libary
import torch
import pandas as pd
import numpy as np
!pip install transformers
from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer
```

```
Collecting transformers
  Downloading transformers-4.16.2-py3-none-any.whl (3.5 MB)
     |████████████████████████████████| 3.5 MB 4.2 MB/s
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-pa
Collecting sacremoses
  Downloading sacremoses-0.0.47-py2.py3-none-any.whl (895 kB)
     |████████████████████████████████| 895 kB 55.4 MB/s
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.7/c
Collecting tokenizers!=0.11.3,>=0.10.1
  Downloading tokenizers-0.11.6-cp37-cp37m-manylinux_2_12_x86_64.manylinux201C
     |████████████████████████████████| 6.5 MB 51.7 MB/s
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/
Collecting pyyaml>=5.1
```

```
  Downloading PyYAML-6.0-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.man
     |████████████████████████████████| 596 kB 76.3 MB/s
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packa
Collecting huggingface-hub<1.0,>=0.1.0
  Downloading huggingface_hub-0.4.0-py3-none-any.whl (67 kB)
     |████████████████████████████████| 67 kB 7.3 MB/s
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dis
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/py
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/pyth
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/d
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-package
Installing collected packages: pyyaml, tokenizers, sacremoses, huggingface-hub
  Attempting uninstall: pyyaml
    Found existing installation: PyYAML 3.13
    Uninstalling PyYAML-3.13:
      Successfully uninstalled PyYAML-3.13
Successfully installed huggingface-hub-0.4.0 pyyaml-6.0 sacremoses-0.0.47 toke
```

```
# Create class for data preparation
class SimpleDataset:
    def __init__(self, tokenized_texts):
        self.tokenized_texts = tokenized_texts

    def __len__(self):
        return len(self.tokenized_texts["input_ids"])

    def __getitem__(self, idx):
        return {k: v[idx] for k, v in self.tokenized_texts.items()}


# Load tokenizer and model, create trainer
model_name = "siebert/sentiment-roberta-large-english"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSequenceClassification.from_pretrained(model_name)
trainer = Trainer(model=model)
```

```
Downloading: 100%                                  256/256 [00:00<00:00, 7.67kB/s]

Downloading: 100%                                  687/687 [00:00<00:00, 30.0kB/s]

Downloading: 100%                                  780k/780k [00:00<00:00, 621kB/s]

Downloading: 100%                                  446k/446k [00:00<00:00, 656kB/s]

Downloading: 100%                                  150/150 [00:00<00:00, 3.77kB/s]

Downloading: 100%                                  1.32G/1.32G [00:23<00:00, 66.0MB/s]
```

```
# Tokenize texts and create prediction data set
```

```
pred_texts = enTweets["text"].dropna().astype('str').tolist()
tokenized_texts = tokenizer(pred_texts,truncation=True,padding=True)
pred_dataset = SimpleDataset(tokenized_texts)


# Run predictions
predictions = trainer.predict(pred_dataset)
```

```
***** Running Prediction *****
  Num examples = 198430
  Batch size = 8
                                    [21728/24804 2:01:18 < 17:10, 2.99 it/s]
                                    [24804/24804 2:18:28]
```

```
# Transform predictions to labels
preds = predictions.predictions.argmax(-1)
labels = pd.Series(preds).map(model.config.id2label)
scores = (np.exp(predictions[0])/np.exp(predictions[0]).sum(-1,keepdims=True)).max(


# Create DataFrame with texts, predictions, labels, and scores
df_results = pd.DataFrame(list(zip(pred_texts,preds,labels,scores)), columns=['text
df_results.insert(0, 'Unique_ID', range(0, len(df_results)))
df_results.head()
```

	Unique_ID	text	pred	label	score
0	0	RT @TheElders: "True peace is never won by dip...	1	POSITIVE	0.998705
1	1	My thoughts on COVID and its effects on younge...	1	POSITIVE	0.997372
2	2	Thank you Arch for your love, life, laughter a...	1	POSITIVE	0.998721
3	3	I'm so sad that Archbishop Tutu has passed awa...	0	NEGATIVE	0.996188
4	4	Happy Christmas from my family to yours. https...	1	POSITIVE	0.998654

Emotion Analysis

This section will perform the emotion analysis based on a pretrained transformer model. As the transformer below is very similiar to the one above used for sentiment analysis, it reuses certain code sections and variables. Therefore, please make sure to run first the section of sentiment analysis and only afterwards the section Emotion Analysis

```
# load tokenizer and model, create trainer
model_name = "j-hartmann/emotion-english-roberta-large"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSequenceClassification.from_pretrained(model_name)
trainer = Trainer(model=model)
```

```
https://huggingface.co/j-hartmann/emotion-english-roberta-large/resolve/main/t
Downloading: 100%                                   328/328 [00:00<00:00, 11.7kB/s]
storing https://huggingface.co/j-hartmann/emotion-english-roberta-large/resolv
creating metadata file for /root/.cache/huggingface/transformers/f039b3b4df1ee
https://huggingface.co/j-hartmann/emotion-english-roberta-large/resolve/main/v
Downloading: 100%                                   780k/780k [00:00<00:00, 641kB/s]
storing https://huggingface.co/j-hartmann/emotion-english-roberta-large/resolv
creating metadata file for /root/.cache/huggingface/transformers/89a81a716bd56
https://huggingface.co/j-hartmann/emotion-english-roberta-large/resolve/main/m
Downloading: 100%                                   446k/446k [00:00<00:00, 636kB/s]
storing https://huggingface.co/j-hartmann/emotion-english-roberta-large/resolv
creating metadata file for /root/.cache/huggingface/transformers/3653ca003db05
https://huggingface.co/j-hartmann/emotion-english-roberta-large/resolve/main/t
Downloading: 100%                                   1.29M/1.29M [00:01<00:00, 1.05MB/s]
storing https://huggingface.co/j-hartmann/emotion-english-roberta-large/resolv
creating metadata file for /root/.cache/huggingface/transformers/aef4a5624ed9e
https://huggingface.co/j-hartmann/emotion-english-roberta-large/resolve/main/s
Downloading: 100%                                   239/239 [00:00<00:00, 9.90kB/s]
storing https://huggingface.co/j-hartmann/emotion-english-roberta-large/resolv
creating metadata file for /root/.cache/huggingface/transformers/11f22a314e072
loading file https://huggingface.co/j-hartmann/emotion-english-roberta-large/r
loading file https://huggingface.co/j-hartmann/emotion-english-roberta-large/r
loading file https://huggingface.co/j-hartmann/emotion-english-roberta-large/r
loading file https://huggingface.co/j-hartmann/emotion-english-roberta-large/r
loading file https://huggingface.co/j-hartmann/emotion-english-roberta-large/r
loading file https://huggingface.co/j-hartmann/emotion-english-roberta-large/r
https://huggingface.co/j-hartmann/emotion-english-roberta-large/resolve/main/c
Downloading: 100%                                   1.01k/1.01k [00:00<00:00, 42.4kB/s]
storing https://huggingface.co/j-hartmann/emotion-english-roberta-large/resolv
creating metadata file for /root/.cache/huggingface/transformers/2bfc7aed293b5
loading configuration file https://huggingface.co/j-hartmann/emotion-english-r
Model config RobertaConfig {
  "_name_or_path": "j-hartmann/emotion-english-roberta-large",
  "architectures": [
    "RobertaForSequenceClassification"
  ],
  "attention_probs_dropout_prob": 0.1,
  "bos_token_id": 0,
  "classifier_dropout": null,
  "eos_token_id": 2,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 1024,
  "id2label": {
    "0": "anger",
    "1": "disgust",
    "2": "fear",
    "3": "joy",
    "4": "neutral",
    "5": "sadness",
    "6": "surprise"
  },
  "initializer_range": 0.02,
  "intermediate_size": 4096,
  "label2id": {
    "anger": 0,
    "disgust": 1,
    "fear": 2,
    "joy": 3,
    "neutral": 4,
    "sadness": 5,
    "surprise": 6
  },
  "layer_norm_eps": 1e-05,
  "max_position_embeddings": 514,
  "model_type": "roberta",
  "num_attention_heads": 16,
  "num_hidden_layers": 24,
  "pad_token_id": 1,
  "position_embedding_type": "absolute",
  "problem_type": "single_label_classification",
  "torch_dtype": "float32",
  "transformers_version": "4.16.2",
  "type_vocab_size": 1,
  "use_cache": true,
  "vocab_size": 50265
}

https://huggingface.co/j-hartmann/emotion-english-roberta-large/resolve/main/p
Downloading: 100%                                1.32G/1.32G [01:11<00:00, 18.9MB/s]

storing https://huggingface.co/j-hartmann/emotion-english-roberta-large/resolv
creating metadata file for /root/.cache/huggingface/transformers/158746e237965
loading weights file https://huggingface.co/j-hartmann/emotion-english-roberta
All model checkpoint weights were used when initializing RobertaForSequenceCla

All the weights of RobertaForSequenceClassification were initialized from the
If your task is similar to the task the model of the checkpoint was trained or
No `TrainingArguments` passed, using `output_dir=tmp_trainer`
```

```
# Run predictions
predictions = trainer.predict(pred_dataset)
```

```
***** Running Prediction *****
  Num examples = 198430
  Batch size = 8
[22572/24804 2:05:59 < 12:27, 2.99 it/s]
[24804/24804 2:18:27]
```

```
# Transform predictions to labels
preds = predictions.predictions.argmax(-1)
labels = pd.Series(preds).map(model.config.id2label)
scores = (np.exp(predictions[0])/np.exp(predictions[0]).sum(-1,keepdims=True)).max(


# scores raw
temp = (np.exp(predictions[0])/np.exp(predictions[0]).sum(-1,keepdims=True))


# container
anger = []
disgust = []
fear = []
```

```
joy = []
neutral = []
sadness = []
surprise = []

# extract scores (as many entries as exist in pred_texts)
for i in range(len(pred_texts)):
  anger.append(temp[i][0])
  disgust.append(temp[i][1])
  fear.append(temp[i][2])
  joy.append(temp[i][3])
  neutral.append(temp[i][4])
  sadness.append(temp[i][5])
  surprise.append(temp[i][6])


# Create DataFrame with texts, predictions, labels, and scores
df_restuls_emotions = pd.DataFrame(list(zip(pred_texts,preds,labels,scores,  anger,
df_restuls_emotions.insert(0, 'Unique_ID', range(0, len(df_restuls_emotions)))
df_restuls_emotions.head()
```

	Unique_ID	text	pred	emotion_label	score	anger	disgust	fea
0	0	RT @TheElders: "True peace is never won by dip...	2	fear	0.318116	0.264952	0.000666	0.3181
1	1	My thoughts on COVID and its	4	neutral	0.845129	0.013473	0.011618	0.0202

Data combination and storage

This section will combine all the results / data collected in different data frames and merge all relevant data into one file which will contain all relevant data for visualization of the results.

```
# Check that results and tweet data has the same length
print(len(enTweets))
print(len(df_results))
print(len(df_restuls_emotions))
```

```
198430
198430
198430
```

```
# Merge sentiment and emotion analysis results with tweets
merged_df1 = pd.merge(enTweets, df_results, on="Unique_ID") #Merge sentiment result
merged_df2 = pd.merge(merged_df1, df_restuls_emotions, on="Unique_ID") #Merge emoti
merged_df2.head()
```

	Unique_ID	Unnamed: 0	referenced_tweets	text_x	author_id	created_a
0	0	0	[{'type': 'retweeted', 'id': '1476916508265783...	RT @TheElders: "True peace is never won by dip...	8161232.0	2021-12 31T15:38:31.000
1	1	1	NaN	My thoughts on COVID and its effects on younge...	8161232.0	2021-12 27T10:00:18.000
2	2	2	NaN	Thank you Arch for your love, life, laughter a...	8161232.0	2021-12 26T10:00:04.000
3	3	3	NaN	I’m so sad that Archbishop Tutu has passed awa...	8161232.0	2021-12 26T08:09:55.000
4	4	4	NaN	Happy Christmas from my family to yours. https...	8161232.0	2021-12 25T09:23:01.000

```
# Merge demographics into above merged dataframe
demographics = demographics.rename(columns={"ID": "author_id"}) # rename to be able
merged_df3 = pd.merge(merged_df2, demographics, on="author_id") # merge demographic
merged_df3.tail()
```

	Unique_ID	Unnamed: 0	referenced_tweets	text_x	author_id

```
# Clean out unnecessary columns
merged_df3.drop(['Unnamed: 0','text','text_y'], axis=1, inplace=True)
merged_df3 = merged_df3.rename(columns={"text_x": "text"})
merged_df3.head()
```

	Unique_ID	referenced_tweets	text	author_id	created_at	
0	0	[{'type': 'retweeted', 'id': '1476916508265783...	RT @TheElders: "True peace is never won by dip...	8161232.0	2021-12-31T15:38:31.000Z	14769408
1	1	NaN	My thoughts on COVID and its effects on younge...	8161232.0	2021-12-27T10:00:18.000Z	14754061
2	2	NaN	Thank you Arch for your love, life, laughter a...	8161232.0	2021-12-26T10:00:04.000Z	14750437
3	3	NaN	I’m so sad that Archbishop Tutu has passed awa...	8161232.0	2021-12-26T08:09:55.000Z	14750160
4	4	NaN	Happy Christmas from my family to yours. https...	8161232.0	2021-12-25T09:23:01.000Z	14746720

```
# Save to csv
merged_df3.to_csv("FinalResults.csv", index = False)
files.download('FinalResults.csv')
```

```
# Check that FinalResults and tweet data has the same length
print(len(enTweets))
print(len(merged_df3))
```

```
198430
198430
```

● ✕