🤨 How do people feel about Hasanabi?

A little of backstory first. I'm a 2nd year uni student on a computer science course. One of my classes is computational intelligence, where we are introduced to the world of machine learning algorithms and ai. Our last assignment of the semester is to use a Twitter scraper and analyze feelings of people engaging in some topic. I wanted to make this interesting so I chose something that would make me finish this. When I asked my friend on suggestions what should I do he suggested me Hasan.

He is a popular twitch streamer, internet personality, ¿socialist activist? (for sure someone with socialistic views). He has an online following and is overall liked by the internet crowd, I too engaged with his works (mainly entertainment). Because of his political and economic views he could be named someone that raises a lot of noise and makes people feel a certain way so he is a perfect match for a subject of this project! He also raises a lot positive emotions since he is an entertainer.

What we will do is we will download a lot of Mr. Piker's tweets and people's comments under those posts. We will put those comments through language analyzing algorithms and see how people feel about his political views.

DISCLAIMER: Before we continue I'd like to add that this is mainly for entertainment and you SHOULD NOT base any thoughts on it. You should remember that this was made by a uni CS student. ML (machine learning) is also something flawed:

- it generates slightly different outcomes every time it's run
- is based on math
- is something problematic because it's hard to explain a computer how does "feelings" and human language work

But we can laugh about what will turn out of this. Maybe I could even get Hasan to show it on stream? So without further ado let's continue!

In [ ]:

```
# Importing python libraries nothing special
import pandas as pd
import snscrape.modules.twitter as sntwitter
from snscrape.modules.twitter import Tweet
from datetime import datetime
from datetime import timedelta
from typing import Union
import nltk
from dateutil.parser import parse as date_parse
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import matplotlib.pyplot as plt
import text2emotion as te
from wordcloud import WordCloud
import json
import html
import re
```

Getting the data

We will now proceed to the code that will make a search request on twitter site and scrape everything we want from it's result. For that we will use snscraper's twitter module.

In [ ]:
```
# Saving Hasan's twitter handle for later use
HASAN_TWITTER_HANDLE = "hasanthehun"
```

In [ ]:
```
# Helper function that gets parameters from us parses it into a valid twitter api que
# gets tweets and returns them as a list
def get_hasan_tweets(start_date: datetime, end_date: datetime, num_tweets: int = -1)
    tweets: list[Tweet] = []

    query = f"from:{HASAN_TWITTER_HANDLE} since:{start_date.strftime('%Y-%m-%d')} unt
    for tweet in sntwitter.TwitterSearchScraper(query).get_items():
        tweets.append(tweet)

        if len(tweets) == num_tweets:
            break

    return tweets
```

Let's make a test request to twitter!

In [ ]:
```
tweets = get_hasan_tweets(
    date_parse('2023-03-05'), date_parse('2023-03-06'), num_tweets=5)
```

In [ ]:
```
tweets[0].rawContent
```

Out[ ]:
```
'@MoistCr1TiKaL 4 companies practically own their own territories and for some reason
we can’t hold them accountable despite the billions we have spent building and mainta
ining the infastructure they now own 🤔 '
```

Yay it works! Now for the less fun part.

Because, at the time of me writing this, Twitter has turned off, literally two days ago, a search filter on their site that is STILL in their API query reference in their docs. I cannot use conversation_id filter to search only for replies to a tweet. We need to get creative. (THANK YOU ELON!)

We will get every reply to hasan and then match them to their respectable "parent" tweet.

In [ ]:
```
def get_replies_to_hasan(start_date: datetime, end_date: datetime, num_tweets: int =
    replies: list[Tweet] = []

    query = f"to:{HASAN_TWITTER_HANDLE} since:{start_date.strftime('%Y-%m-%d')} until
    for tweet in sntwitter.TwitterSearchScraper(query).get_items():
        replies.append(tweet)

        if len(replies) == num_tweets:
            break

    return replies
```

In [ ]:
```
replies = get_replies_to_hasan(date_parse(
    '2023-03-05'), date_parse('2023-03-06'), num_tweets=5)
```

In [ ]:
```
for reply in replies[:5]:
    print(reply.rawContent)
```

```
@hasanthehun Why are you talking shit already the game isn't even out
@hasanthehun Good, it's about time
@hasanthehun I'm sad there won't be any more IRL Japan streams - loved them so much @
hasanthehun
@hasanthehun I stg the accent at the end racist as hell
@hasanthehun Marginalized populations? The alphabet bunch is calling all the shots in
the democrat party right now.
```

Our filters work. So now we're going to get every Hasan's tweet and every reply. It'll be a lot of data.

In [ ]:
```python
# Helper function that will get only the data we want from a tweet
# We don't want to save data that we won't be using
def parse_post(tweet: Tweet):
    parsed_tweet = {}
    parsed_tweet['id'] = tweet.id
    parsed_tweet['url'] = tweet.url
    parsed_tweet['date'] = tweet.date.strftime('%Y-%m-%d %H:%M:%S')
    parsed_tweet['rawContent'] = tweet.rawContent
    parsed_tweet['renderedContent'] = tweet.renderedContent
    parsed_tweet['conversationId'] = tweet.conversationId
    parsed_tweet['lang'] = tweet.lang
    parsed_tweet['inReplyToTweetId'] = tweet.inReplyToTweetId

    return parsed_tweet
```

In [ ]:
```python
tweets = get_hasan_tweets(date_parse('2020-01-01'), datetime.now())
```

In [ ]:
```python
# parsing tweets
parsed_tweets = []
for tweet in tweets:
    parsed_tweets.append(parse_post(tweet))


# Saving them to a json file
with open("hasan_tweets.json", "w") as file:
    json.dump([parsed_tweet for parsed_tweet in parsed_tweets], file)
```

In [ ]:
```python
replies = get_replies_to_hasan(date_parse('2020-01-01'), datetime.now())
```

In [ ]:
```python
# Parsing replies
parsed_replies = []
for reply in replies:
    parsed_replies.append(parse_post(reply))

# Saving them to a json file
with open("hasan_replies.json", "w") as file:
    json.dump([parsed_reply for parsed_reply in parsed_replies], file)
```

Out of that we got 11.7 thousand tweets from hasan (he do post a lot) and 84.5 thousand replies at him. This is not a humongous amount of data but is big enough for our experiments. I decided to get all of the tweets from 1st of January 2020 until 27th of May 2023 (the day of writing this). This should include many interesting events. COVID-19, BLM protests, US election and recent stuff, probably comments about current post COVID economic state of the US.

Because of those interesting events and hope that Hasan engaged in the conversation on those topics should stir up some conversations and most importantly extreme feelings (I'd like to remind you that we are still talking about twitter).

Our data also includes things like Hasan announcing that he will be live soon, replies to his work friends and other non-political topics. Those tweets should encourage people to reply in a positive way and this would make our data with a tendency to lean in to more positive feelings.

We will see what will turn out of this.

Our data look something like this:

```
{
  "id": 1662165779498795010, // id of the tweet
  "url": "https://twitter.com/hasanthehun/status/1662165779498795010", //
tweets url
  "date": "2023-05-26 18:36:18", // date in a string format
  "rawContent": "DEBT CEILING QUESTIONS ANSWERED W/ @ddayen + HOGWATCH...",
// raw content of the tweet it's normalized to ASCII
  "renderedContent": "DEBT CEILING QUESTIONS ANSWERED W/ @ddayen +
HOGWATCH...", // content how it's being showed in a browser
  "conversationId": 1662165779498795010, // id of the conversation, if it's
a normal tweet it is its id
  "lang": "en", // lang that has been detected by twitter
  "inReplyToTweetId": null // if it's a reply here it shows to which tweet
}
```

In [ ]:
```python
# Loading normalized tweets and replies
hasan_tweets = pd.read_json('hasan_tweets.json')

print(hasan_tweets.shape)
```

```
(11714, 8)
```

In [ ]:
```python
replies_to_hasan = pd.read_json('hasan_replies.json')

print(replies_to_hasan.shape)
```

```
(84488, 8)
```

Preprocessing

Before we continue we need to make a little bit of preprocessing. We need to remove any duplicates from the replies, replies with links (they might be bots promoting scam), and empty tweets.

In [ ]:
```python
# Removing duplicates
replies_to_hasan.drop_duplicates(subset=['id', "rawContent"], inplace=True)

# Removing tweets that contain @@@@@@@@ or http or https
replies_to_hasan = replies_to_hasan[~replies_to_hasan['rawContent'].str.contains(
    '@@@@@@@@')]
replies_to_hasan = replies_to_hasan[~replies_to_hasan['rawContent'].str.contains(
    'http://')]
replies_to_hasan = replies_to_hasan[~replies_to_hasan['rawContent'].str.contains(
    'https://')]

# Printing shape of the data frame (how much data we are left with)
print(replies_to_hasan.shape)
```

```
(70540, 8)
```

We also have to do something with html codes. What are they? You see everything on the web is written in html. This is a markup language that eases formatting text. Html has some that do some things in it but we also want to put them into the text. Those characters are "<", ">" and others like "/". So they are not encoded into html code, which is what we have in our data. What we have to do is map those symbols into ASCII.

Also we gotta do something with emojis. They also are represented in a string of characters like "\ud83d\ude14" which is translated to "😔 ".

In [ ]:
```
# This is a helper function that will remove all emojis from a string
def remove_emojis(text: str) -> str:
    emoji_pattern = re.compile("["
                               u"\U0001F600-\U0001F64F"  # emoticons
                               u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                               u"\U0001F680-\U0001F6FF"  # transport & map symbols
                               u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                               u"\U00002702-\U000027B0"
                               u"\U000024C2-\U0001F251"
                               "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', text)
```

We also want to remove any twitter handles, since they also doesn't contribute much to the overall meaning of the sentence.

In [ ]:
```
def remove_twitter_handles(text: str) -> str:
    return re.sub(r'@\w+', '', text)
```

Now we will apply this function and unescape any html entities.

In [ ]:
```
hasan_tweets['rawContent'] = hasan_tweets['rawContent'].apply(
    html.unescape).apply(remove_emojis).apply(remove_twitter_handles)

hasan_tweets['renderedContent'] = hasan_tweets['renderedContent'].apply(
    html.unescape).apply(remove_emojis).apply(remove_twitter_handles)
```

In [ ]:
```
print(hasan_tweets.head())
```

```
                    id                                                url
0  1662165779498795010  https://twitter.com/hasanthehun/status/1662165...  \
1  1661955536580132864  https://twitter.com/hasanthehun/status/1661955...
2  1661874461170348035  https://twitter.com/hasanthehun/status/1661874...
3  1661800157409873920  https://twitter.com/hasanthehun/status/1661800...
4  1661784494301646848  https://twitter.com/hasanthehun/status/1661784...

                 date                                         rawContent
0 2023-05-26 18:36:18  DEBT CEILING QUESTIONS ANSWERED W/  + HOGWATCH...  \
1 2023-05-26 04:40:52   if you tell them you watch me they will elimi...
2 2023-05-25 23:18:42                              this is a gay man btw
3 2023-05-25 18:23:26  RON DESASTEROUS CAMPAIGN LAUNCH, OATHKEEPER LE...
4 2023-05-25 17:21:12                         we are never getting healthcare.

                                     renderedContent       conversationId
0  DEBT CEILING QUESTIONS ANSWERED W/  + HOGWATCH...  1662165779498795010  \
1   if you tell them you watch me they will elimi...  1661918058888433664
2                              this is a gay man btw  1661874461170348035
3  RON DESASTEROUS CAMPAIGN LAUNCH, OATHKEEPER LE...  1661800157409873920
4                   we are never getting healthcare.  1661784494301646848

  lang  inReplyToTweetId
0   en               NaN
1   en      1.661918e+18
2   en               NaN
3   en               NaN
4   en               NaN
```

In [ ]:
```
replies_to_hasan['rawContent'] = replies_to_hasan['rawContent'].apply(
    html.unescape).apply(remove_emojis).apply(remove_twitter_handles)
```

```
replies_to_hasan['renderedContent'] = replies_to_hasan['renderedContent'].apply(
    html.unescape).apply(remove_emojis).apply(remove_twitter_handles)
```

In [ ]:
```
print(replies_to_hasan.head())
```

```
                    id                                                 url
1  1661955536580132864  https://twitter.com/hasanthehun/status/1661955...  \
2  1661874461170348035  https://twitter.com/hasanthehun/status/1661874...
4  1661784494301646848  https://twitter.com/hasanthehun/status/1661784...
5  1661782036074598400  https://twitter.com/hasanthehun/status/1661782...
6  1661761204648546306  https://twitter.com/hasanthehun/status/1661761...

                 date                                         rawContent
1 2023-05-26 04:40:52   if you tell them you watch me they will elimi...  \
2 2023-05-25 23:18:42                          this is a gay man btw
4 2023-05-25 17:21:12                     we are never getting healthcare.
5 2023-05-25 17:11:26                          is she the new sinema?
6 2023-05-25 15:48:39                                       you’re old

                                     renderedContent       conversationId
1   if you tell them you watch me they will elimi...  1661918058888433664  \
2                              this is a gay man btw  1661874461170348035
4                   we are never getting healthcare.  1661784494301646848
5                             is she the new sinema?  1661782036074598400
6                                         you’re old  1661451589071130635

  lang  inReplyToTweetId
1   en      1.661918e+18
2   en               NaN
4   en               NaN
5   en               NaN
6   en      1.661654e+18
```

Now we will create a stopwords list. Stopwords are words like "a", "and", "the", "an", so words that do not change the meaning of the sentence. We will also add Hasan's name second name, twitter handle and his twitch username. What we are doing here is called lemmatization. The more you know.

In [ ]:
```
stopwords_list = stopwords.words('english')
stopwords.encoding('')
stopwords_list.extend(
    ['hasan', "parker", HASAN_TWITTER_HANDLE, "hasanabi"])

print(stopwords_list[:5])
```

```
['i', 'me', 'my', 'myself', 'we']
```

This is a function that will tokenize the tweets. Tokenizing refers to the process of breaking down a text or a sentence into smaller units called tokens. Tokens are typically words, but they can also be phrases, sentences, or even individual characters, depending on the level of granularity desired. This will make our tweets easier to analyze by our NLP (natural language processing) algorithms.

In [ ]:
```
def tokenize(text: str) -> list[str]:
    words = nltk.word_tokenize(text)
    return [word.lower() for word in words if word.isalpha()]
```

This function will apply our stopwords to the tweets. This is called "lemmatization". It is the process of reducing words to their base or canonical form. The lemma represents the dictionary or canonical form of a word, from which all inflected forms (such as different tenses, plurals, or derivations) are derived. It is used to reduce different forms of a word to a common base. This is because they will be treated as the same item by our algorithms. If they were different they could make some differences making our results inaccurate.

```
In [ ]: lemmatizer = WordNetLemmatizer()


        def lemmatize(text: str) -> list[str]:
            words = [word for word in text if word not in stopwords_list]
            return [lemmatizer.lemmatize(word) for word in words]
```

Now we will apply this to our data.

```
In [ ]: hasan_tweets['rawContent'] = hasan_tweets['rawContent'].apply(
            tokenize).apply(lemmatize)

        hasan_tweets['renderedContent'] = hasan_tweets['renderedContent'].apply(
            tokenize).apply(lemmatize)
```

```
In [ ]: replies_to_hasan['rawContent'] = replies_to_hasan['rawContent'].apply(
            tokenize).apply(lemmatize)

        replies_to_hasan['renderedContent'] = replies_to_hasan['renderedContent'].apply(
            tokenize).apply(lemmatize)
```

Analyzing

Now we will do the most interesting part. We will use some of the NLP algorithms to calculate a sentiment and add this to our dataset.

```
In [ ]: sid = SentimentIntensityAnalyzer()

        replies_to_hasan['sentiment'] = replies_to_hasan['rawContent'].apply(
            lambda text: sid.polarity_scores(' '.join(text))['compound'])
```

Extract the minimum and maximum sentiments.

```
In [ ]: sentiment_max = replies_to_hasan['sentiment'].max()
        sentiment_min = replies_to_hasan['sentiment'].min()

        print(sentiment_max, sentiment_min)
```

```
0.9724 -0.9764
```

Let's print 5 most positive comments

```
In [ ]: most_positive_replies = replies_to_hasan.sort_values(
            by='sentiment', ascending=False)[:5]

        for i in range(most_positive_replies.shape[0]):
            print(f"{i+1}. {most_positive_replies.iloc[i]['renderedContent']}")
```

```
1. ['ik', 'nice', 'twitter', 'really', 'love', 'community', 'grateful', 'every', 'single', 'one', 'also', 'loved', 'hanging', 'w', 'friend', 'week', 'love', 'international', 'content', 'creator']
2. ['blacklisted', 'also', 'advocate', 'love', 'free', 'speech', 'pretty', 'sure', 'sam', 'would', 'love', 'talk', 'vaxxed', 'would', 'well', 'sure', 'travel', 'though']
3. ['devalue', 'anyones', 'accomplishment', 'however', 'one', 'escape', 'probability', 'luck', 'play', 'tremendous', 'role', 'success', 'must', 'build', 'system', 'true', 'equality', 'opportunity', 'ppls', 'material', 'need', 'taken', 'care', 'everyone', 'deserves', 'life', 'dignity']
4. ['meant', 'win', 'best', 'chatting', 'streamer', 'got', 'robbed', 'award', 'pretty', 'funny', 'many', 'thought', 'actually', 'got', 'robbed', 'excited', 'anything', 'expensive', 'porsche']
5. ['lmao', 'imagine', 'claiming', 'love', 'speech', 'losing', 'mind', 'bernie', 'attend', 'israeli', 'special', 'interest', 'conference', 'successfully', 'rolled', 'back', 'amendment', 'well', 'openly', 'stating', 'desire', 'foreign', 'interference', 'american', 'politics']
```

Some of them are gibberish two of them relate to Hasan getting a dog recently and the rest relate to political topics. Hasan's crowd at it's finest.

Now for the less fun ones. Let's see what are the 5 most negative sentiments

In [ ]:
```
most_positive_replies = replies_to_hasan.sort_values(
    by='sentiment', ascending=True)[:5]

for i in range(most_positive_replies.shape[0]):
    print(f"{i+1}. {most_positive_replies.iloc[i]['renderedContent']}")
```

```
1. ['bloomberg', 'worse', 'trump', 'muslim', 'kid', 'police', 'force', 'terrorize', 'black', 'brown', 'ppl', 'w', 'stop', 'n', 'frisk', 'post', 'redlining', 'housing', 'market', 'crash', 'accusation', 'harassment', 'sexual', 'assault', 'fingerprint', 'iraq', 'war']
2. ['also', 'element', 'injustice', 'overlooked', 'lot', 'big', 'rw', 'medium', 'racial', 'agitprop', 'point', 'horrific', 'murder', 'killer', 'apprehended', 'justice', 'served', 'reason', 'ppl', 'still', 'protesting', 'breonna', 'taylor', 'death', 'cu', 'killer', 'walk', 'free']
3. ['pat', 'tillman', 'became', 'conflicted', 'w', 'service', 'outspokenly', 'war', 'end', 'life', 'murdered', 'u', 'soldier', 'u', 'govt', 'covered', 'cause', 'death', 'lied', 'family', 'disgusting', 'cretin', 'use', 'image', 'war', 'monger', 'disrespect', 'value']
4. ['absolutely', 'ruined', 'mood', 'morning', 'trying', 'kill', 'draconic', 'tree', 'sentinel', 'great', 'bridge', 'right', 'maliketh', 'black', 'blade', 'bos', 'fight', 'utterly', 'feeling', 'humiliated', 'foe', 'stupid', 'fire', 'breathing', 'horse', 'stupid', 'lighting', 'damage', 'cleaver']
5. ['warren', 'good', 'debate', 'stage', 'bloomberg', 'took', 'insane', 'amount', 'damage', 'sander', 'campaign', 'damage', 'could', 'hurt', 'sander', 'sexism', 'smear', 'following', 'loser', 'consultant', 'worst', 'instinct', 'destroyed']
```

Wow, three of them relate to guns in US. One is about the war in Ukraine and the last one relates to a problem of porn addiction.

With our processed data we can now calculate the mean sentiment for each day and see how did sentiment in replies changed over the days.

In [ ]:
```
# Grouping by date and getting mean sentiment for each day
aggregated_df = replies_to_hasan.groupby(replies_to_hasan['date'].dt.date)[
    'sentiment'].mean().reset_index()

aggregated_df.head()
```

Out[ ]:

	date	sentiment
0	2020-01-01	0.572900
1	2020-01-03	-0.351257
2	2020-01-04	-0.680800
3	2020-01-05	0.206867
4	2020-01-06	-0.143300

In [ ]:

```
# Get 5 most positive days
aggregated_df.sort_values(by='sentiment', ascending=False).head(5)
```

Out[ ]:

	date	sentiment
179	2020-06-30	0.8720
1059	2022-12-02	0.8481
1097	2023-01-09	0.7783
1127	2023-02-11	0.7650
469	2021-04-19	0.7326

In [ ]:

```
# Get 5 most negative days
aggregated_df.sort_values(by='sentiment', ascending=True).head(5)
```

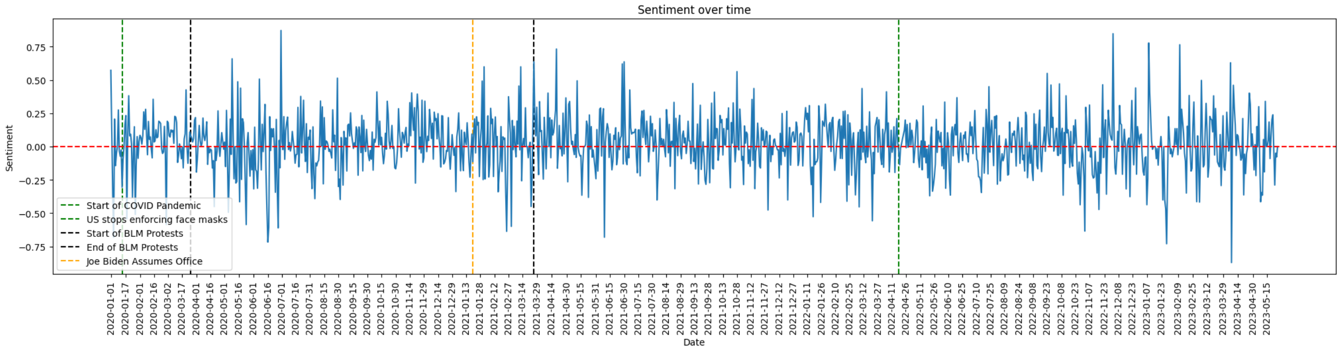
Out[ ]:

	date	sentiment
1178	2023-04-07	-0.871600
1115	2023-01-28	-0.729925
165	2020-06-16	-0.717800
519	2021-06-09	-0.680800
2	2020-01-04	-0.680800

We will not plot a graph tha will show us how did the sentiment change over the days.

In [ ]:

```
plt.figure(figsize=(25, 5))
plt.axvline(datetime(2022, 4, 18), linestyle='--', color='green')
plt.axvline(datetime(2020, 1, 13), linestyle='--', color='green')

plt.axvline(datetime(2020, 3, 26), linestyle='--', color='black')
plt.axvline(datetime(2021, 3, 26), linestyle='--', color='black')

plt.axvline(datetime(2021, 1, 20), linestyle='--', color='orange')

plt.plot(aggregated_df['date'], aggregated_df['sentiment'])

plt.axhline(0, linestyle='--', color='red')

plt.legend(['Start of COVID Pandemic', 'US stops enforcing face masks', 'Start of BLM
           'End of BLM Protests', 'Joe Biden Assumes Office'], loc='lower left')


plt.xlabel('Date')
plt.ylabel('Sentiment')
plt.title('Sentiment over time')
plt.xticks(aggregated_df['date'][::15], rotation=90)
plt.show()
```



At the start of the pandemic and during the BLM protests sentiment was a little over to the positive side. We also see spikes when Joe Biden assumed office and at the end of BLM protests. After the protests the sentiment was really positive. Than it fallen to the tendency of negativity.

But the graph doesn't belong to the most readable so let's get some more concrete values. We will get a sentiment score of all words and print them out.

In [ ]:
```
# Negativity, neutrality and positivity of all tweets
all_words = []
for i in range(replies_to_hasan.shape[0]):
    all_words.extend(replies_to_hasan['rawContent'].iloc[i])

scores = sid.polarity_scores(' '.join(all_words))
print(scores)
```

```
{'neg': 0.18, 'neu': 0.614, 'pos': 0.206, 'compound': 1.0}
```

Now we can say that most of the time sentiment is negative than positive and negative like 18% of the time. That is a good score.

Let's see what emotions are associated with all of those words.

In [ ]:
```
te.get_emotion(' '.join(all_words))
```

Out[ ]:
```
{'Happy': 0.14, 'Angry': 0.06, 'Surprise': 0.21, 'Sad': 0.24, 'Fear': 0.35}
```

Now we will copy the replies and add a column with values that will represent what emotions value does the tweet have.

In [ ]:
```
emotion_df = replies_to_hasan.copy()
emotion_df['emotion'] = emotion_df['rawContent'].apply(
    lambda text: te.get_emotion(' '.join(text)))
```

In [ ]:
```
# Save to json
emotion_df.to_json('emotions.json')
```

In [ ]:
```
# Read it from json
emotion_df = pd.read_json("emotions.json")
```

We need to split all of this columns into separate one to make it easier for us to aggregate next.

In [ ]:
```
emotion_df['happy'] = emotion_df['emotion'].apply(lambda x: x['Happy'])
emotion_df['angry'] = emotion_df['emotion'].apply(lambda x: x['Angry'])
emotion_df['surprise'] = emotion_df['emotion'].apply(lambda x: x['Surprise'])
emotion_df['sad'] = emotion_df['emotion'].apply(lambda x: x['Sad'])
emotion_df['fear'] = emotion_df['emotion'].apply(lambda x: x['Fear'])
```

We are grouping by date and then aggregating the means of each emotion.

In [ ]:
```
grouped_emotion_df = emotion_df.groupby(emotion_df['date'].dt.date)
grouped_emotion_df = grouped_emotion_df.agg(
    {'happy': 'mean', 'angry': 'mean', 'surprise': 'mean', 'sad': 'mean', 'fear': 'me

grouped_emotion_df = grouped_emotion_df.sort_values(by=['date'])
grouped_emotion_df = grouped_emotion_df.reset_index()
```

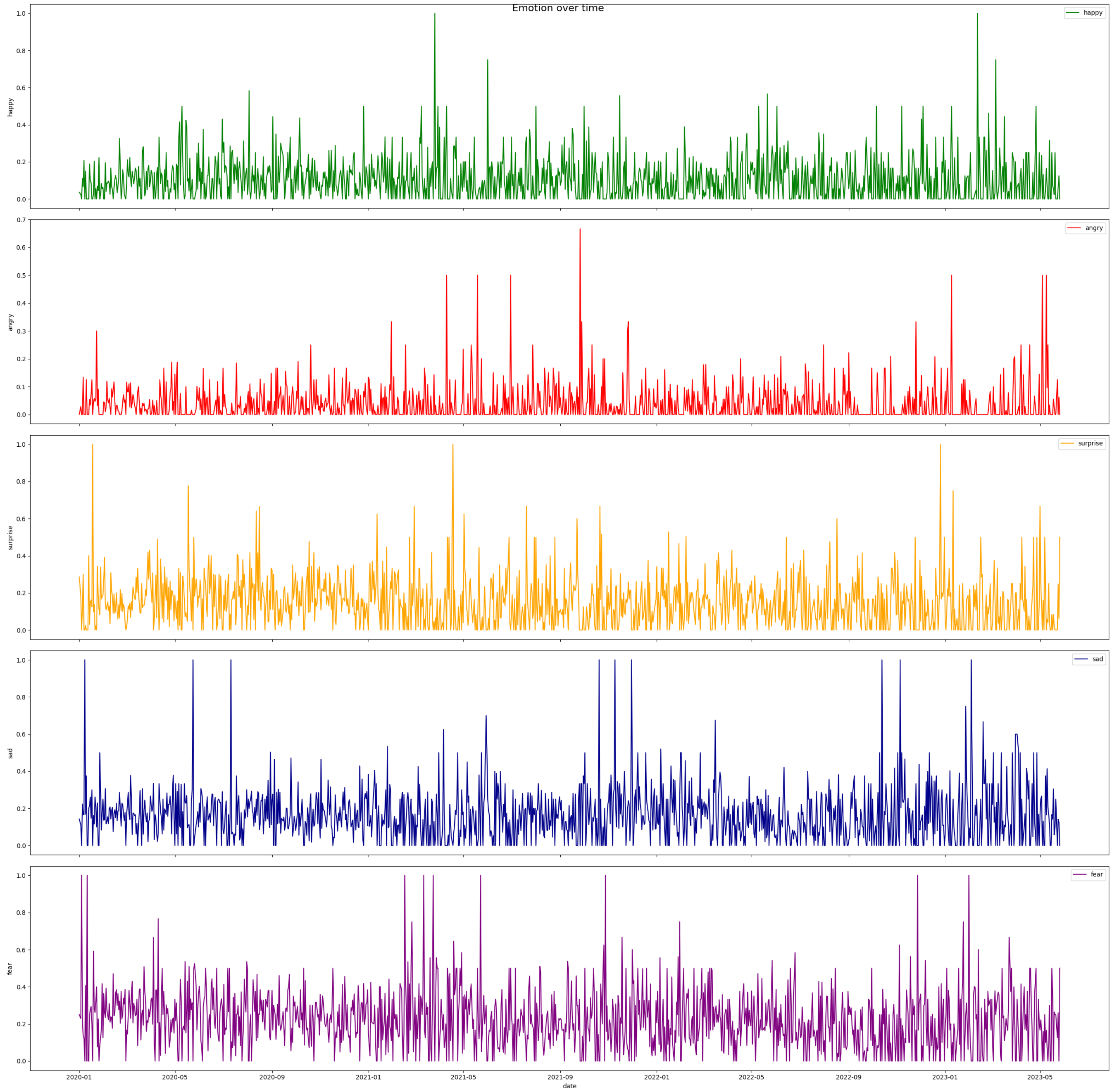
Let's see how it looks.

In [ ]:
```
print(grouped_emotion_df.head(5))
```

```
         date     happy     angry  surprise       sad      fear
0  2020-01-01  0.035714  0.000000  0.285714  0.142857  0.250000
1  2020-01-03  0.024286  0.028571  0.181429  0.110000  0.228571
2  2020-01-04  0.000000  0.000000  0.000000  0.000000  1.000000
3  2020-01-05  0.110000  0.000000  0.110000  0.223333  0.220000
4  2020-01-06  0.066000  0.134000  0.300000  0.166000  0.134000
```

Not bad if I can say so, now we will proceed to plotting those emotions.

In [ ]:
```
# plt.figure(figsize=(25, 5))

# colors = ['green', 'red', 'orange', 'darkblue', 'purple']

# for i, emotion in enumerate(grouped_emotion_df.columns[1:]):
#     plt.plot(grouped_emotion_df['date'], grouped_emotion_df[emotion], color=colors[

# plt.xlabel('date')
# plt.ylabel('emotion')
# plt.title('emotion over time')
# plt.xticks(grouped_emotion_df['date'][::15], rotation=45)
# plt.legend()
# plt.show()

num_emotions = len(grouped_emotion_df.columns[1:])
fig, axes = plt.subplots(num_emotions, 1, figsize=(
    25, 5*num_emotions), sharex=True)

colors = ['green', 'red', 'orange', 'darkblue', 'purple']

for i, emotion in enumerate(grouped_emotion_df.columns[1:]):
    axes[i].plot(grouped_emotion_df['date'],
                 grouped_emotion_df[emotion], color=colors[i], label=emotion)
    axes[i].set_ylabel(emotion)
    axes[i].legend()

plt.xlabel('date')
plt.suptitle('Emotion over time', fontsize=16)

plt.tight_layout()

plt.show()
```

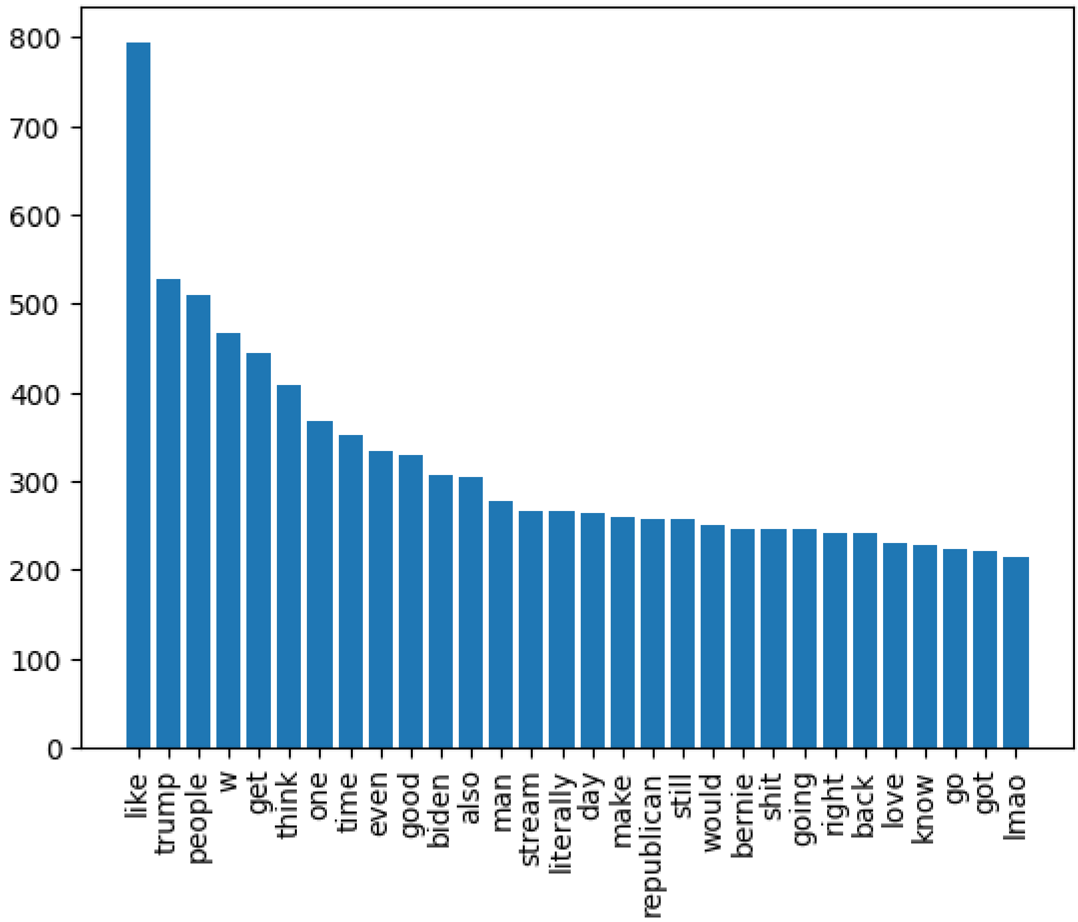


We can see that happy and angry are the lowest here. In the middle there is surprise and sad, the most common is fear. I could see why. Because of those recent times, because a lot of people on twitter are young, they are scared. Scared of what was and what will happen. Those are some difficult times that's all.

In [ ]:
```
wordcloud = WordCloud().generate(' '.join(all_words))

plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

Funny that most common word here is "trump". Others are pretty positive like "W" or neutral like "people" and "even"

In [ ]:
```
dictionary = {}
for w in all_words:
    if w in dictionary:
        dictionary[w] += 1
    else:
        dictionary[w] = 1

list_of_tuples = [(k, v) for k, v in dictionary.items()]
list_of_tuples.sort(key=lambda tup: tup[1], reverse=True)

top_10 = list_of_tuples[:30]

plt.bar([t[0] for t in top_10], [t[1] for t in top_10])
plt.xticks(rotation=90)
plt.show()
```



Here is a plot that shows how common are words.

Summary

Mr. Piker is an interesting guy. He is attracting a lot of people with different views and discussing with them. He raises a lot of emotions. Most of them neutral. Sometimes positive sometimes negative. But more positive. He can be controversial because of his "unpopular" in US views. Interestingly enough fear was the outlier here. Not the anger, which you could associate with politics.

Fear is an interesting one since like I said before. It shows what, mostly twitter demographic, feels like. Maybe we could look at it and ask ourselves "Why is that?", "What should we do?" and do it.

Bibliography:

- https://www.nltk.org/
- https://pypi.org/project/text2emotion/
- https://twitter.com/hasanthehun
- https://www.geeksforgeeks.org/tokenize-text-using-nltk-python/
- https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
- https://www.geeksforgeeks.org/python-lemmatization-with-nltk/
- https://pypi.org/project/wordcloud/

- https://pypi.org/project/snscrape/
- https://betterprogramming.pub/how-to-scrape-tweets-with-snscrape-90124ed006af
- https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk
- https://realpython.com/python-nltk-sentiment-analysis/