Extract prevalent topics from Twitter messages

Code link: https://github.com/hennypurwadi/twitter_analysis

Collect Twitter's data

With the snscrape library, we can collect tweets without having a Twitter API developer account, fetching twitter data for usernames, tweets, hashtags, at locations within a 30 km radius around Downing Street London, from 29 to 30 July 2022.

A total of 40,000 tweets in the data set, with 10,289 of tweets containing hashtags.

Collect Tweets Without Twitter Developer Account /API: with snscrape

In [7]:
```
#Create and append to csv file and then write tweets into csv file
tweet_data = open('London_tweets_sns.csv', 'a', newline='', encoding='utf8')
csv.writer(tweet_data).writerow(['username','tweet','hashtags'])

#Radius around 30 km around particular geocode
max_tweets = 40000
for n,tweet in enumerate(sntwitter.TwitterSearchScraper('geocode:51.50352,-0.12764,30km \
        + since:2022-07-29 until:2022-07-30 lang:en -filter:links -filter:replies').get_items()):

    if n > max_tweets:
        break
    csv.writer(tweet_data).writerow([tweet.user.username, tweet.content, tweet.hashtags])
tweet_data.close()
```

In [8]:
```
df = pd.read_csv('London_tweets_sns.csv')
df = df.dropna(subset = ['hashtags']) #Only use tweets which contained hashtags
df.head()
```

Out[8]:

	username	tweet	hashtags
2	TyroneChambers_	Independent Women\n\nWho run the world\n\nYour u...	['TALKSWITHASHAFTERDARK']
6	Z4ch3ryW	Wordle 406 4/6\n\n⬛⬛⬛⬛⬛\n⬛⬛⬛⬛⬛\n🟨🟩⬛🟨⬛\n🟩🟩🟩🟩🟩\n...	['wordle', 'wordle406']
17	FredoHnDz_20	U can't say no to an empty gym. Vamos 🔥🤟🏽 #wor...	['workout', 'pickurlazyassup']
21	MillieSansoye	There's nothing straight about Xanadu #Uncoupled	['Uncoupled']
23	Teach_WA	Would someone please attest that driving is no...	['gtfa', 'MarioKart']

In [9]:
```
len(df)
```

Out[9]: 10289

Entity analysis to find 5 most active users

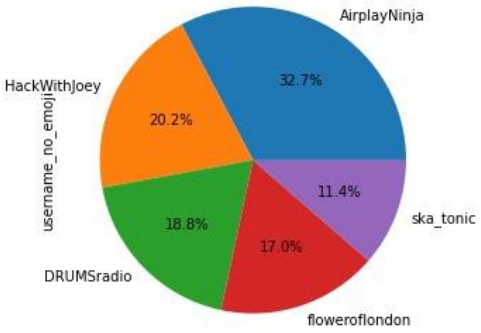
The frequency for the most active user is 164 tweets in two days, from July 29 to July 30, 2022.

1. Entity analysis to find 5 most active users.

In [11]:
```
#df['user'].unique()
userlist = df['username_no_emoji'].value_counts()
most_active_users = userlist.head(5)
most_active_users
```

Out[11]:
```
AirplayNinja      164
HackWithJoey      101
DRUMSradio         94
floweroflondon     85
ska_tonic          57
Name: username_no_emoji, dtype: int64
```

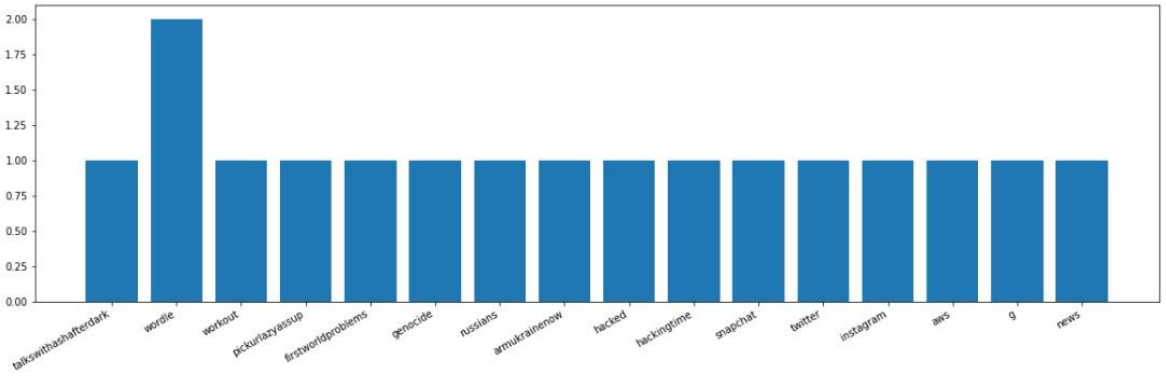
In [12]:
```
plot_users=most_active_users.plot.pie(autopct='%1.1f%%', figsize=(5, 5))
plt.rc('axes', unicode_minus=False)
plt.savefig("MostActiveUser_chart.jpeg",transparent=False, bbox_inches='tight',pad_inches=0.1)
```



Entity analysis to find 5 most frequently used hashtags

After lite cleaning with regex to make lowercase special characters etc, we can count values to find five most popular hastags.

In [15]:
```
import matplotlib.pyplot as plt
fig, ax1 = plt.subplots(figsize=(20, 6))
ax1.bar(hashtag_count.keys(), hashtag_count.values())
fig.autofmt_xdate()
plt.savefig('hashtag_count_graph.jpeg')
plt.show()
```



In [16]:
```
MostPopularHashtags = sorted(hashtag_count, key=hashtag_count.get, reverse=True)[:5]
#' '.join(MostPopularHashtags)
MostPopularHashtags
```

Out[16]:
```
['wordle',
 'talkswithashafterdark',
 'workout',
 'pickurlazyassup',
 'firstworldproblems']
```

Entity analysis to extract five most prevalent topics in the tweets.

Pre-processing data cleaning:

1. Using regex to make lowercase, removing URLs, special characters, web scraping, lemmatization, stemming.
2. Using nltk for stop words, tokenization.
3. Vectorize cleaned tweets with CountVectorizer with ngrame_range.
4. N-gram range sets if features to be used to characterize texts will be:

   - Unigrams or words (n-gram size = 1).

   - Bigrams or terms compounded by two words (n-gram size = 2).

   - Trigrams or terms compounded by up to three words (n-gram size = 3).

   ngram_rangetuple (min_n, max_n), default = (1, 1).

   (1, 2) means unigrams and bigrams.

   (2, 2) means only bigrams.

   (1, 3) means unigrams, bigrams, and trigrams.

Vectorize cleaned tweets with CountVectorizer with ngram_range(1,3).

Vectorize cleaned tweets with Tf-IDF ngram_range(1,3).

Vectorize cleaned tweets with CountVectorizer with ngrame_range

ngram_range of (1, 1) means only unigrams. (1, 2) means unigrams and bigrams. (1, 3) means unigrams, bigrams, and trigrams. (2, 2) means only bigrams.

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

In [23]:
```
countv = CountVectorizer(min_df=1,max_df=0.95,ngram_range=(1,3),stop_words='english')
countv_tweet = countv.fit_transform(df['cleaned_tweet'])
countv_tweet
```

Out[23]:
```
<10289x127282 sparse matrix of type '<class 'numpy.int64'>'
        with 208432 stored elements in Compressed Sparse Row format>
```

In [24]:
```
countv_tweet.shape
```

Out[24]:
```
(10289, 127282)
```

Vectorize cleaned tweets with Tf-IDF with ngram_range

In [25]:
```
tfidf = TfidfVectorizer(min_df=1 ,max_df=0.95, ngram_range=(1,3),stop_words='english')
tfidf_tweet = tfidf.fit_transform(df['cleaned_tweet'])
tfidf_tweet
```

Out[25]:
```
<10289x127282 sparse matrix of type '<class 'numpy.float64'>'
        with 208432 stored elements in Compressed Sparse Row format>
```

Topics extraction using LDA/ Latent Dirichlet Allocation from CountVectorizer:

```
Most prevalent topic #0:
['paig', 'year', 'luca', 'gemma', 'dami indiyah', 'like', 'watch', 'love', 'indiyah', 'dami']
Most prevalent topic #1:
['way', 'good', 'let', 'peopl', 'win', 'year', 'love', 'time', 'like', 'know']
Most prevalent topic #2:
['day', 'good', 'think', 'say', 'love', 'watch', 'time', 'luca', 'gemma', 'like']
Most prevalent topic #3:
['love', 'andrew', 'ekin david', 'look', 'luca', 'date', 'gemma', 'vote', 'ekin', 'david']
Most prevalent topic #4:
['paig', 'date', 'radio', 'final', 'bbc radio', 'bbc', 'need', 'luca', 'like', 'gemma']
```

Topics extraction using LDA/ Latent Dirichlet Allocation from Tf-IDF:

```
Most prevalent topic #0:
['like', 'vote', 'luca', 'paig', 'love', 'alert', 'dami indiyah', 'gemma', 'indiyah', 'dami']
Most prevalent topic #1:
['date', 'ekin', 'luca', 'david', 'like', 'vote', 'win', 'love', 'gemma', 'know']
Most prevalent topic #2:
['come', 'gemma luca', 'vote', 'good', 'love', 'like', 'luca', 'gemma', 'time', 'watch']
Most prevalent topic #3:
['electron', 'worldwid', 'african', 'luca', 'andrew', 'ekin', 'gemma', 'david', 'love', 'vote']
Most prevalent topic #4:
['oh', 'david', 'revolut', 'paig', 'final', 'love', 'like', 'luca', 'gemma', 'date']
```

From these two results, we can see that LDA is not good for topic modeling for incoherent texts like twitter, because word like 'gemma' and 'love' are repeated across all topics.

In contrast, using NMF without TF-IDF, we get better result than LDA.

Topics extraction using NMF/Non-Negative Matrix from CountVectorizer:

```
Most prevalent topic #0:
['gemma luca date', 'luca date', 'tasha', 'adam', 'paig', 'luca gemma', 'date', 'gemma luca', 'luca', 'gemma']
Most prevalent topic #1:
['say', 'final', 'know', 'peopl', 'look', 'year', 'time', 'love', 'watch', 'like']
Most prevalent topic #2:
['paig', 'indiyah', 'vote ekin', 'dami', 'ekin su', 'su', 'ekin david', 'vote', 'ekin', 'david']
Most prevalent topic #3:
['mixplay african electron', 'african electron music', 'electron music', 'african electron', 'music worldwid', 'electron music
worldwid', 'electron', 'worldwid', 'african', 'music']
Most prevalent topic #4:
['alert servic alert', 'rail statu jul', 'rail statu', 'statu jul', 'rail', 'jul', 'statu', 'train', 'servic', 'alert']
```

Topic #0, #1, and #2 are about names in Love Island, British reality show final episode. Topic #3 is all about music. Topic #4 is about train and rail.

Topics extraction using NMF/Non-Negative Matrix from Tf-IDF:

```
Most prevalent topic #0:
['statu jul train', 'statu jul', 'train alert servic', 'alert servic alert', 'alert servic', 'jul train alert', 'servic alert',
'train alert', 'jul train', 'alert']
Most prevalent topic #1:
['paig adam', 'luca date', 'like', 'adam', 'paig', 'luca gemma', 'date', 'gemma luca', 'luca', 'gemma']
Most prevalent topic #2:
['su david', 'david ekin', 'ekin su', 'su', 'vote ekin david', 'vote ekin', 'ekin david', 'vote', 'ekin', 'david']
Most prevalent topic #3:
['mixplay african', 'worldwid', 'electron', 'music worldwid', 'african electron music', 'electron music worldwid', 'african ele
ctron', 'electron music', 'african', 'music']
Most prevalent topic #4:
['island', 'andrew', 'tasha', 'final', 'time', 'watch', 'dami indiyah', 'indiyah', 'dami', 'love']
```

Topic #1, #2, and #4 are about names in Love Island, British reality show final episode. Topic #3 is all about music. Topic #0 is about train and rail.

While the first assumption was that the topic of the new British Prime Minister might become a hot topic within a 30km radius around the government offices on 10 Downing Street London, the result was that the reality show Love Island became a hot topic there on 30 July 2022.

Problems while working on the project:

Error opening dataset while code is still running to collect tweets.

Solution:

Problem solved after restarting Kernel in Jupyter Notebook, and closing csv dataset.

Conclusion:

LDA is good in identifying **coherent** texts, while **NMF** usually good for **incoherent** texts like tweets. For further analysis, need to compare topic modelling result from LDA and NMF for coherent texts, with and without Tf-IDF.

Although doesn't need Tf-IDF to infer topics, using Tf-IDF can improve result. Tf-IDF outperforms CountVectorizer, because it not only focuses on the frequency of words present in the corpus but also provides the importance of the words.

Justification:

TF-IDF is better than Count Vectorizers because it not only focuses on the frequency of words present in the corpus but also provides the importance of the words. We can then remove the words that are less important for analysis, hence making the model building less complex by reducing the input dimensions. (Sheel Saket, 2020).

Applied topic modeling over the tweets to obtain meaningful data from Twitter, comparing and analyzing topics detected by two popular topic modeling algorithms; Non-negative Matrix Factorization (NMF) and Latent Dirichlet Allocation (LDA). The observed results show that **LDA outperforms NMF** in terms of their topic **coherence**. (Nassera Habat, 2021).

Literature:

Marco Bonzanini.(2016). Mastering Social Media Mining with Python.p.177.

Mohamed Ben Ahmed. İsmail Rakıp Karaş Domingos Santos. Olga Sergeyeva Anouar Abdelhakim. Boudhir.(2021).Innovations in Smart Cities Applications Volume 4.

Web Article:

Sheel Saket.(2020). Count Vectorizer vs TFIDF Vectorizer | Natural Language Processing.

Topic Modeling and Sentiment Analysis with LDA and NMF on Moroccan Tweets. Retrieved from

https://link.springer.com/chapter/10.1007/978-3-030-66840-2_12

Anand Borad(2020). NLP text Pre-Processing: Text Vectorization. Retrieved from:

https://www.einfochips.com/blog/nlp-text-vectorization/#:~:text=There%20are%20three%20most%20used,separate%20article%20for%20word%20embedding

scikit-learn. 1.1.1. Retrieved from:
https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

metro.co.uk. (2022). When does Love Island 2022 end? Final episode date revealed. https://metro.co.uk/2022/07/27/when-does-love-island-2022-end-final-episode-date-revealed-2-17079275/