

Learn to rehydrate Twitter data using Python: case #helloworld

Abstract

In research, it is common practice to share Twitter datasets using only tweet identifiers; this is done by way of dehydration and the subsequent rehydration of the dataset for further qualitative analysis. This tutorial demonstrates the process of rehydration that simply refers to using the Twitter API to recollect or retrieve the tweets that the tweet identifiers refer to using Python programming language, Jupyter Notebooks and a third-party tool named Twarc. Rehydration is the standard way to share Twitter data in accordance with Twitter Terms of Service (ToS) that only allows the sharing of tweet identifiers, not the full tweet data. In addition, this tutorial explores the degradation and disappearance of data that occurs when tweets are removed by users or through moderation, and which becomes evident once the data is rehydrated and not all of the dataset can be retrieved. Rehydration is especially useful for mixed-method approaches that include qualitative ethnographic analysis of computational data because rehydration allows the sharing of datasets between researchers, but also enables the (re)construction of the field for ethnographic analysis. The tweetset under investigation in this tutorial includes tweets tagged with the commemorative hashtag #helloworld, collected in the context of the Christchurch mosque attacks in 2019. Although the #helloworld tweetset is mainly commemorative in nature, some content might be perceived as more sensitive. This tutorial includes the #helloworld tweetset in the form of tweet IDs, a How-to Guide for rehydrating the data, as well as an analytical notebook for exploring the rehydrated tweetset.

Student Guide

Introduction

Twitter often serves as the go-to platform for information and communication during sudden and unexpected events, capturing the first impressions. Thus, as a “first draft of the present” (Bruns & Weller, 2016), Twitter frequently serves as the data source for analyses in different academic disciplines, including in the study of hybrid media events (Sumiala et al., 2018), the domain of this tutorial. This tutorial guides you through the process of sharing a Twitter dataset for research purposes and setting up a shared dataset for analysis. Unlike many of the other datasets in SAGE Datasets that are openly downloadable for further analysis, Twitter Terms of Service (ToS) sets major restrictions to how tweet datasets, or *tweetsets*, can be shared; instead of sharing the full set of data, Twitter ToS only allows sharing the list of tweet identifiers (tweet IDs). Specifically, the ToS allow sharing 50 000 hydrated tweets (that is, including all the details) privately between researchers and a total of 1 500 000 tweet IDs “to a single entity within a 30 day period” (Twitter Developer Platform, 2022). To be able to analyse the data, the analyst must then recollect or *rehydrate* the tweetset using the list of tweet IDs. In this tutorial, you will learn to conduct this

process and also to explore the implications of this practice. Moreover, we will demonstrate one of the key issues relating to Twitter data that becomes evident during the rehydration process, that is, the degradation and disappearance of Twitter data: tweets that have been deleted from Twitter before the rehydration process is carried out are no longer available for research as these cannot be recollected (Bruns, 2019; Freelon, 2018; Tromble, 2021).

The tweetset under investigation in this tutorial, #helloworldbrother, was collected in the context of the Christchurch mosque attacks in New Zealand where 51 Muslims were killed during Friday prayer by a single perpetrator on 15th March, 2019. The commemorative hashtag #helloworldbrother has its roots in the final words of the first victim, seen greeting the killer on the live footage taken by the perpetrator. It thus represents a unique dataset specific to this incident and comprises tweets of solidarity, remembrance, and condemnation of the attack. In addition to public commemoration, the #helloworldbrother tweetset includes shows of solidarity and expressions of commiseration, but also of shock. While the majority of the tweets serve to remember and respect the life of this particular individual, some content might, however, be perceived as sensitive by some as the tweets link to and reference the attack itself. No images or photos embedded in the tweets will come up following this tutorial and in the process of rehydration; however, these are still recoverable on Twitter and will be visible if rehydrated tweets are viewed on Twitter. At the same time, however, the potentially sensitive nature of the dataset raises important research ethical questions pertaining to the collection and use of online data in research, also bringing up questions relating to the permanence, degradation, and disappearance of data (for example, through content moderation which will affect the dataset and the amount of tweets that can in fact be rehydrated). It is thus important to note that, depending on the institution, an ethical approval to use (and, effectively, re-collect) this dataset might be required. The qualitative, ethnographic analysis of the dataset is described in detail elsewhere (Harju & Huhtamäki, 2021). Our approach to analysing #helloworldbrother combines qualitative ethnographic research with computational methods. We draw on digital methods (Rogers, 2019) and data science as a *kind of* ethnography (Lindgren, 2020). However, although the examples we present here draw on this multimethod approach, the focus of this tutorial is on the *technical process required to rehydrate a tweetset* for further qualitative analysis.

The process of rehydrating the dataset allows more than sharing of the dataset: it also allows the reconstruction of the ethnographic field by other researchers, or more precisely, reconstructing what we call an *ethnographic data field* (Harju & Huhtamäki, 2021). The ethnographic data field is, in essence, the set of tweets in the dataset that are still recoverable (that is, have not been deleted) via the Twitter API through the process of rehydration and as such, one iteration of the field.

Introducing Twitter API

Twitter provides an Application Programming Interface (API) for collecting data where the API enables programmatic access to the web platforms. Twitter API allows not only collecting tweets but also other means of automation (the development of bots, for example). Here, we focus our attention on collecting Twitter data. To be able to use the Twitter API, you first need to create a developer account at developer.twitter.com and create a new [Developer App](#) that enables you to

start using the API. Please note that it might take days or even weeks to get the developer account. Therefore, we urge you to follow the instructions in the How-to Guide for Python to submit your application as early as possible.

There are two main approaches to collect tweets: a search-based or a streaming-based approach. In the search-based approach, a collector software sends a series of requests to the API to receive results, whereas a streaming-based collector registers a set of search terms with the Twitter API and then receives tweets in real time as they are sent by the users. We used the streaming approach to collect the #hellobrother tweets.

Searching the Twitter API with the basic access that is available for all the developers is limited to recent tweets, going back roughly two weeks or some tens of thousands of tweets. Importantly, however, Twitter provides a separate access for academic researchers that enables searching the full timeline of historical tweets. Similar to rehydration, the major limitation here is that tweets that have since been removed from Twitter are no longer available for researchers.

To circumvent the aforementioned issues, we used the streaming-based approach for the Christchurch mosque attacks. We developed and set up a dedicated Twitter data collector that uses the Twitter Streaming API to collect tweets in real time, up to 180 000 per hour as limited by the Twitter API. Importantly, you should be aware that even the streaming approach yields only a sample of tweets (see Campan et al. 2018) and not “all of the data” linked to the search terms. In our case, out of the full dataset of almost 12 million tweets collected with a variety of search terms relating to the Christchurch attacks, we extracted 43 659 tweets with the hashtag #hellobrother. This dataset is included in this tutorial as a list of tweet identifiers (IDs) for these tweets.

The API provides representations of tweets in a JSON format. JSON is shorthand for Javascript Object Notation, and is a commonly used data format in Web development and data analysis. As #hellobrother might by some be perceived as asensitive topic, let’s instead use a tweet from Jacinda Ardern, the Prime Minister of New Zealand, as an illustrative example (see Listing 1 below).

```
{
  'created_at': 'Fri Mar 15 03:33:02 +0000 2019',
  'id': 1106397870628847617,
  'id_str': '1106397870628847617',
  'text': 'What has happened in Christchurch is an extraordinary act of
unprecedented violence. It has no place in New Zealand...
https://t.co/XOHF9hHe8H',
  'truncated': True,
  ...
  'user': {
    'id': 22959763,
    'id_str': '22959763',
    'name': 'Jacinda Ardern',
```

```
'screen_name': 'jacindaardern',  
'location': 'Auckland, New Zealand',  
'description': "Prime Minister of NZ. Leader @nzlabour. Won't tweet what I ate  
for breakfast-make no promises beyond that. Authorised by Timothy Grigg 160  
Willis St, Wellington",  
...
```

Listing 1. JSON representation of a tweet

You can access the tweet here: <https://twitter.com/jacindaardern/status/1106397870628847617>.
As the example shows, JSON allows the hierarchical representation of data. That is, data can include key-value-pairs (e.g., `created_at` with value `Fri Mar 15 03:33:02 +0000 2019`) and embedded structures such as the `user` field with a set of key-value pairs as its value. As you will learn in this tutorial, Python allows easy ways to access and process data in JSON.

How to share and rehydrate Twitter data?

Once the tweetset is collected, it can be analysed in different ways, for example, by carrying out a computationally augmented ethnography (Harju & Huhtamäki, 2021) that draws on interpretive data science (Lindgren 2020) or by a fully quantitative statistical analysis. However, if researchers wish to share the tweetset to others as open data, they must do so in accordance with the Twitter ToS. Sharing data for reuse is common practice in open science and in research in general; however, Twitter ToS insists on sharing tweet IDs only. Thus, we first have to *dehydrate* the dataset in order to share it. See Listing 2. for an example of a dehydrated dataset.

```
1106477228337127424  
1106484041438674944  
1106484262302289920  
1106484776914092033  
1106486835788894210  
1106487521725145090  
1106488954784763904  
1106489002738081792  
1106493724148801538  
1106494087518187521
```

Listing 2. Excerpt of the dehydrated #hellobrother dataset

To (re)construct and analyse the dataset, one has to use the Twitter API to fetch each of the tweets in the list of tweet IDs again, that is, to *rehydrate* it. Although implementing a simple rehydration process is straightforward, we recommend using existing implementations to ensure a smooth process and reduce the likelihood of errors. In the How-to tutorial, we will guide you through the rehydration process using Twarc, a popular command-line tool for collecting and rehydrating Twitter data.

Whether those conducting academic research should strictly follow the guidelines and ToS set up by the platform is an ongoing discussion (see e.g., Bruns, 2019; Freelon, 2018). Recently, a group of academic researchers were forced by Facebook to delete an existing Instagram dataset due to alleged privacy concerns (for more information about the dispute, see Kayser-Bril, 2021) while the platform is also known to have closed researchers' accounts over disagreement (Hatmaker, 2021). At the same time, discussion on the prevalent issue of accessing social media data through partly or fully black box mechanisms moves forward (Tromble, 2021). Following Bruns (2019), we, too, "demand that social media platforms provide transparent data access to critical, independent, public-interest research."

Reconstructing the ethnographic data field through rehydration

In this tutorial, our aim is not only to demonstrate the rehydration of a tweetset, but also the rehydration of an ethnographic data field to better illustrate the purpose and use(s) of sharing tweetsets among researchers. Ethnography is a qualitative research approach with roots in anthropology where one of the key concepts is the 'field', forming the object of study. When researching fluid, digital contexts (see Markham & Lindgren, 2014; Markham & Gammelby, 2018), however, the field often proves challenging as it is constantly shifting and difficult to grasp (see Markham, 2017). Rehydration is a useful tool for (re)constructing the ethnographic field, or the ethnographic data field, for further qualitative analysis.

In line with our multi-method approach, we follow Harju & Huhtamäki's (2021) conceptualisation of the *ethnographic data field* to better capture the way the ethnographic field is the result of rehydration: as we rehydrate the dataset, we in fact (re)construct the ethnographic field by recollecting those tweets still available. This reconstruction happens each time we rehydrate the tweetset by retrieving the tweets from the Twitter API and thus the field is potentially different every time we reconstruct it.

Due to data loss and data degradation, the resulting ethnographic data field typically does not correspond to the full dataset (and therefore does not fully correspond to the ethnographic field at the time of data collection and ethnographic observation in the field). Rather, it reflects the current 'live' situation on Twitter and the tweets in the dataset that are still available on Twitter. To demonstrate this major issue with the rehydration process, we will next compare the original and the rehydrated datasets to see how data is lost during the rehydration cycle. While the full reconstruction of the ethnographic data field for #helloworld is beyond the scope of this tutorial, we will focus our attention on the hashtags that received the most and least attention in the original tweetset.

For the benefit of this tutorial, we ran the rehydration process for the original tweetset of 43 659 tweets. At the time of writing, we were able to recollect only 25 603 tweets from the original tweetset; that is, almost 40% of the tweets have already been removed since they were collected. In terms of qualitative ethnographic research, this means that the reconstructed field rather represents the state of the field during the time of rehydration rather than the time of the event. However, sharing and rehydrating tweetsets is nevertheless very useful: not only does it allow sharing the dataset

as open data, but also sharing the field in its reconstructed form, as an ethnographic data field, among the research team for further qualitative research, potentially opening up new avenues of interrogation.

For details on why a particular tweet was removed, we could use the [Twitter compliance API that allows a tweet data redistributor to check whether they have to remove or refine some of the data to reflect the changes on the Twitter](#). However, following strictly the Twitter ToS, we should limit the use of this additional information to revise the dataset accordingly (i.e., not using the compliance information to augment the original tweetset for researching, for example, the reasons why a particular tweet was deleted or account suspended) (see Meeks, 2016). However, as an academic researcher, you should weigh your options carefully, including consideration of the importance of researching the topic you plan to study and the possible repercussions that going against Twitter ToS might entail.

Considerations and Cautions

- Twitter streaming and recent search APIs provide only a sample of the data, and the details of the sampling process are not transparent
- The Academic Research access enables full-archive search, yet it does not provide data that has been removed from the platform
- Rehydration only re-collects tweets that are available when running the process and thus the resulting tweetset is likely to be different each time rehydration is carried out
- Rehydration process always provides the status of the field at the time of the rehydration process, not the time of the event
- Research ethics, Terms of Service, and legislation should all be considered separately
- When strictly following Twitter ToS, we are not able to identify what is missing from a rehydrated dataset

How-to Guide for Python

For instructions on how to set up Python, Anaconda, and Jupyter, please refer to the How-to Guide for Python.

Review

This dataset tutorial gave a quick introduction to the basics of the Twitter API, demonstrated the processes of dehydration and rehydration that allows researchers to share and recollect tweetsets and the process for exploring the tweetset. Through comparing the rehydrated and the originally collected and shared tweetsets, the inevitable degradation of Twitter data was illustrated.

You should know:

- What is Twitter API?
- What are the different ways to collect Twitter data?
- What is rehydration?
- Why is rehydration needed in sharing Twitter data?

- What are the benefits of rehydration for qualitative ethnographic research and analysis?
- How to explore a rehydrated tweetset?
- What are the different representations of Twitter data?
- What are the main weaknesses and limitations of rehydration?

References

- Bruns, A. (2019). After the 'APIcalypse': social media platforms and their fight against critical scholarly research. *Information, Communication & Society*, 22(11), 1544–1566. <https://doi.org/10.1080/1369118X.2019.1637447>
- Bruns, A., & Weller, K. (2016). Twitter as a first draft of the present - And the challenges of preserving it for the future. *WebSci 2016 - Proceedings of the 2016 ACM Web Science Conference*, 183–189. <https://doi.org/10.1145/2908131.2908174>
- Freelon, D. (2018). Computational Research in the Post-API Age. *Political Communication*, 35(4), 665–668. <https://doi.org/10.1080/10584609.2018.1477506>
- Harju, A. A., & Huhtamäki, J. (2021). '#helloworld needs to trend': methodological reflections on the digital and emotional afterlife of mediated violence. *International Review of Sociology*, 31(2), 1–32. <https://doi.org/10.1080/03906701.2021.1947951>
- Hatmaker, T. (2021). Facebook cuts off NYU researcher access, prompting rebuke from lawmakers. TechCrunch. Available:
- Kayser-Bril, N. (2021). *AlgorithmWatch forced to shut down Instagram monitoring project after threats from Facebook*. AlgorithmWatch. Available: <https://algorithmwatch.org/en/instagram-research-shut-down-by-facebook/>
- Lindgren, S. (2020). *Data Theory: Interpretive Sociology and Computational Methods*. Wiley.
- Markham, A. N. (2017). Ethnography in the Digital Internet Era: from fields to flow, descriptions to interventions. In N. Denzin, & Y. Lincoln (Eds.), *The SAGE handbook of qualitative research* (5th ed., pp. 650–668). Sage.
- Markham, A. N., & Gammelby, A. K. (2018). Moving through digital flows: An epistemological and practical approach. In U. Flick (Ed.), *The SAGE handbook of qualitative data collection* (pp. 451–465). Sage. <https://doi.org/10.4135/9781526416070>
- Markham, A. N., & Lindgren, S. (2014). From object to flow: Network sensibility, symbolic interactionism, and social media. *Symbolic Interaction and New Social Media* (pp. 7–41). Emerald. <https://doi.org/10.1108/S0163-239620140000043012>

- Meeks, L. (2016). Tweeted, deleted: theoretical, methodological, and ethical considerations for examining politicians' deleted tweets. *Information, Communication & Society*, 21(1), 1–13. <https://doi.org/10.1080/1369118X.2016.1257041>
- Rogers, R. (2019). *Doing Digital Methods*. SAGE Publications Ltd.
- Sumiala, J., Valaskivi, K., Tikka, M., & Huhtamäki, J. (2018). *Hybrid Media Events: The Charlie Hebdo Attacks and the Global Circulation of Terrorist Violence*. Emerald Publishing Limited. <https://doi.org/10.1108/9781787148512>
- Tromble, R. (2021). Where Have All the Data Gone? A Critical Reflection on Academic Digital Research in the Post-API Age: *Social Media + Society*, 7(1). <https://doi.org/10.1177/2056305121988929>
- Twitter Developer Platform (2022). More about restricted uses of the Twitter APIs. Twitter developer terms. Available: <https://developer.twitter.com/en/developer-terms/more-on-restricted-use-cases>

How-to Guide for Python

Introduction

The Twitter Terms of Service (ToS) allows only one way to share tweet datasets, or *tweetsets*. That is, instead of sharing the actual data, one can only share the list of tweet identifiers (IDs). Therefore, when sharing a dataset, one has to *dehydrate* it. This simply refers to the process of removing all data beyond the tweet IDs. When reusing the shared dataset, the first task is the *rehydration* of the said dataset. Rehydration refers to the process of recollecting the tweets using the Twitter Application Programming Interface (API).

In this tutorial, you will first install the necessary tools needed in the dehydration and rehydration processes, then rehydrate an example dataset, and start the exploration of the rehydrated dataset. This tutorial does not cover all the details of installing the key tools, including Python, Jupyter Notebooks, and Twarq; instead, we will introduce the basics and point you to relevant tutorials that allow you to set up your working environment.

1 Before rehydration: Dehydrating and sharing a tweetset

Sharing a tweetset requires *dehydrating* it first. After dehydration, the #helloworld tweetset consists of 43 659 lines of tweet identifier values.

Let's review the first lines of our code for this tutorial. For our analysis project, we use the Pandas DataFrames to manage the tweetset. [Pandas](#) is one of the key Python packages in data science. Essentially, it implements the spreadsheet functionality for Python, and thus allows managing and operating the data in a tabular format. To dehydrate a tweetset, we simply remove all fields that fall beyond the tweet ID and save the resulting list as a text file.

Please note that unlike all the other code examples in this tutorial, you are not able to run the following lines because you do not have access to the full hydrated tweetset:

```
import pandas as pd
df = pd.read_pickle('data/hellobrother-full-dataset_idfix_str.pkl.gz')
```

So far, we have imported the Pandas library and read the tweetset in a DataFrame format. The term *pickling* refers to a way of saving Python data structures and other objects between development sessions. For development use, we prefer pickling over representing data in Comma separated Values (CSV) format because it maintains the specific data structure.

The `info()` function gives details about the DataFrame contents:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 43659 entries, 1106477228337127424 to 1173570452389863424
Data columns (total 19 columns):
 #   Column                Non-Null Count  Dtype
---  -
 0   id_str                43659 non-null  object
```

```

1   created_at          43659 non-null  datetime64[ns]
2   user_screen_name    43659 non-null  object
3   user                43659 non-null  object
4   text                43659 non-null  object
5   user_mentions       43659 non-null  object
6   user_mentions_screen_name 43659 non-null  object
7   hashtags            43659 non-null  object
8   full_text           18105 non-null  object
9   full_text_user_mentions 18105 non-null  object
10  full_text_user_mentions_screen_name 18105 non-null  object
11  full_text_hashtags   18105 non-null  object
12  retweet_count        43659 non-null  int64
13  collection_method     43659 non-null  object
14  harmonized_text       43659 non-null  object
15  harmonized_user_mentions 43659 non-null  object
16  harmonized_user_mentions_screen_name 43659 non-null  object
17  harmonized_hashtags   43659 non-null  object
18  hellobrother         43659 non-null  bool
dtypes: bool(1), datetime64[ns](1), int64(1), object(16)
memory usage: 6.4+ MB

```

We now have 43 659 tweets, each with 19 columns of metadata, most of which are composed of several additional fields.

As we can see in the following, dehydration is a fairly simple task.

```
df[['id_str']].to_csv('hellobrother_idlist.txt', index=False,
header=False)
```

Here, we first select the `id_str` column of the `DataFrame` and use the `to_csv()` function to write the `DataFrame` to a file. Moreover, we remove the index and header from the resulting file, resulting in a simple list of tweet IDs (see below).

An excerpt from `hellobrother_idlist.txt`:

```

1106477228337127424
1106484041438674944
1106484262302289920
1106484776914092033
1106486835788894210
1106487521725145090
1106488954784763904
1106489002738081792
1106493724148801538
1106494087518187521

```

In case you have collected the tweetset with `Twarc`, dehydration can be implemented with a single command.

```
twarc dehydrate hellobrother.jsonl > hellobrother_idlist.txt
```

More details on installing and operating Twarc will be covered later on in this tutorial.

Before the next step of *rehydrating* the tweetset, we will first set up the technical environment. This includes getting access to the Twitter API and installing Python, Twarc, and Jupyter Notebooks.

2 Twitter developer access

The first step in getting access to the Twitter API is to create a Twitter account and apply for a developer account at developer.twitter.com. Depending on your status, you can either apply for standard access or, should you qualify, for the [Academic Research product track](#).

Please note that it might take days or even weeks to get the developer account. Therefore, we urge you to follow the instructions in the How-to Guide for Python to submit your application as early as possible.

Once your application is accepted, you can create a new App that comes with the API access credentials. In the [Developer portal Overview](#), select Create App, give it a name, and move to Keys & Tokens. The next note is important: copy the API Key, API Secret, and Bearer Token to a local text file for now because, for security reasons, they will only appear once when created and you will need them later.

Next, let's set up the Python environment so that we are able to use our newly created credentials.

3 Implementation in Python

3.1 Installing Python

Python is an open source programming language [freely available](#) to developers. Over the last few years, Python has gained major popularity in the machine learning and data science communities. There are several ways to make use of Python in data science. We use [Jupyter Notebooks](#) here. A form of literate programming (Keys et al., 2018), these analytical notebooks combine textual documentation with Python code and its outputs, including data listing and visualisations.

There are several alternative approaches for setting up a Python environment. [Google Colaboratory](#) provides a cloud-based option that does not require installing software on your local computer. However, if you are able to install software on your computer, we recommend using [Anaconda](#), a data science-specific Python platform that includes not only Jupyter Notebooks but a curated set of Python packages needed in this tutorial. The getting-started process varies between different operating environments so we recommend following the [Getting started with Anaconda tutorial](#).

Once you have Anaconda installed, open Anaconda Navigator and fire up a new notebook.

Alternatively, for a more advanced option you can install a Python virtual environment and use the requirements.txt file provided in the [Github repository of this tutorial](#) to install all the prerequisites needed to run the code for this tutorial.

We admit, configuring and managing a Python environment may seem unnecessarily complicated. However, the good thing is that once set up, you can keep using the environment for your upcoming analysis projects.

3.2 Rehydrating the first tweet

Now that we have Python installed, we are ready to try out the Twitter API. Here, you can either run the notebook provided in this tutorial or copy the necessary code to your own notebook.

To get to know the details of the API, we use [Twython](#) to access the Twitter API here instead of Twarc. Whereas Twarc is a full-blown command line application that allows running entire data collection processes, Twython is a Python package that a developer can use to access the API more easily. First, let's install Twython. Enter and run the following command in the Jupyter notebook:

```
pip install twython
```

Now we can import Twython so that we can start using the library:

```
from twython import Twython
```

Next, we need to get Twitter API access. For this, you will use the two credentials, API Key and API Secret, from the Twitter App that you already received earlier on. The two remaining credentials, Access Token and Access Secret, can be accessed from the App page under Keys and Tokens. You can always regenerate all four keys if you misplace them.

Next, create the following four variables and assign the credentials as their values. That is, copy the lines to your notebook and replace each of the four `copy_from_Twitter` placeholders with your own API key/secret value:

```
api_key = 'copy_from_Twitter'
api_secret = 'copy_from_Twitter'
access_token = 'copy_from_Twitter'
access_secret = 'copy_from_Twitter'
```

Note that the actual values are a lot longer. Now that we have the values set, we can create a Twython instance:

```
twitter = Twython(api_key, api_secret, access_token, access_secret)
```

Now, we should be ready to rehydrate our first tweet. We will use one of the tweets by New Zealand Prime Minister Jacinda Ardern as an example here:

```
tweet = twitter.show_status(id='1106397870628847617', include_entities=True)
```

The `id` parameter takes the tweet ID and the `include_entities` value `True` asks the Twitter API to include further information about the entities in each tweet, including hashtags and users mentioned.

To view the results of our API query, you can simply write the variable name `tweet` to the notebook cell and run it:

```
tweet
```

This is an excerpt from the expected output:

```
{
  'created_at': 'Fri Mar 15 03:33:02 +0000 2019',
  'id': 1106397870628847617,
  'id_str': '1106397870628847617',
  'text': 'What has happened in Christchurch is an extraordinary act of
unprecedented violence. It has no place in New Zealand...
https://t.co/XOHF9hHe8H',
  ...
}
```

Use the `keys()` function for a full list of details for the tweet:

```
tweet.keys()
```

Output:

```
dict_keys(['created_at', 'id', 'id_str', 'text', 'truncated', 'entities',
'source', 'in_reply_to_status_id', 'in_reply_to_status_id_str',
'in_reply_to_user_id', 'in_reply_to_user_id_str', 'in_reply_to_screen_name',
'user', 'geo', 'coordinates', 'place', 'contributors', 'is_quote_status',
'retweet_count', 'favorite_count', 'favorited', 'retweeted', 'lang'])
```

As we can see, there is plenty of metadata for any single tweet. Now that we are more familiar with the basics of Twitter API, we are ready to rehydrate our full dataset.

3.3 Rehydrating the full dataset

In principle, rehydrating the full tweetset simply means that you implement a simple loop that runs the process we implemented in the previous phase for each tweet separately. However, there are nuances one has to consider, including the rate limits that the Twitter API sets (<https://developer.twitter.com/en/docs/twitter-api/rate-limits>) and the management of large datasets of data represented in JSON.

To minimise the risk of errors in the rehydration process, we will use Twarc, a commonly used Python-based tool for archiving and rehydrating Twitter data. Compared to a low-level

Python developer framework, Twarc is a command line application designed specifically to collect, rehydrate, and manage large datasets. To install and learn how to use Twarc, we highly recommend the [Collect Twitter Data with Twarc! tutorial](#). Do note that the Twarc tutorial covers the basics of command-line use in MacOS and Windows as well as gives tips on how to set up Twitter and Python.

If you chose to install a dedicated Python virtual environment, you can either install Twarc on the same Python environment that you use for the notebook installation or create a separate one. We prefer to use a project-specific virtual environment and therefore we would install Jupyter, Twarc, and the other Python packages all in the same virtual environment. The important thing is that in order to process the data in the next phase, you may need to copy the rehydrated data to the folder where your notebook operates.

The easiest way to install Twarc is to use the Python package manager pip on the command line:

```
pip install twarc
```

Next, we have to provide Twarc with the same Twitter API credentials we used to read the first tweet. To do that, run Twarc configuration on the command line:

```
twarc configure
```


If you have previously used Twarc, you may be asked if you would like to use your existing keys. Please answer either y(es) or n(o). If you create several profiles, please specify the one you are using when running the script. More on this later.

Insert your API key as a consumer key and your API secret as a consumer secret when prompted, then select “generate access keys by visiting Twitter.” Copy the provided URL to your browser and select the Authorize app option. Alternatively, you can insert all the four key values through the prompt.

Authorize First collector to access your account?

Authorize app

Cancel



First collector
This app was created to use the Twitter API.

This application will be able to:

- See Tweets from your timeline (including protected Tweets) as well as your Lists and collections.
- See your Twitter profile information and account settings.
- See accounts you follow, mute, and block.

Learn more about third-party app permissions in the [Help Center](#).

Finally, copy the PIN number provided back to the command line to finalise the authentication process.

Now, we should be ready to rehydrate the tweetset. Make sure that the text file including the list of IDs is placed in the folder where you run the script. Run Twarc on the command line:

```
twarc hydrate hellobrother_idlist.txt > hellobrother_rehydrated.jsonl
```

You can skip using the `--profile` parameter if you just installed Twarc for the first time.

```
twarc --profile ProfileName hydrate hellobrother_idlist.txt >
hellobrother_rehydrated.jsonl
```

Rehydrating tweets and exploring the contents of the output may insist on moving between the command line and the Jupyter notebook. To simplify the process, you can run Twarc inside the Jupyter notebook environment simply by adding an exclamation mark (!) at the beginning of the line and run it as a notebook cell:

```
!twarc hydrate hellobrother_idlist.txt > hellobrother_rehydrated.jsonl
```

Note that the rehydration process will take a while. Once the script is completed, we can move to exploring the results of the process, that is, the full rehydrated tweetset.

4 Exploring the results

We are now ready to explore the rehydrated dataset represented in JSON and take the first steps toward reconstructing the ethnographic data field (see Student Guide; see also Harju & Huhtamäki, 2021) for qualitative research.

We will process the data in JSON and use Pandas to manage it.

```
import pandas as pd
import json
```

In case importing Pandas fails, you can install it:

```
pip install pandas
```

The first processing step is to read the data and create a Pandas DataFrame. Twarc uses JSON Lines format to represent the tweets. That is, each tweet object is represented as one line of JSON. This eases managing large data files as they can be processed one record at a time with Unix-style command line tools.

Run the following lines:

```
with open('helloworld_rehydrated.jsonl') as f:
    tweets_jsonl = f.read()

df_hydrated = pd.DataFrame([json.loads(jline) for jline in
    tweets_jsonl.splitlines()]])
```

We first read each line of the tweetset, open them as JSON objects, and finally create a Pandas DataFrame of these objects.

Again, the info() function gives us an overview of the data frame.

```
df_hydrated.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25603 entries, 0 to 25602
Data columns (total 31 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   created_at                            25603 non-null  object
1   id                                     25603 non-null  int64
2   id_str                                25603 non-null  object
3   full_text                             25603 non-null  object
4   truncated                             25603 non-null  bool
5   display_text_range                    25603 non-null  object
6   entities                              25603 non-null  object
7   source                                25603 non-null  object
8   in_reply_to_status_id                  1713 non-null   float64
9   in_reply_to_status_id_str              1713 non-null   object
10  in_reply_to_user_id                     1796 non-null   float64
```



```

11 in_reply_to_user_id_str    1796 non-null    object
12 in_reply_to_screen_name   1796 non-null    object
13 user                      25603 non-null   object
14 geo                       82 non-null      object
15 coordinates               82 non-null      object
16 place                    1323 non-null    object
17 contributors              0 non-null       object
18 is_quote_status           25603 non-null   bool
19 quoted_status_id          2463 non-null    float64
20 quoted_status_id_str       2463 non-null    object
21 quoted_status_permalink    2463 non-null    object
22 retweet_count             25603 non-null   int64
23 favorite_count            25603 non-null   int64
24 favorited                 25603 non-null   bool
25 retweeted                 25603 non-null   bool
26 possibly_sensitive        15538 non-null   object
27 lang                      25603 non-null   object
28 quoted_status             2055 non-null    object
29 extended_entities         10361 non-null   object
30 withheld_in_countries     8 non-null       object
dtypes: bool(4), float64(3), int64(3), object(21)
memory usage: 5.4+ MB

```

As mentioned, the ethnographic data field degrades over time. Let's calculate the proportion of tweets removed since they were first collected:

```
len(df_hydrated.index)/43659*100
```

We now see that we have lost more than 40% of the tweets.

```
58.643120547882454
```

Now that we have successfully rehydrated the dataset, we have two main options to choose from. We can either continue processing the data in Python or export the data in CSV or as a spreadsheet for importing it to your favorite analysis tool, including ATLAS.ti, SPSS, Tableau, or Excel. Most of the analysis processes combine these two approaches as data preprocessing is often more expressive in Python. Let's export the data first:

```
df_hydrated.to_csv('helloworld_rehydrated.csv', encoding='utf-8')
```

In addition to specifying an encoding that can represent characters beyond the regular a-z, the [to_csv\(\) function](#) takes several different parameters. Setting the field delimiter from comma (',') to tabulator ('\t'), for example, may be of use when importing the data to a tool that insists on a specific data format.

In case the research team is using Microsoft Excel to review the data, creating a spreadsheet in XLSX format may be useful:

```
df_hydrated.to_excel('helloworld_rehydrated.xlsx')
```

For XLSX support, you need to install openpyxl:

```
pip install openpyxl
```

A practical benefit for the Excel-specific data representation is the ability to simply double-click a file to open whereas opening CSV data in Excel insists on importing it.

One way to explore the tweetset is to look at the distribution of hashtags. However, it is important to note that constructing a data field for ethnographic research goes way beyond these types of simplistic representations and includes interpretative analysis that considers the context and the event in a more holistic manner.

```
df_hydrated.entities.head()
```

To see the full contents of the cell, you can ask Pandas not to truncate:

```
pd.set_option('display.max_colwidth', None)
```

To create a simple list of hashtags for each tweet, we can create a simple function that extracts the hashtags as text:

```
def extract_hashtags(entities):  
    hashtags = list()  
  
    for hashtag in entities['hashtags']:  
        hashtags.append(hashtag['text'])  
  
    return hashtags
```

Do note that you have to run the lines of code to enable the use of the function. We can now use the apply() function of Pandas DataFrame to run the function for each row.

```
df_hydrated['hashtags'] = df_hydrated.entities.apply(extract_hashtags)  
df_hydrated.hashtags
```

Next, we will create another function to concatenate the list of hashtags for each tweet into a list of hashtags for the full tweetset. Moreover, we will transform the hashtags to lowercase text. This is needed because, unlike Twitter, Python treats hashtags in a case-sensitive manner.

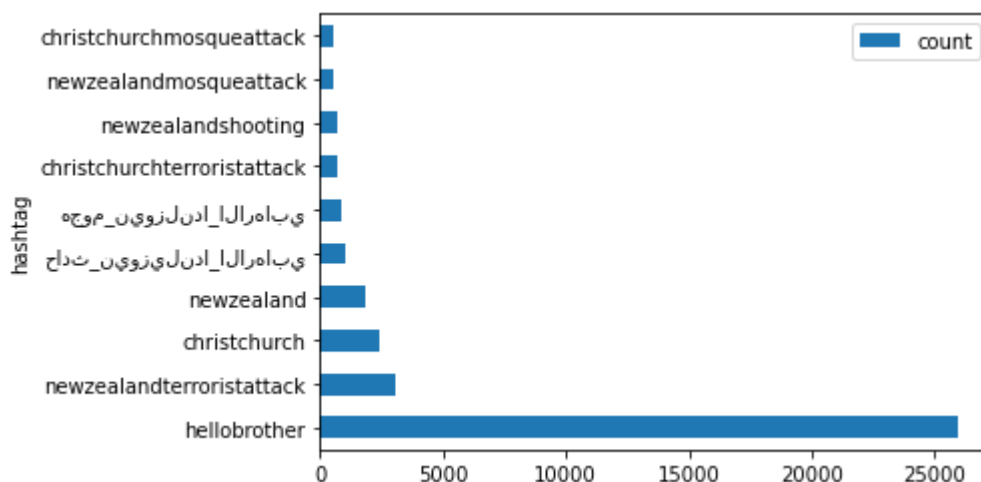
```
def to_1D(series):  
    return pd.Series([x.lower() for _list in series for x in _list])
```

The `to_1D()` function is part of a [tutorial on dealing with list values in Pandas DataFrames](#). You can figure out the details of this process by exploring the outputs of each of the phases.

```
df_hashtags = pd.DataFrame(to_1D(df_hydrated.hashtags).value_counts(),
columns=['count'])
df_hashtags['hashtag'] = df_hashtags.index
```

Now we have a DataFrame of hashtags and their frequency in the tweetset. Pandas introduces straightforward means to plot a histogram of hashtags.

```
df_hashtags.head(10).plot.barh(y='count', x='hashtag')
```



Plotting the histogram of Twitter accounts sending tweets and receiving mentions might be the next logical step in reconstructing the field.

5 Your turn

1. First, apply for a Twitter Developer account to get access to Twitter API
2. Install the necessary development tools, including Python, Jupyter Notebooks or JupyterLab, and Twarf
3. Explore the Twitter API by searching and/or rehydrating tweets
4. Rehydrate the provided dataset
5. Compare the rehydrated dataset to the statistics of our original and rehydrated datasets
6. Explore Twitter datasets available for rehydration, see for example what is available at [Zenodo](#) and [Harvard Dataverse](#).