Processing large amounts of Twitter data

Sometimes, the data we have to process reaches a size that is too much for a computer's memory to handle. This is a common problem faced by data scientists. A solution to this is to process an entire data source chunk by chunk, instead of as a single go at all once.

In this exercise, you will do just that. You will process a large CSV file of Twitter data in the same way that you processed tweets.csv in Bringing it all together exercises of the previous course, but this time, working on it in chunks of 10 rows at a time.

If you are interested in learning how to access Twitter data, you can do so with the help of our system. Refer to part 2 of the DataCamp course on importing data in Python.

Instructions

1. Initialize an empty dictionary, counts_dict, for storing the results of processing Twitter data.

2. Iterate over the file tweets.csv file using a for loop. Use the loop construct along with pandas.read_csv() which takes the chunksize=10.

3. In the inner logic, iterate over the column lang in each chunk. Use the entry variable.

Python Code

```
import pandas as pd

# Initialize an empty dictionary: counts_dict
counts_dict = {}

# Iterate over the file chunk by chunk
for chunk in pd.read_csv('tweets.csv', chunksize=10):

    # Iterate over the column in DataFrame
    for entry in chunk['lang']:
        if entry in counts_dict.keys():
            counts_dict[entry] += 1
        else:
            counts_dict[entry] = 1

# Print the populated dictionary
```

```
print(counts_dict)
```

Explanation

1. Initialization:
- counts_dict is an empty dictionary initialized to store language counts from the CSV file.

2. Reading Data in Chunks:
- The pd.read_csv() function reads the CSV file tweets.csv in chunks of 10 rows using chunksize=10.
- This ensures that only 10 rows of data are loaded into memory at any time, making it memory-efficient for large datasets.

3. Processing Each Chunk:
- For each chunk (a DataFrame of 10 rows), the code iterates over the lang column using a nested for loop.
- For each entry (language code) in the lang column:
  - If the language code exists in counts_dict, its count is incremented by 1.
  - Otherwise, the language code is added to counts_dict with an initial count of 1.

4. Output:
- After processing all chunks, the final counts_dict dictionary contains the total counts of each language in the entire dataset.
- The result is printed at the end.