# Zehrish Iqbal

**You said:**

I don’t understand how to not count zeros. please explain

**ChatGPT said:**

Great question!

When you say you **don’t want to count zeros**, you mean:

**Don't include any place–month combination in your name\_frequency\_by\_month dictionary if that place had zero mentions in that article.**

Let’s walk through how to **avoid counting zeros** step by step.

**❌ What's currently happening?**

Right now, for **every** pattern in your patterns dictionary, you're doing this:

python

Copy code

matches = re.findall(pattern, text, re.IGNORECASE)

count = len(matches)

# EVEN IF count is 0, you still do this:

if place not in name\_frequency\_by\_month:

name\_frequency\_by\_month[place] = {}

if month\_str not in name\_frequency\_by\_month[place]:

name\_frequency\_by\_month[place][month\_str] = 0

name\_frequency\_by\_month[place][month\_str] += count # count could be 0!

So if a place name doesn't appear at all in a file (i.e., count = 0), you still initialize it and set its count to 0. This creates noisy output and unnecessary dictionary entries.

**✅ How to fix it?**

👉 Add a check to skip adding the count if it’s zero.

**✅ Just before updating the dictionary, add:**

python

Copy code

if count == 0:

continue

**🔁 Updated code snippet:**

python

Copy code

for place in patterns:

pattern = patterns[place]["pattern"]

matches = re.findall(pattern, text, re.IGNORECASE)

count = len(matches)

if count == 0:

continue # Skip if no matches found — do NOT add zero counts

patterns[place]["count"] += count

month\_str = date\_str[:7]

if place not in name\_frequency\_by\_month:

name\_frequency\_by\_month[place] = {}

if month\_str not in name\_frequency\_by\_month[place]:

name\_frequency\_by\_month[place][month\_str] = 0

name\_frequency\_by\_month[place][month\_str] += count

**✅ Result:**

This way, your name\_frequency\_by\_month dictionary will **only include place names that were actually mentioned at least once** in a given month.

**Script 2:**

My code successfully removes the zeros from the output but the count is still not correct. it is adding up the count of all various spelling separately

**ChatGPT said:**

The issue you're encountering is likely due to the way you're handling the counting process for multiple alternate spellings of a place. Since you are combining various spellings (like alternate names) into one regular expression for each place, but you are still counting them separately for each alternate spelling, the final count may not be correct.

**Problem:**

When you match the place names, you are using a regex pattern that combines all the alternate names into one pattern. However, the way the count is being accumulated could be incorrectly adding the counts from each individual alternate name separately.

**Solution:**

To fix this, you need to ensure that:

1. The count for a particular place (considering all its alternate spellings) is aggregated properly.
2. The regex should match all variations of the place name at once, but you should only update the count for that place once, regardless of how many alternate spellings matched.

**Code Adjustment:**

Instead of adding the counts for each alternate name separately, you should:

* Use the compiled regex to match any of the alternate names in the text.
* Add the total matches for that place (regardless of which specific spelling matched) to the count for that place in patterns and name\_frequency\_by\_month.

Here's the updated approach to fix the count logic:

**Updated Code:**

python

Copy code

import re

import os

folder = "../articles"

path = "../gazetteers/geonames\_gaza\_selection.tsv"

# Reading the file

with open(path, encoding="utf-8") as file:

data = file.read()

patterns = {}

# Splitting the gazetteer data into rows

rows = data.split("\n")

for row in rows[1:]:

columns = row.split("\t")

if len(columns) < 6:

continue # Skip rows with less than 6 columns

asciiname = columns[0]

alternatenames = columns[5].strip()

detailed\_list = [asciiname]

if alternatenames:

detailed\_list.extend([alternate.strip() for alternate in alternatenames.split(",")])

regex\_pattern = re.compile(r"\b(" + "|".join(detailed\_list) + r")\b", re.IGNORECASE)

patterns[asciiname] = {"pattern": regex\_pattern, "count": 0}

# Initialize an empty dictionary for name frequency by month

name\_frequency\_by\_month = {}

war\_begin\_date = "2023-10-07"

for filename in os.listdir(folder):

date\_str = filename.split("\_")[0]

if date\_str < war\_begin\_date:

continue

file\_path = os.path.join(folder, filename)

with open(file\_path, encoding="utf-8") as file:

text = file.read()

for asciiname, pattern\_data in patterns.items():

pattern = pattern\_data["pattern"]

matches = re.findall(pattern, text)

count = len(matches)

if count == 0:

continue

# Update the count for the place (asciiname)

patterns[asciiname]["count"] += count

month\_str = date\_str[:7]

# Manually check if the place exists in the dictionary

if asciiname not in name\_frequency\_by\_month:

name\_frequency\_by\_month[asciiname] = {}

if month\_str not in name\_frequency\_by\_month[asciiname]:

name\_frequency\_by\_month[asciiname][month\_str] = 0

# Add the count to the monthly count for the place

name\_frequency\_by\_month[asciiname][month\_str] += count

# Output the frequency by month for each place

for place in name\_frequency\_by\_month:

print(f'"{place}":{{')

month\_list = list(name\_frequency\_by\_month[place].keys())

for month in month\_list:

count = name\_frequency\_by\_month[place][month]

if month != month\_list[-1]:

print(f' "{month}": {count},')

else:

print(f' "{month}": {count}')

print("},")

**Key Points:**

1. **Regex Pattern**:
   * The pattern is built using all the alternate names (including the primary name) using | to match any of them in the text.
   * This way, the regex matches any occurrence of any spelling of the place name.
2. **Counting**:
   * When re.findall is used, it counts all matches of any alternate name in the text (so, for example, if both "Gaza" and "Gaza City" are found, they are counted together).
   * The count is updated in the patterns[asciiname]["count"] to ensure you're counting all occurrences of that place name correctly.
3. **Adding to Monthly Count**:
   * The count for each place is added to the name\_frequency\_by\_month dictionary for the relevant month.

**Expected Behavior:**

* The total count for a place will now correctly include matches from all its alternate spellings, instead of counting them separately.
* The output will reflect the correct counts for each place across the relevant months.

Let me know if this resolves your issue!

4o mini

**You said:**

import re # built-in library for regex import os # built-in library for os-dependent functions #define folder folder= "../articles" #setting path for loading gazatteer tsv file path = "../gazetteers/geonames\_gaza\_selection.tsv" #reading the file with open(path, encoding="utf-8") as file: data = file.read() #building dictionary of patterns for place names and a count of matches patterns = {} #split the gazatteer data into different rows using newline character rows = data.split("\n") for row in rows[1:]: #skipping the header columns = row.split("\t") #separating the columns using tab if len(columns) < 6: continue # skip the row if it dooes not have atleast 6 colunms asciiname = columns[0] alternatenames = columns[5].strip() detailed\_list=[asciiname] #starting to add the first column to the larger list #split column[5] by comma and add various spellings from col[5] to larger list if alternatenames: detailed\_list.extend ([alternate.strip() for alternate in alternatenames.split(",")]) #build a single regex pattern that matches any alternate using '|' for alterations regex\_pattern = r"\b("+"|".join(alternatenames) + r")\b" patterns[place] = {"pattern": regex\_pattern, "count":0} #creating a dictionary to count the number of each name occurred per month name\_frequency\_by\_month = {} #set the starting date of war to filter articles with names occured after that war\_begin\_date= "2023-10-07" #looping through each file to count the occurrences of each pattern across entire folder for filename in os.listdir(folder): #extracting the date from the filename(as the format is YYYY-MM-DD date\_str = filename.split("\_")[0] #skip the file if it is before the start of the war if date\_str < war\_begin\_date: continue #build filepath for the current articles file\_path = os.path.join(folder, filename) #open and read the articles with open(file\_path, encoding="utf-8")as file: text = file.read() #looping through places to search for matches in the text for place, pattern\_data in patterns.items(): pattern = pattern\_data["pattern"] #getting regex pattern matches = re.findall(pattern, text, re.IGNORECASE) count = len(matches) #number of times the place is occured if count==0: continue #adding the number of times places found to the total frequency patterns[asciiname]["count"]+=count #extracting the month from the string for dates month\_str= date\_str[:7] #initializing place and month in name\_frequency\_by\_month dictionary if place not in name\_frequency\_by\_month: name\_frequency\_by\_month[place]={} if month\_str not in name\_frequency\_by\_month[place]: name\_frequency\_by\_month[place][month\_str]=0 #adding the new matches on the place names to the number of times it was found that month name\_frequency\_by\_month[place][month\_str] += count #loop through each place in the name\_frequency\_by\_month dictionary for place in name\_frequency\_by\_month: print(f'"{place}":{{') #get a list of all the months in which the place names are mentioned month\_list = list(name\_frequency\_by\_month[place].keys()) #loop through each month to print the corresponding mention count for month in month\_list for month in month\_list: count = name\_frequency\_by\_month[place][month]#find count for that month #display the output by a comma except for the last item to keep it clean if month!= month\_list[-1]: print(f' "{month}":{count},') else: print(f' "{month}":{count}') print("},") #close the dictionary block and print the output

**ChatGPT said:**

You're getting close, but there are a few **important bugs** in your current script that are causing your counts to be wrong:

**✅ Fixes You Need:**

**1. Incorrect variable in regex pattern creation:**

You're doing this:

python

Copy code

regex\_pattern = r"\b(" + "|".join(alternatenames) + r")\b"

But alternatenames is just a string. You should be using detailed\_list, which contains the asciiname and all alternate spellings **as a list**:

✅ Replace with:

python

Copy code

regex\_pattern = r"\b(" + "|".join(map(re.escape, detailed\_list)) + r")\b"

* This also escapes special characters that may appear in names (like . or (), to avoid regex issues.

**2. Wrong key in patterns[place] = ...:**

You’re trying to store the regex under patterns[place], but place is **undefined** at that point in your loop.

✅ Replace with:

python

Copy code

patterns[asciiname] = {"pattern": regex\_pattern, "count": 0}

**3. Using the wrong key (asciiname) while updating counts inside loop:**

Inside the loop where you read article files, you're iterating over:

python

Copy code

for place, pattern\_data in patterns.items():

So you should be using place, **not** asciiname.

But you wrote:

python

Copy code

patterns[asciiname]["count"] += count

✅ Fix that to:

python

Copy code

patterns[place]["count"] += count

**✅ Final Cleaned-Up Version of Your Code:**

python

Copy code

import re

import os

folder = "../articles"

path = "../gazetteers/geonames\_gaza\_selection.tsv"

with open(path, encoding="utf-8") as file:

data = file.read()

patterns = {}

rows = data.split("\n")

for row in rows[1:]:

columns = row.split("\t")

if len(columns) < 6:

continue

asciiname = columns[0]

alternatenames = columns[5].strip()

detailed\_list = [asciiname]

if alternatenames:

detailed\_list.extend([alt.strip() for alt in alternatenames.split(",")])

# Build escaped regex pattern

regex\_pattern = r"\b(" + "|".join(map(re.escape, detailed\_list)) + r")\b"

patterns[asciiname] = {"pattern": regex\_pattern, "count": 0}

name\_frequency\_by\_month = {}

war\_begin\_date = "2023-10-07"

for filename in os.listdir(folder):

date\_str = filename.split("\_")[0]

if date\_str < war\_begin\_date:

continue

file\_path = os.path.join(folder, filename)

with open(file\_path, encoding="utf-8") as file:

text = file.read()

for place, pattern\_data in patterns.items():

pattern = pattern\_data["pattern"]

matches = re.findall(pattern, text, re.IGNORECASE)

count = len(matches)

if count == 0:

continue

patterns[place]["count"] += count

month\_str = date\_str[:7]

if place not in name\_frequency\_by\_month:

name\_frequency\_by\_month[place] = {}

if month\_str not in name\_frequency\_by\_month[place]:

name\_frequency\_by\_month[place][month\_str] = 0

name\_frequency\_by\_month[place][month\_str] += count

# Output

for place in name\_frequency\_by\_month:

print(f'"{place}":{{')

month\_list = list(name\_frequency\_by\_month[place].keys())

for month in month\_list:

count = name\_frequency\_by\_month[place][month]

if month != month\_list[-1]:

print(f' "{month}": {count},')

else:

print(f' "{month}": {count}')

print("},")

**✅ This will now:**

* Correctly build regex for all name variants.
* Escape problematic regex characters.
* Use the right dictionary keys for pattern creation and updating counts.
* Aggregate all name variants under their canonical name (asciiname).

Would you like to test this on a specific sample file or output format (e.g., JSON or CSV)?

**Script 3:**

so, here is my script for my task of mapping, for which the instructions are following : "4A. Map the regex-extracted placenames Use plotly.express to map the frequency of the place names extracted using regex (which you stored in regex\_counts.tsv) onto an interactive, animated map (one frame per month). You can experiment with how to best display the frequency of these places; please explain in your README file what you tried and why you think the approach you took is the best. Save your map in html (interactive) and png (image) format: regex\_map.html and regex\_map.png." i have written the whole script but there is a issue while merging the two files (regex\_counts.tsv and the gazetteer/geonames\_gaza\_selection.tsv). this is because there is no common column between the files. the regex\_counts.tsv file have the colunm name placename and the gazetter one has asciiname. so i want you to tell me that how can i merge two dataframes with pandas if they don't have a column name in common.

**ChatGPT said:**

To merge two dataframes in **pandas** where the **columns to match on have different names**, you **explicitly specify** them using the left\_on and right\_on arguments in pd.merge().

Here’s exactly how you do it:

python

CopyEdit

import pandas as pd

# Load the data

regex\_df = pd.read\_csv("regex\_counts.tsv", sep="\t")

gazetteer\_df = pd.read\_csv("geonames\_gaza\_selection.tsv", sep="\t")

# Merge using placename (from regex\_df) and asciiname (from gazetteer\_df)

merged\_df = pd.merge(regex\_df, gazetteer\_df, left\_on="placename", right\_on="asciiname", how="inner")

This merges the two dataframes where regex\_df["placename"] == gazetteer\_df["asciiname"].

**A few things to watch out for:**

1. **Case sensitivity**: "gaza" and "Gaza" will not match. Normalize casing first:

python

CopyEdit

regex\_df["placename"] = regex\_df["placename"].str.lower()

gazetteer\_df["asciiname"] = gazetteer\_df["asciiname"].str.lower()

1. **Stripping whitespace**:

python

CopyEdit

regex\_df["placename"] = regex\_df["placename"].str.strip()

gazetteer\_df["asciiname"] = gazetteer\_df["asciiname"].str.strip()

1. **Check unmatched values** (optional but helpful during debugging):

python

CopyEdit

unmatched = regex\_df[~regex\_df["placename"].isin(gazetteer\_df["asciiname"])]

print(unmatched)

1. **If you're using fuzzy matching**, let me know — you'll need a different approach (e.g., fuzzywuzzy or rapidfuzz).

Would you like help refining the merged data to ensure good matches for mapping (e.g., including lat/lon columns from the gazetteer)?

4o

Top of Form

Bottom of Form