Zainab Murtaza Ali part 2A

1. okay i am a student of digital humanities and i am doing a project where i have to write a python code but im a little stuck so help me #imports regular expression to find text patterns import re # imports to enable interaction with file system import os #imports to hadle tabular data and export tsv file import pandas as pd # function writes a list of data rows intotsv file using panadas def write\_tsv(rows, column\_list, path): #list of rows is converted into panads DataFrame df = pd.DataFrame(rows, columns=column\_list) #DataFrame is written into tsv df.to\_csv(path, sep="\t", index=False) #Setup paths #defining folder where articles are present repo\_path = "FASDH25-portfolio2" folder = "C:\Users\batoo\Downloads\FASDH25-portfolio2\articles" #define path and load gazetteer from the tsv file having place names and alternate names gazetteer\_path = "C:\Users\batoo\Downloads\FASDH25-portfolio2\gazetteers\geonames\_gaza\_selection.tsv" my folder and path are not working for some reason tell me what ive done wrong

**ChatGPT said:**

You're on the right track, Zainab! The issue you're running into is likely due to **incorrect formatting of file paths in Windows**. In Python strings, the backslash (\) is an **escape character**, so you either need to:

**✅ Option 1: Use raw strings (recommended for Windows paths)**

Prefix the string with r to make it a *raw string* so that backslashes are treated literally:

python

CopyEdit

folder = r"C:\Users\batoo\Downloads\FASDH25-portfolio2\articles"

gazetteer\_path = r"C:\Users\batoo\Downloads\FASDH25-portfolio2\gazetteers\geonames\_gaza\_selection.tsv"

**✅ Option 2: Use double backslashes**

Escape each backslash manually:

python

CopyEdit

folder = "C:\\Users\\batoo\\Downloads\\FASDH25-portfolio2\\articles"

gazetteer\_path = "C:\\Users\\batoo\\Downloads\\FASDH25-portfolio2\\gazetteers\\geonames\_gaza\_selection.tsv"

Either of these will fix the "path not working" issue.

Would you like help writing the rest of the code to process the gazetteer and find place mentions?

4o

**You said:**

1. im having a little problem in writing my regex pattern i think either something is missing because its not giving me results can you pls check and tell me what is wrong? dont give me the direct answer but help me in how it can be better #imports regular expression to find text patterns import re # imports to enable interaction with file system import os #imports to hadle tabular data and export tsv file import pandas as pd # function writes a list of data rows intotsv file using panadas def write\_tsv(rows, column\_list, path): #list of rows is converted into panads DataFrame df = pd.DataFrame(rows, columns=column\_list) #DataFrame is written into tsv df.to\_csv(path, sep="\t", index=False) #Setup paths #defining folder where articles are present repo\_path = "FASDH25-portfolio2" folder = r"C:\Users\batoo\Downloads\FASDH25-portfolio2\articles" # used chatgpt to add "r" before the path so it works #define path and load gazetteer from the tsv file having place names and alternate names gazetteer\_path = r"C:\Users\batoo\Downloads\FASDH25-portfolio2\gazetteers\geonames\_gaza\_selection.tsv" #Read Gazetteer File with open(gazetteer\_path, encoding="utf-8") as file: data = file.read() #an empty dictionary is created for each place names and a count for matches patterns = {} #split gazetteer data in new lines to get eeach row on file rows = data.split("\n") #Skip header because the pattern starts from next row for row in rows[1:]: #seprates each column in tsv by tabs columns = row.split("\t") #ensures that first column has names for the place asciiname = columns[0] #lists name variants consisting of the standard name name\_variants = [asciiname] #gets the alternate names from the 6th column which is counted as the 5th column, if present alternate\_names = columns[5].strip() if alternate\_names: #splits the alternate names with comma and gets list of other names alternate\_list = alternate\_names.split(",") #loops through each alternate name in the list for name in alternate\_list: #.strip will remove whitespace from alternate names and .append will add alternate names to the list if prenest name\_variants.append(name.strip()) # build a regex pattern that will work to find all names and match diffrent varianats of the place names aswell #using re.escape to escape any special characters in place names and "|" is used for alternation regex\_pattern = "|"join (re.escape(name) ) #includes all names and their variants with numbers patterns[asciiname] = {"pattern": regex\_pattern, "count": 0} this is my code uptil now but the regex part is giving problems

**ChatGPT said:**

You're very close, and your logic makes sense overall — but there are **two key things** going wrong in your regex pattern construction:

**🔍 1. Syntax mistake in your join statement**

This line is not written correctly:

python

CopyEdit

regex\_pattern = "|"join (re.escape(name) )

Issues:

* You're missing **parentheses** around the iterable you're joining.
* You're passing **a single string** (re.escape(name)) rather than a list.
* You probably meant to join **all the escaped variants** (name\_variants), not just name.

**🧠 Ask Yourself:**

* What *type* of object does join() work on?
* Are you escaping all variants?
* Are you combining them correctly with "|" to match **any** one of the variants?

**🛠 Suggestion (without directly giving the full code):**

* Use a **list comprehension or loop** to apply re.escape() to every item in name\_variants.
* Then use "|".join(...) to create a single pattern string.
* That pattern should look like: r"Name1|Name2|Name3" — so the regex engine can match **any one of the names**.

Would you like to test it out and share what you try next?

4o

**You said:**

okay so basically i have forgoten to write name.variants which is essential?

**ChatGPT said:**

Exactly! Yes — you *forgot to use* name\_variants, which is essential because that list contains:

* the main place name (asciiname)
* all its **alternate names**

And then, instead of just escaping one name, you need to escape **each name in name\_variants**, and combine them using "|" so the regex can match **any one of them**.

**Think of it this way:**

If you just write:

python

CopyEdit

re.escape(name)

You're escaping only **one** name. But what you actually want is:

python

CopyEdit

[re.escape(n) for n in name\_variants]

Then you can join those with "|" to create the full pattern:

python

CopyEdit

"|".join(...)

So yes — using name\_variants is crucial to ensure you're searching for **all possible ways** that a place could be mentioned in the articles.

Do you want to try fixing that part and share what you write?

4o

**You said:**

regex\_pattern = "|"join (re.escape(name) for name in name\_variants) okay i have written this but its giving me syntax error

**ChatGPT said:**

You're really close again — just a tiny **syntax mistake** in how you're using .join().

**🔧 Problem:**

You're missing **parentheses** around the iterable passed to join().

This is invalid:

python

CopyEdit

"|"join (re.escape(name) for name in name\_variants)

**✅ Correct Syntax:**

python

CopyEdit

regex\_pattern = "|".join(re.escape(name) for name in name\_variants)

"separator".join(iterable) is the correct format — no space between "|" and .join, and the generator expression goes directly inside the parentheses.

Once you fix that, your pattern will be a proper regex string like:

arduino

CopyEdit

"Rafah|Rafa|Rafāḥ"

Let me know if you want help testing the pattern on some sample text!

4o

1. **You said:**

matches = re.findall(pattern, text) this is one of the code in my script that i wrote but its not working without giving me the output tell me what ive done wrong

**ChatGPT said:**

Good question! The line you wrote is fundamentally correct in structure, but based on your earlier code, there’s **one likely reason** why it's not working:

**⚠️ Possible Issue:**

You're using:

python

CopyEdit

matches = re.findall(pattern, text)

But the variable pattern may not be defined in this context.

From your earlier code, you stored each regex like this:

python

CopyEdit

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

So your patterns are inside a **dictionary** called patterns. That means pattern isn't a standalone variable — you’d need to **loop through the dictionary** to get each pattern like this:

**✅ What to check:**

* Are you doing something like this before your findall()?

python

CopyEdit

for place, data in patterns.items():

pattern = data["pattern"]

matches = re.findall(pattern, text)

If not, that’s probably why pattern seems to "not work" — it's either **undefined** or you're not calling it from the right place.

**🧠 Ask Yourself:**

* Where is your regex pattern stored?
* Are you extracting it from the right dictionary and key?
* Is text defined and loaded with content from the article?

Want to share the surrounding lines of this code so I can help you debug it better?

4o

**You said:**

#imports regular expression to find text patterns import re # imports to enable interaction with file system import os #imports to hadle tabular data and export tsv file import pandas as pd # function writes a list of data rows intotsv file using panadas def write\_tsv(rows, column\_list, path): #list of rows is converted into panads DataFrame df = pd.DataFrame(rows, columns=column\_list) #DataFrame is written into tsv df.to\_csv(path, sep="\t", index=False) #Setup paths #defining folder where articles are present repo\_path = "FASDH25-portfolio2" folder = r"C:\Users\batoo\Downloads\FASDH25-portfolio2\articles" # used chatgpt to add "r" before the path so it works #define path and load gazetteer from the tsv file having place names and alternate names gazetteer\_path = r"C:\Users\batoo\Downloads\FASDH25-portfolio2\gazetteers\geonames\_gaza\_selection.tsv" #Read Gazetteer File with open(gazetteer\_path, encoding="utf-8") as file: data = file.read() #an empty dictionary is created for each place names and a count for matches patterns = {} #split gazetteer data in new lines to get eeach row on file rows = data.split("\n") #Skip header because the pattern starts from next row for row in rows[1:]: #seprates each column in tsv by tabs columns = row.split("\t") #ensures that first column has names for the place asciiname = columns[0] #lists name variants consisting of the standard name name\_variants = [asciiname] #gets the alternate names from the 6th column which is counted as the 5th column, if present alternate\_names = columns[5].strip() if alternate\_names: #splits the alternate names with comma and gets list of other names alternate\_list = alternate\_names.split(",") #loops through each alternate name in the list for name in alternate\_list: #.strip will remove whitespace from alternate names and .append will add alternate names to the list if prenest name\_variants.append(name.strip()) # build a regex pattern that will work to find all names and match diffrent varianats of the place names aswell #using re.escape to escape any special characters in place names and "|" is used for alternation regex\_pattern = "|".join (re.escape(name) for name in name\_variants) # used chatgpt in writing rregex #includes all names and their variants with numbers patterns[asciiname] = {"pattern": regex\_pattern, "count": 0} #dictionary stores how many times each name was mentioned mentions\_per\_month = {} #set the date to filter of gaza articles from this date war\_start\_date = "2023-10-07" #loop through each file as it counts the number of time patterns are found in the folder for filename in os.listdir(folder): #extract the dates from file name in "YYYY-MM-DD" format date\_str = filename.split("\_")[0] #if file starts before the start war skip that if date\_str < war\_start\_date: continue file\_path = f"{folder}/{filename}" #build file path to current articles with open(file\_path, encoding="utf-8") as file: #open and read the articles text = file.read() #loop through each places and search for matches in text for place in patterns: pattern = patterns[place]["pattern"] #find all matches of the place names matches = re.findall(pattern, text) okay this is the whole code but the last one is giving error

**ChatGPT said:**

Thanks for sharing the full code — you're doing great overall! Now let’s carefully pinpoint the issue with this line:

python

CopyEdit

matches = re.findall(pattern, text)

**✅ The line itself is syntactically correct.**

But there’s **one major logic error** in how you wrote the for loops and indentation — and that’s what’s causing your problem.

**🔍 The Core Problem:**

This part of your code is incorrectly indented:

python

CopyEdit

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

text = file.read()

#loop through each places and search for matches in text

for place in patterns:

pattern = patterns[place]["pattern"]

matches = re.findall(pattern, text)

Your for place in patterns: loop is **not indented inside** the file-reading block. So, when Python reaches text = file.read(), it reads the file, but then text goes out of scope (is no longer available) **by the time the loop tries to use it**.

**✅ What you need to do:**

Make sure that the loop which uses text is **inside the block where text is defined**. Like this:

python

CopyEdit

for filename in os.listdir(folder):

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

if date\_str < war\_start\_date:

continue

file\_path = f"{folder}/{filename}"

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

text = file.read()

# ✅ Loop should be inside here

for place in patterns:

pattern = patterns[place]["pattern"]

matches = re.findall(pattern, text)

# you can count matches etc. here

**💡 Pro Tip:**

Always be careful with indentation in Python — blocks of logic depend completely on it. If your code can't find a variable like text, it's often because the variable is **defined inside a block** but used **outside** it.

Would you like help writing the counting and saving part now that this is fixed?

4o

**You said:**

i saw my friends code and she has added re.IGNORECASE what is that for and should i use that too?

**ChatGPT said:**

Great question! re.IGNORECASE (also written as re.I) is a **flag** in Python’s re module that tells the regular expression engine to **ignore case** when matching.

**🔍 Without re.IGNORECASE**

Your regex is **case-sensitive** — so it will match "Gaza" but **not** "gaza" or "GAZA".

**✅ With re.IGNORECASE**

The regex will match all **capitalization variants**, like:

* Gaza
* gaza
* GAZA
* GaZa

**💡 Should *you* use it?**

Yes — **definitely** in your case. Since place names in articles can appear in **any capitalization**, using re.IGNORECASE ensures **you don’t miss matches**.

**✅ How to add it**

In your re.findall() line, add it like this:

python

CopyEdit

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

This small change can significantly improve your results — especially in real-world texts like news articles.

Would you like to go over how to count and store these matches next?

4o

**You said:**

i think i need to use re.IGNORECASE because my instructions say that i need to ignore case when matching even if there are capital or small letters

**ChatGPT said:**

Exactly — if your instructions require you to **ignore case**, then using re.IGNORECASE is the **right and necessary choice**.

It ensures that all capitalization versions (like rafah, RAFAH, Rafah, etc.) are matched by your regex pattern, **without you needing to list every version** in your gazetteer.

**✅ Update this line in your code:**

python

CopyEdit

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

That’s all you need to do — the rest of your pattern logic can stay the same.

Would you like help with the next step — like **counting matches** and saving them to your TSV file?