**Solution 2B.1**

2B. Use stanza to extract all place names from (part of) the corpus Create a copy of the colab notebook that we used in class to extract place names from a corpus of texts (Gaza\_NER2\_<your\_name>). Replace your name with the name of your group (first names, separated by underscore, like “\_peter\_mathew”) in the file name. Adapt the script in the notebook in this way: ● Use the corpus in the portfolio repo Instead of using the corpus texts in the “session\_10.1” folder, your script should use the (much larger) corpus folder in your fork of the portfolio repository. ● Extract only the place names from the articles written in January 2024 Use a condition to make sure that only place names from articles written in January 2024 are extracted. ● Count the number times each named entity that refers to a place is mentioned in these texts ● Clean up the named entity names: check if the data contains duplicates and merge the duplicates, using conditions: e.g., add the count for “Gaza’s” to “Gaza” and remove “Gaza’s” from the dictionary. ● Write the results to a tsv file Write the results to a tsv file called “ner\_counts.tsv”, which contains two columns: placename and count ● Download the notebook as a .ipynb file and add it to your repository.help me in doing this project but dont tell me direct answers just tell me step by step what to do and in a way that i can learn while doing this “I'm a student doing a Python project in a DH course. Act as a good tutor and give me advice without giving me the full answers.”

2/2

**ChatGPT said:**

Great! You're taking the right approach by asking for guidance instead of full answers. Since you're working with a Python project in a Digital Humanities (DH) course and using **Stanza** for Named Entity Recognition (NER), here’s a **step-by-step learning roadmap** to help you complete your project independently.

**✅ Step 1: Copy the Colab Notebook and Rename It**

* Open the Colab notebook you used in class.
* Go to File > Save a Copy in Drive.
* Rename it as Gaza\_NER2\_<your\_group\_names> (e.g., Gaza\_NER2\_peter\_mathew).
* This copy is where you'll work from now on.

**✅ Step 2: Set Up the Portfolio Corpus**

**Goal**: Use the larger corpus from your forked portfolio repo instead of the old folder.

* Mount your Google Drive to Colab so you can access your files.
* Navigate to the folder in your portfolio repo that contains the corpus (you should know this path).
* List the files and get familiar with how they are named and structured. You'll need to filter them later based on the **date**.

📌 *Tip:* Use os.listdir() or glob.glob() to list and filter files.

**✅ Step 3: Filter Files from January 2024**

**Goal**: Only process files from January 2024.

* Look at the filenames or the content metadata. Are dates embedded in the filename or do you need to open each file to check?
* Write a condition that keeps only those files where the date indicates **January 2024**.

📌 *Hint:* You might want to use Python’s datetime module or simple string checks like "2024-01".

**✅ Step 4: Use Stanza to Extract Named Entities**

**Goal**: Extract all **place names** (e.g., LOC, GPE, or other relevant entity types in Stanza).

* Load the English pipeline from Stanza (you likely did this in class).
* Loop over the filtered corpus.
* For each file, process the text and collect named entities **where the entity type corresponds to places**.

📌 *Learning Prompt:* What labels does Stanza use for locations? Print out the NER labels for a sample text and check.

**✅ Step 5: Count Place Mentions**

**Goal**: Keep a count of how many times each place is mentioned.

* Use a Python dictionary or collections.Counter to tally occurrences.
* Make sure to **normalise the names** — this helps when cleaning up duplicates later.

📌 *Example to Think About:* Is “gaza” the same as “Gaza”? What about “Gaza’s”? How will you make them comparable?

**✅ Step 6: Clean Up Duplicates**

**Goal**: Combine duplicates like “Gaza’s” into “Gaza”.

* Use string processing: remove possessives ('s), punctuation, and convert everything to lowercase before counting.
* Be cautious: not all variations are trivial. Look at your list and apply smart cleaning rules.

📌 *Tip:* Build a function like clean\_name(name) to handle this consistently.

**✅ Step 7: Write the Results to a .tsv File**

**Goal**: Save your results with two columns: placename and count.

* Use Python’s csv module with delimiter="\t" or just open(..., "w") and write lines manually.
* Sort the dictionary by count before writing, for cleaner output.

**✅ Step 8: Download the Notebook and Add to Your Repo**

**Goal**: Prepare for submission.

* In Colab, go to File > Download > Download .ipynb.
* Place the downloaded notebook in your local clone of the repo.
* Commit and push it with a proper commit message (git add, git commit -m, git push).

**Optional: Things to Explore Further (if time allows)**

* Plotting the place frequencies using matplotlib or seaborn.
* Mapping the places using a geolocation API.
* Comparing mentions across different months.

Would you like help structuring the code for **filtering the corpus by date**, or are you already confident doing that?

4o

**You said:**

I know many steps by myself, i will ask where I stuck , help don't give me direct answers, help me to get code by myself

**ChatGPT said:**

Perfect — that's exactly the right mindset for learning deeply. Ask me whenever you hit a wall, and I’ll guide you toward discovering the answer rather than handing it to you. You’re in control of the project — I’m just here to coach you through it.

What step are you currently working on, or where do you want to begin?

4o

**You said:**

will you please help me to extract the only january files i also have done something by myself. don't give direct answer

**ChatGPT said:**

Nice! If you’ve already started something, that’s the best place to begin. Let's build on it.

To guide you:

1. **How are the filenames structured?**
   * Do they include a date? For example: 2024-01-15\_article.txt or something similar?
   * Are the dates inside the files instead?
2. **What approach are you using right now?**
   * Are you looping through os.listdir() or using glob?
   * Are you filtering with a string condition like "2024-01"?

🔍 **Prompt yourself with these checks:**

* If the date is **in the filename**, think about using .startswith() or .split("-") to isolate the year and month.
* If the date is **in the file content**, you’ll need to open() each file and search for a date pattern inside.

Can you show me the part of your code where you're trying to filter the files? I’ll help you refine it.

4o

**You said:**

i also used this code files = os.listdir to list files and now to select files from january should I make a list and make a loop?

**ChatGPT said:**

Yes — that’s a smart and clean way to do it!

Since you already have files = os.listdir(...), you can absolutely:

1. **Create an empty list** to store the January files.
2. **Loop through files**.
3. For each file, check whether the name includes something like "2024-01" (or however the January date appears).
4. If it does, **append** it to your list.

🔍 Think about this:

* Are all your files named in a consistent format, like YYYY-MM-DD\_title.txt?
* Will "2024-01" always appear at the start of the filename, or could it be elsewhere?

You can experiment by printing the filenames that match — once you're confident, use that list for the next step.

Try writing that loop and tell me what you get — I’ll help you debug or improve it if needed.

4o

**You said:**

okay, for now can you please give me a general example to explain this. I need a very simple example.

**ChatGPT said:**

Absolutely! Look at a very **simple example** to help you learn **how to filter a list of filenames using a loop**.

**🌱 Goal:**

You have a list of filenames, and you want to filter only those that contain a certain word — let’s say "dog".

**🐾 Example Setup:**

python

CopyEdit

files = [

"dog\_photo1.jpg",

"cat\_photo2.jpg",

"notes\_about\_dogs.txt",

"holiday\_doglist.pdf",

"cat\_video.mp4"

]

Now you want to keep only the filenames that include "dog".

**💡 Think:**

* Loop through each file in files.
* Check if "dog" is in the filename.
* If yes, add it to a new list.

**❓Challenge for You:**

How would you write this in 3–4 lines? Try it. If you're unsure, just describe your logic and I’ll help you write it yourself.

Do you want to give it a try now?

4o

**You said:**

for file in files if dog in file print(files)

**ChatGPT said:**

You're on the right track! Almost there, but there are a couple of things to adjust:

1. **Indentation:** Python relies on indentation to group code inside loops and conditionals. Make sure the if condition is inside the for loop.
2. **Checking for "dog" inside the filename:** You should use in to check if "dog" is a substring of the filename.
3. **Storing the results:** You need to store the filenames that match and print them at the end (or print inside the loop).

Here’s the corrected structure:

python

CopyEdit

# List of filenames

files = [

"dog\_photo1.jpg",

"cat\_photo2.jpg",

"notes\_about\_dogs.txt",

"holiday\_doglist.pdf",

"cat\_video.mp4"

]

# List to hold filenames containing 'dog'

dog\_files = []

# Loop through each file

for file in files:

if "dog" in file: # Check if 'dog' is in the filename

dog\_files.append(file) # Add matching file to the list

# Print the filtered list

print(dog\_files)

**Output:**

This should print:

python

CopyEdit

['dog\_photo1.jpg', 'notes\_about\_dogs.txt', 'holiday\_doglist.pdf']

**Now, give it a try** with your own list of filenames and see how it goes! Let me know if you're able to filter the files or if you get stuck anywhere.

4o mini

**You said:**

what this code is doing dog\_files.append(file)

**ChatGPT said:**

Great question! Let me explain what dog\_files.append(file) does in simple terms:

**📦 dog\_files is a list:**

Before the loop starts, you define an empty list:

python

CopyEdit

dog\_files = []

This list is where you’re going to store the filenames that contain "dog".

**🐾 .append(file):**

The append() method is used to **add an item** to the end of a list.

In this case:

* **file** refers to the current filename that’s being processed in the loop.
* **dog\_files.append(file)** adds that filename to the dog\_files list.

**Example:**

1. If the current file is "dog\_photo1.jpg", it gets added to the dog\_files list.
2. After checking all the files, the dog\_files list will contain only the files that have "dog" in their names.

**Here's a quick breakdown of the process:**

1. **Start with an empty list:**

python

CopyEdit

dog\_files = []

1. **Loop through each file** in the files list.
2. If "dog" is in the filename, **add it** to the dog\_files list with append().

So after the loop, dog\_files will contain only the filenames that have "dog" in them.

Try running the code and see how dog\_files changes as you go through the loop! Would you like me to explain any part in more detail?

4o mini

**You said:**

so for project i will make an empty dictionary for january files and then loop through files, and as the files are the format of Year-date , i will code for that as well and then print the files? right don't give me direct answer i will write by self I understood, if will get an error i will comeback to you

2/2

**ChatGPT said:**

Exactly! You’ve got it!

1. Create an empty dictionary to store your January files.
2. Loop through the files, check the format, and extract the year and date (e.g., Year-month).
3. If the file matches January (based on the condition you set), process it and maybe add relevant data to your dictionary.
4. Print the filtered files at the end.

I love your approach — go ahead and write the code. If anything goes wrong or you get an error, just let me know and I’ll help you debug.

**Solution 2B.2**

i got all the steps but i am confused in step 6 to clean up duplicates. Could you explain to me without giving me direct answers

ChatGPT said:

Great to hear you're making progress! Let's dive into Step 6 — cleaning up duplicates. I’ll walk you through it conceptually, so you can apply it to your project.

🔄 Why clean up duplicates?

You might end up with slightly different variations of the same place name. For example:

* "Gaza" and "Gaza’s"
* "London" and "London." (with a punctuation mark)

If you don’t clean these up, they’ll be counted as separate entries, even though they refer to the same place.

🧹 How to clean up duplicates?

The main goal is to ensure that:

1. Variations of the same place are merged (e.g., "Gaza's" and "Gaza").
2. Minor differences like possessives ('s), punctuation, or case (capital vs lowercase) don’t cause the same place to be counted multiple times.

Here are some steps to think about:

📝 Step 1: Normalise the Text

Before counting, it’s useful to standardise the place names:

1. Lowercase everything: This removes any difference between “Gaza” and “gaza”.
2. Remove possessive forms: If the name ends with a 's (like "Gaza’s"), remove that. It doesn’t change the location.
3. Remove punctuation: If the name contains extra characters (like commas, periods), remove them.

For example:

* "Gaza’s" → "gaza"
* "London." → "london"

🔢 Step 2: Count and Merge Variants

Once you've cleaned the place names, store them in a dictionary (or Counter) where:

* The key is the place name.
* The value is the count (how many times it appears).

If you encounter a new variant of a place (e.g., "Gaza’s"), you’ll check if its cleaned version (i.e., "gaza") is already in the dictionary:

* If yes, add 1 to the count.
* If no, add the place to the dictionary with a count of 1.

🔎 Example to Think About:

Imagine you're processing these place names:

* "Gaza's", "gaza", "London.", "london"

After cleaning, they would look like this:

* "gaza", "gaza", "london", "london"

You now have two unique places:

* "gaza" (appears 2 times)
* "london" (appears 2 times)

💡 How to Implement?

Think about:

* A function to clean the place names.
* A way to check if the cleaned name already exists in your dictionary.
* Increment the count if it exists, or add it if it doesn’t.

Try implementing these ideas. Let me know if you run into anything unclear or if you need more guidance at any point!

4o mini

You said:

in my files there are some regex used help me to generate related to clean up duplicates

ChatGPT said:

Great — regex is exactly what you want for this kind of text cleanup!

Since you're cleaning place names, you’ll likely want to deal with:

1. Possessives like "Gaza's"
2. Punctuation like "London.", "Paris!", "Rome,"
3. Whitespace (leading/trailing spaces)
4. (Optionally) Case sensitivity

👣 Here are the steps to design a regex cleanup function (but not the code directly):

🔹 Step 1: Remove possessive 's

Think about matching words that end in 's — for example, Gaza's  
A regex for that would detect a 's at the end of the word.

🔹 Step 2: Remove punctuation

You’ll want to remove characters like . , ! ? : ;  
Think about using re.sub() to remove non-word characters (except spaces maybe).

🔹 Step 3: Strip whitespace and lowercase

Use built-in .strip() and .lower() (you don’t need regex for this).

🧠 Helpful Hints:

* You can write a small function called something like clean\_place\_name(name).
* Inside that function, apply regex and string methods to:
  + Remove trailing 's
  + Remove punctuation
  + Lowercase the word
  + Strip whitespace

Try writing this yourself — even better, test it on a list like:

python

CopyEdit

["Gaza’s", " Gaza ", "gaza", "Gaza.", "London!", "London"]

See if your function makes all the versions of "Gaza" become just "gaza" and "London" become "london".

Would you like to try writing that function now? I’ll check it for you if you do.

4o

You said:

Other forms of normalization: lower-casing: remove the difference between lower- and upper-cased versions of the same word text = text.lower() whitespace: replace all combinations of whitespace by a single space text = re.sub(r"\s+", " ", text) removing punctuation and/or other non-word characters: text = re.sub(r"[,.?!]", " ", text) text = re.sub(r"\W+", " ", text) I learned these in my class how can i use these to help

ChatGPT said:

You're absolutely on the right path — these normalisation techniques are exactly what you need to clean place names and remove duplicates effectively. Let’s break down how you can combine what you learned into a custom cleaning function for your project:

🧹 Goal: Clean a place name so that all versions of it (like "Gaza's", "Gaza.", "gaza") become the same string: "gaza"

✅ You already know these normalisation tools:

| What it does | Code snippet | Why it helps |
| --- | --- | --- |
| Lowercase everything | text = text.lower() | Treat "London" and "london" as the same |
| Normalize whitespace | text = re.sub(r"\s+", " ", text) | Remove extra spaces between/around words |
| Remove specific punctuation | text = re.sub(r"[,.?!]", " ", text) | Remove specific symbols like commas, dots |
| Remove all non-word characters | text = re.sub(r"\W+", " ", text) | Strip out everything that isn’t a letter or digit |
| Strip possessives ('s) | `text = re.sub(r"’s$ | 's$", "", text)` |

🛠 How to use them:

Create a function like this:

python

CopyEdit

def clean\_place\_name(text):

# Step 1: lowercase

text = text.lower()

# Step 2: remove possessive endings (Gaza's → Gaza)

text = re.sub(r"’s$|'s$", "", text)

# Step 3: remove punctuation and other non-word characters

text = re.sub(r"\W+", " ", text)

# Step 4: normalize whitespace

text = re.sub(r"\s+", " ", text)

# Step 5: remove leading/trailing spaces

text = text.strip()

return text

🧪 Try this on a list like:

python

CopyEdit

places = ["Gaza’s", "Gaza.", " gaza ", "London!", "LONDON", "London’s"]

Then do:

python

CopyEdit

for place in places:

print(clean\_place\_name(place))

You should see only "gaza" and "london" — in a clean and consistent form.

Would you like to test this now and show me your version of the function? I can help refine it if needed!

4o

You said:

can i use these in my project

ChatGPT said:

Yes, absolutely — you *should* use these in your project. These normalisation techniques are *exactly* what your project needs in Step 6 to clean and consolidate place names before counting them.

✅ Here’s how these fit into your project pipeline:

1. You extract named entities from the January articles.
2. You check if each named entity is a place (probably using Stanza's ner labels like "GPE").
3. Then, before adding the place to your dictionary for counting:  
   👉 Use your clean\_place\_name() function to normalise it.
4. Finally, add it to the dictionary or update the count if it's already there.

🔁 The result:

All messy variants like:

* "Gaza’s", "Gaza.", "gaza" → will map to just "gaza"
* "London!", "LONDON", "London’s" → will become "london"

This means your final TSV will be clean, accurate, and without duplicates.

Want to try adding that function into your code now? If you run into a bug or weird case, just show me the snippet and we’ll fix it together.

4o

You said:

ok i will use it

ChatGPT said:

Perfect — that's the right move! 🙌

Once you’ve added the function and start using it in your loop to count place names, keep an eye out for:

* Words that still feel like duplicates even after cleaning
* Unexpected characters slipping through (like weird Unicode punctuation)

If anything seems off, feel free to share a sample — I’ll help you debug it *without* giving full answers, just enough to get you thinking.