# Sara Baig

**You said:**  
I want to extract place names using Stanza, but only from the January 2024 articles in our project repo. Should I write a condition inside the loop to filter dates?

**ChatGPT said:**  
Yes, you're on the right track!  
When looping through the files in your /articles folder, you can check if the filename includes "2024-01" using an if condition like:

python

CopyEdit

if "2024-01" in filename:

This ensures you only process files from January 2024, based on the YYYY-MM-DD format in the filenames.

**You said:**  
I’m confused about how to count the number of times a place is mentioned. Is using a dictionary the best way?

**ChatGPT said:**  
Yes, using a dictionary is a good approach!  
You can store place names as keys and increment their counts as values. For example:

python

CopyEdit

places = {}

places[place] = places.get(place, 0) + 1

This will automatically create a new key if the place isn’t already in the dictionary.

**You said:**  
Why are we using GPE and LOC only in Stanza for identifying place names?

**ChatGPT said:**  
Great question!  
In Stanza’s Named Entity Recognition (NER) system:

* GPE stands for **Geo-Political Entity** (e.g., countries, cities).
* LOC stands for **Location** (e.g., regions, rivers).

These two categories together help capture most real-world place references while avoiding unrelated entities like organizations or people.

**You said:**  
Where exactly do I insert doc = nlp(text) — before or after checking entities?

**ChatGPT said:**  
You should call doc = nlp(text) **right after reading the article text**, but **before** the entity loop.  
This applies the Stanza pipeline to your text, enabling you to loop over doc.entities right after that.

**You said:**  
I’ve printed the raw output of places dictionary and it has things like "Gaza's" and "The United States". How do I clean these?

**ChatGPT said:**  
That's an important step!  
To normalize place names, you can:

* Remove 's using regex like r"[’']s\b"`
* Strip leading "the " using re.sub(r"^the\s+", "", place, flags=re.IGNORECASE)  
  This way "Gaza's" becomes "Gaza" and "The United States" becomes "United States" — avoiding duplicate entries.

**You said:**  
If I clean the names after counting, how do I merge duplicate entries like “Gaza” and “Gaza’s”?

**ChatGPT said:**  
To fix this, create a **new dictionary** called normalized\_places.  
Loop through your original places dict, clean each key, and use the cleaned name as the key in the new dictionary:

python

CopyEdit

normalized\_places[cleaned\_name] = normalized\_places.get(cleaned\_name, 0) + count

This merges entries like “Gaza’s” and “Gaza”.

**You said:**  
What’s the purpose of using re.sub(r"[^\w\s]", "", place)?

**ChatGPT said:**  
That regular expression removes **punctuation characters** from place names.  
For example, "Gaza's" becomes "Gazas", and "New-York" becomes "NewYork".  
This helps standardize names but use it carefully — it can also remove useful hyphens or apostrophes.

**You said:**  
Do I need to tokenize or lemmatize the text manually before NER?

**ChatGPT said:**  
No need! When you run nlp = stanza.Pipeline(...), and call doc = nlp(text), the pipeline **automatically** handles:

* Tokenization
* Multi-word token expansion
* Named Entity Recognition  
  So you don’t need to process tokens manually.

**You said:**  
How should I save my final counts to a .tsv file?

**ChatGPT said:**  
You can open a file in write mode and write the header followed by each row from your places or normalized\_places dictionary. Here's the basic structure:

python

CopyEdit

with open("ner\_counts.tsv", "w", encoding="utf-8") as file:

file.write("Place\tCount\n")

for place, count in normalized\_places.items():

file.write(f"{place}\t{count}\n")

**You said:**  
how to join two .tsv files like ner\_counts.tsv and NER\_gazetteer.tsv. I want to plot frequencies, but I need to merge them first. What function should I look into?

**ChatGPT said:**  
That’s a great question!  
What you need is a **merge operation** — which is like joining two tables in Excel using a common column (in your case, probably “Place” and “Name”). The pandas.merge() function helps with this. It’s often used to combine datasets that share a key.

**You said:**  
I noticed some of the coordinates are empty or NA. Is it okay to drop them before plotting?

**ChatGPT said:**  
Yes, that’s expected — some locations might not have been geocoded properly. You can **safely drop rows** with missing data using:

python

CopyEdit

df.dropna(subset=["Latitude", "Longitude", "Count"])

This ensures your map doesn’t crash when trying to plot empty values.

**You said:**  
Why did I need to convert “Latitude”, “Longitude”, and “Count” to float?

**ChatGPT said:**  
Great observation — this step becomes important when using **Plotly**, especially for sizing markers.  
Even if the numbers look numeric, they’re read as strings from .tsv.  
If not converted using .astype(float), Plotly might either throw errors or misinterpret them.