**ChatGPT Conversation 1**

I am a digital humanities student and i am working on a project where i have to work on a DataFrame of the results from topic modeling the dataframe consists of dates, titles, file name, Topics, count, topic 1, topic 2, topic 3 and topic 4 in separate columns. Now my task is to explore and visualize this data in however way i what. what i want to do is find the most used words per topic in the dataset. I have already created another column called topic label which includes the human readable names from each of the topic columns. Could you help me with finding the words per topic with only pandas and plotly

**ChatGPT said:**

Absolutely! Since you're working with a DataFrame from topic modeling and you want to find the **most used words per topic**, we can do this using only **pandas** and **plotly** — and we’ll use the **melt()** function to make this easier.

**🎯 Goal:**

You want to:

* Find the **most frequently used words** (like “climate”, “economy”, etc.) for each **topic label** (like “Climate”, “Politics”, etc.).
* Create a **visualization** to show the top words per topic.

**🧾 What your DataFrame looks like:**

Let’s say your DataFrame looks like this:

| **date** | **title** | **file\_name** | **Topics** | **count** | **topic\_1** | **topic\_2** | **topic\_3** | **topic\_4** | **topic\_label** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2024-01-01 | ... | ... | ... | ... | climate, global | warming, heat | change | emissions | Climate |
| 2024-01-02 | ... | ... | ... | ... | policy, action | climate, change | future | emissions | Climate |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

Each topic\_1, topic\_2, etc. contains **comma-separated words** (i.e., keywords from that topic), and topic\_label tells you the **human-readable topic name**.

**✅ Solution using melt()**

python

Copy code

import pandas as pd

import plotly.express as px

# 🧱 Step 1: Melt the topic columns into one column

# This turns topic\_1 to topic\_4 into rows, not separate columns

keyword\_df = df[["topic\_label", "topic\_1", "topic\_2", "topic\_3", "topic\_4"]].melt(

id\_vars="topic\_label", # Keep the topic\_label as it is

value\_name="Keyword" # Put all keywords into one column called "Keyword"

)

# 🧱 Step 2: Remove rows where the keyword is missing (NaN)

keyword\_df = keyword\_df.dropna(subset=["Keyword"])

# 🧱 Step 3: Split keywords by comma and flatten them into individual rows

# Example: "climate, global" becomes "climate" and "global"

# We'll collect all the cleaned rows in a new list

expanded\_rows = []

# Go through each row

for i, row in keyword\_df.iterrows():

topic = row["topic\_label"] # Get the topic label (e.g., Climate)

keyword\_string = row["Keyword"] # Get the string of keywords (e.g., "climate, global")

# Split the keyword string by comma and remove spaces

words = [w.strip() for w in keyword\_string.split(",") if w.strip() != ""]

# For each word, create a new row with topic and word

for word in words:

expanded\_rows.append({"Topic": topic, "Word": word})

# 🧱 Step 4: Convert the list of rows into a DataFrame

word\_df = pd.DataFrame(expanded\_rows)

# 🧱 Step 5: Count how many times each word appears per topic

# This gives us word frequency for each topic

word\_counts = word\_df.groupby(["Topic", "Word"]).size().reset\_index(name="Count")

# 🧱 Step 6: For each topic, take the top 10 most common words

top\_words\_df = (

word\_counts

.sort\_values(["Topic", "Count"], ascending=[True, False])

.groupby("Topic")

.head(10) # Get top 10 for each topic

)

# 🧱 Step 7: Create a bar chart with Plotly

fig = px.bar(

top\_words\_df,

x="Word",

y="Count",

color="Topic",

facet\_col="Topic", # One chart per topic

title="Top 10 Words Per Topic",

height=600

)

fig.show()

**📊 What You’ll See:**

You’ll get **bar charts** for each topic (like “Climate”, “Health”, “Politics”), showing the **top 10 words** and how often they appear.

**🧠 Key Concepts You Just Used:**

* **melt()**: Turned multiple topic columns into one.
* **.split(",")**: Broke keyword strings into individual words.
* **.groupby() + .size()**: Counted how often each word appeared.
* **plotly.express.bar()**: Made an interactive chart.

Would you like to change the number of words (e.g., top 5 instead of 10), show only one topic at a time, or sort differently? I can help with all of that too!

1/2

**You said:**

could you explain the melt function being used and also the id\_vars="Topic\_Label" code that you have provided

**ChatGPT said:**

Absolutely! Let’s break down what the melt() function does — and specifically what this part of the code means:

python

Copy code

keyword\_df = df[["topic\_label", "topic\_1", "topic\_2", "topic\_3", "topic\_4"]].melt(

id\_vars="topic\_label",

value\_name="Keyword"

)

**🧠 What is melt() in pandas?**

Think of the melt() function as a way to **reshape your data**.

If you have **multiple columns** that hold similar types of values (like topic\_1, topic\_2, etc.), melt() can help **combine them into one single column** — while **keeping certain other columns fixed**.

**🎯 What we’re doing here**

Let’s say your original data looks like this:

| **topic\_label** | **topic\_1** | **topic\_2** | **topic\_3** | **topic\_4** |
| --- | --- | --- | --- | --- |
| Climate | climate, global | warming, heat | change | emissions |
| Politics | government | policy, vote | leader | election |

This is **wide format**: the keywords are spread across columns.

But we want something like this instead (a **long format**) where each row is one word set per topic:

| **topic\_label** | **Keyword** |
| --- | --- |
| Climate | climate, global |
| Climate | warming, heat |
| Climate | change |
| Climate | emissions |
| Politics | government |
| Politics | policy, vote |
| Politics | leader |
| Politics | election |

That’s what melt() gives us.

**🧱 Now, what does each argument mean?**

**✅ id\_vars="topic\_label"**

This tells pandas:

“**Keep this column unchanged** while you’re melting the others.”

So topic\_label will stay fixed in each row — it won’t get reshaped.

**✅ The rest of the columns (topic\_1, topic\_2, topic\_3, topic\_4)**

Since we didn’t include them in id\_vars, pandas knows we want to **combine them into two new columns**:

* One for their **original column name** (called variable by default — you can rename it).
* One for their **values** (we called this Keyword using value\_name="Keyword").

**✅ value\_name="Keyword"**

This tells pandas:

“Name the new column that holds all the keyword values as Keyword.”

**🔧 Summary of this line:**

python

Copy code

keyword\_df = df[["topic\_label", "topic\_1", "topic\_2", "topic\_3", "topic\_4"]].melt(

id\_vars="topic\_label",

value\_name="Keyword"

)

You're telling pandas:

“I want to keep topic\_label the same, but **combine** the topic\_1 to topic\_4 columns into one column called Keyword.”

Let me know if you'd like a small fake example to run and see what melt() outputs — I can provide that too if you'd like to experiment!

**You said:**

can we also combine the topic 1 to 4 by grouping them

**ChatGPT said:**

Great question! Yes — you **can combine the topic\_1 to topic\_4 columns using groupby()**, but that approach is quite different from using melt(). Let’s walk through how that would work, and how it compares.

**✅ What You’re Asking:**

Can we **group all the topic keyword columns together** — topic\_1, topic\_2, topic\_3, and topic\_4 — **without using melt()**, but instead using **groupby() or other pandas operations**?

Short answer: **Yes**, we can **combine** them (merge their values) into one column **before** grouping — but groupby() comes in **after** we’ve gathered the keywords into a single place.

**🧱 Example Method (Without melt())**

Let’s say you want to collect all keywords from topic\_1 to topic\_4 into **one column**, and then **count them per topic label**.

**Step-by-step:**

python

Copy code

import pandas as pd

# Sample DataFrame

df = pd.DataFrame({

"topic\_label": ["Climate", "Climate", "Politics"],

"topic\_1": ["climate, global", "policy", "government"],

"topic\_2": ["warming, heat", "climate, change", "vote"],

"topic\_3": ["change", "future", "leader"],

"topic\_4": ["emissions", "emissions", "election"]

})

# Step 1: Combine topic\_1 to topic\_4 into one string per row

df["all\_keywords"] = df[["topic\_1", "topic\_2", "topic\_3", "topic\_4"]].agg(", ".join, axis=1)

# Step 2: Create a new DataFrame with one row per word

rows = []

for i, row in df.iterrows():

topic = row["topic\_label"]

keywords = [w.strip() for w in row["all\_keywords"].split(",") if w.strip() != ""]

for word in keywords:

rows.append({"Topic": topic, "Word": word})

keywords\_df = pd.DataFrame(rows)

# Step 3: Group and count

word\_counts = keywords\_df.groupby(["Topic", "Word"]).size().reset\_index(name="Count")

# Step 4: (Optional) Get top 10 words per topic

top\_words = (

word\_counts.sort\_values(["Topic", "Count"], ascending=[True, False])

.groupby("Topic")

.head(10)

)

# Output

print(top\_words)

**🧠 What’s Happening Here:**

* We **combine** all 4 topic columns into one column (all\_keywords) using agg(", ".join, axis=1).
* Then we **split that string** into individual words per row.
* Then we use **groupby()** to count how many times each word appears **per topic**.

**💡 So Which Is Better?**

| **Method** | **Pros** | **Cons** |
| --- | --- | --- |
| melt() | Clean, structured, pandas-native way | Slightly more advanced to learn |
| Manual Grouping | More control over string manipulation | More steps and manual code needed |

Both give you the **same final result**: a table of words per topic with frequency counts.

Would you like me to turn this into a Plotly bar chart too (like before)?

**You said:**

which one would be better

**ChatGPT said:**

Great question — here’s a simple answer tailored to your level and goals as a **digital humanities student** using **only pandas and plotly**:

**✅ Use the melt() method if:**

* Your **topic keywords are already clean and separate in columns** (like topic\_1, topic\_2, topic\_3, topic\_4).
* You want a **simple and beginner-friendly way** to organize your data for visualizing word frequency per topic.
* You don’t need to combine all topic columns into one big string.

**Why it’s better:**  
It’s **shorter**, uses only one line to reshape your data, and is easy to group and count after.

**✅ Use the "group and split manually" method if:**

* Your keywords inside topic\_1 to topic\_4 contain **multiple words separated by commas** (e.g., "climate, change").
* You want to **analyze each individual word**, not just each topic column as a whole.
* You’re comfortable using .split(), .strip(), and a bit more code.

**Why it’s better in that case:**  
It gives you **deeper control** — like breaking apart "climate, change" into two words.

**👇 Here's a Quick Comparison:**

| **Situation** | **Best Method** |
| --- | --- |
| You want quick topic-wise keywords for charts | ✅ Use melt() |
| You want **individual word frequency** from multi-word strings | ✅ Use grouping & splitting |

**📌 Final Recommendation for You:**

**Start with melt()** — it’s cleaner, more readable, and good for structured topic modeling output.  
If you later notice that your keywords have **multiple words joined with commas**, then **switch to the splitting method**.

Let me know and I can adapt the code based on **what your actual keyword cells contain** — single words or comma-separated phrases!

**Name for background colors**

<https://stackoverflow.com/questions/22408237/named-colors-in-matplotlib>