**AI SCRIPT EXPLORATION:**

**You said:**

I'm working on topic modeling and I’ve extracted keywords from four different columns topic\_1, topic\_2, topic\_3, and topic\_4. What I want now is a way to bring together all the keywords from these columns into a single list. But I don’t want duplicates just a clean list of all unique keywords across these four columns. What’s the best approach to do that using pandas?

**ChatGPT:**

You’re aiming to flatten multiple columns into one unified list of unique items, which is a common task in preprocessing. A good method here would be to combine the values from each of the keyword columns and then filter out any repeated entries. In pandas, this can be done by concatenating the individual series and applying a method to keep only distinct values. Finally, converting that result to a list will give you what you need for further use.

**You said:**

now that I have this list of keywords, I’ve noticed that some of them are ambiguous. For instance, words like *bank*, *court*, *border*, or *hostages* can have multiple meanings depending on the context. I already have a separate list of ambiguous words I’m concerned about. Is there a way I can check which of these ambiguous words actually show up in my keywords list?

**ChatGPT:**

Yes, this is a great step for refining your analysis. What you’re trying to do is intersect two lists: the list of ambiguous words you already have, and the keywords you just extracted. This comparison will help you identify which terms might need closer inspection or special treatment during interpretation. You can perform this comparison using a simple filtering technique — typically a list comprehension or set-based approach works well.

**You said:**

Another thing I’m working on is assigning **thematic labels** to somE of the topic groups. Right now, each topic is represented by a string of keywords Like *“captives, hamas, release, hostages”* but I want to give them more meaningful names for interpretation, like calling that one *“Hostage Crisis & Hamas Conflict”*. CAn I do that manually by matching certain keyword strings to custom labels?

**ChatGPT:**

That’s an excellent idea for improving readability and understanding of your results. What you can do is create a dictionary where the key is the exact keyword string from your topic label column, and the value is your descriptive theme label. Then you apply that mapping to create a new column — let’s say topic\_model — which contains the human-readable themes. This doesn't just help in documentation but is also useful when creating plots or summaries.

**You said:**

Once i assigned these custom theme names and created the topic\_model column, I want to analyze how these themes appear over time. Specifically, I want to calculate how many articles are associated with each theme in each month. How can I summarize the data this way by counting how many articles belong to each theme per month?

**ChatGPT:**

You’re looking to do a **grouped count over time**, which is a classic case for using a multi-level groupby in pandas. First, you’d need to ensure that your date column is processed to reflect just the month and year — often using a to\_period("M") type transformation. Then, you can group by both that new time column and your theme label column to count the number of rows (i.e., articles) in each category. This result will give you a table of monthly article counts per theme.

**You said:**

In my written interpretation of the results, I’ve included a few natural language observations, like *“Most articles in 2023–2024 focused on hostages and healthcare themes.”* These aren’t part of the code are just comments to help understand the output. Are these types of summary statements okay to include?

**ChatGPT:**

Absolutely — such interpretive statements are not only valid but encouraged. They reflect your ability to understand and explain what the data is telling you. Including them shows your engagement with the results, especially if this is part of a course or project report. They don’t interfere with the logic of the code — they complement it by turning raw numbers into narrative insights. Just make sure to ground them in actual observations from your grouped results.

Would you like this version exported into a Word document or formatted for your project submission?

**AI VISUAL 1:**

**User:**  
Hi, I’m working on monthly trends using topic modeling data, and I want all the articles from the same month to show up together. Like if one article is from May 12, another from May 20, and another from May 31 I want them all to be counted as “May” in my chart. How can I do that?

**ChatGPT:**  
That’s a very useful goal, especially for time series analysis. What you’re trying to do is **normalize your date values to the month level**, so all dates from May 2023 get treated as just “May 2023” instead of showing up separately.

In pandas, one way to do this is to **extract the month** from a full date column. But if you want to still keep it as a datetime-like object (for plotting or grouping), you can transform it into a **monthly timestamp**, where all dates are set to the first of the month.

**User:**  
what do you mean by “monthly timestamp”? Like do I need to change every date to the 1st of the month?

**ChatGPT:**  
Yes! Not because you care about the *actual day* being the first — but because that allows all May articles (whether May 2 or May 30) to share the same value like “2023-05-01”. This makes grouping and plotting them by month much easier.

There’s a way to do this using **pandas datetime formatting** — you basically *floor* your dates to month-level using a special method that converts full dates into **monthly periods**, and then turns those into timestamps.

**User:**  
So it’s not really about changing the real data, it’s about formatting it in a way that helps me group by month, right?

**ChatGPT:**  
Exactly. You’re just standardizing it for the purpose of monthly aggregation. The real article date (like May 27) is still available if you need it, but this new “month” column becomes your x-axis.

**User:**  
Okay I think I can try that. Let me make a new column for it and see how it affects the plot.

**User:**  
Now I want to make the bar chart show the top 5 most frequent topics over time, and I want each bar to show not just the topic number but also the main keywords like maybe the top 4 words that describe that topic. Is that possible?

**ChatGPT:**  
Yes, that’s a great idea. It makes your chart much easier to understand than just using topic numbers.

What you’ll need to do is **combine multiple columns** — like the topic\_1, topic\_2, topic\_3, and topic\_4 columns — into a single string that forms a **custom label**. Then you can use that as the legend or color label in your bar chart.

**User:**  
so instead of saying “Topic 3” or “Topic 25”, I can make it say something like “3: peace, ceasefire, agreement, israel”?

**ChatGPT:**  
Exactly! It gives context directly in the plot, especially useful when your audience doesn’t know what each topic number means. You just have to format the string using the topic number and those keyword columns. That new column will serve as your “Topic Label”.

**User:**  
I’ll try to make that column. Hopefully I can combine all those pieces.

**User:**  
Also… when I plotted the bar chart, I saw that sometimes the bars for different topics are stacked on top of each other. I want them to show side by side in each month. Like grouped bars for comparison. How do I do that?

**ChatGPT:**  
Ah, what you’re referring to is the **bar mode** in Plotly. By default, it may stack the bars when multiple categories share the same x-axis value (like month). But there’s a way to tell Plotly:  
“Instead of stacking these, **line them up next to each other** so I can compare them more easily.”

This mode is called "group" — and it tells Plotly to group bars by their category (like topic) **within** the same month.

**User:**  
That makes sense. I did see grouped bar charts somewhere. That’s probably why their bars were easier to compare.

**ChatGPT:**  
Exactly — it improves readability. You can also adjust things like the angle of the x-axis labels or the font sizes for your titles and axes. Those tweaks help when your data is dense or your labels are long.

**User:**  
We didn’t really do much of that in class, but I get that it’s just for improving how the plot looks, not changing the analysis.

**ChatGPT:**  
Correct. Formatting with layout updates doesn’t change the data or chart type — it just makes your plot more professional and easier to read.

**User:**  
Thanks! I’ll try adding just a few of these improvements and note them in my AI documentation as extra features I explored.

**ChatGPT:**  
That’s the best way to do it! You’re staying within the rules but also showing initiative and deepening your skills.

Would you like this saved as a Word file for submission? I can generate it for you!

**AI FOR VISUALIZATION 2**

**You said:**

I'm working on topic modeling results and want to create readable labels by combining the four keyword columns topic\_1 to topic\_4. I’m not sure how to do that. I know how to work with individual columns, but how do I merge values from several columns into one string?

**ChatGPT:**

That’s a great question! Since you're working with multiple columns and trying to combine their values row by row, the idea is to use a function that goes across the row and glues the values together.

Imagine a single row like this:

"health" | "conflict" | "aid" | "refugees"

You want that to become:

"health, conflict, aid, refugees"

To do that, there’s a way to apply a joining operation across a list of columns. Instead of looping manually, you can tell pandas to apply the join function across each row, using a comma as the separator.

The result is a new column — something like Topic\_Label — which holds a single string for each topic. This makes your data more interpretable and is especially helpful when you want to use it in charts or summaries.

**You said:**

Now I want to find the most discussed topics. I saw a line that was trying to get the top 10 topics but it used something like sum().nlargest(10).index. I'm not sure how that works can you help me think through what’s going on?

**ChatGPT:**

Let’s walk through it together:

* First, you’ve got a grouped result that shows how many articles belong to each topic.
* You want to know: which topics have the highest total article counts?
* To get that, you sum the article counts by topic.
* But instead of sorting everything manually, there’s a method that helps you directly grab the top values — that’s nlargest(10).

So, what’s happening is this: after getting the total count for each topic, the code quickly picks the top 10 with the highest counts. The .index part simply pulls out the names of those topics.

It’s a neat shortcut to rank things and pull out only the most significant ones.