I'm a student working on a Digital Humanities project analyzing the al-Jazeera Gaza corpus. My group is using different datasets. I’m responsible for the topic modelling dataset, while my teammates are working on TF-IDF and n-gram datasets. Please act as a supportive tutor (not a code generator) and help me understand: What topic modelling is, especially in the context of text analysis and distant reading. How topic modelling datasets are usually generated. How I can explore such a dataset in pandas—what should I look at first? Ideas for meaningful visualisations that help interpret topic modelling results in relation to a news corpus like this one. Any common misinterpretations to avoid when using topic modelling for analysis. I’m not looking for full scripts, just concepts, examples, and strategies to start my exploration and build my own code independently. I will document this interaction as per academic honesty guidelines. Let's start by helping me understand topic modelling and what I might expect in the CSV file.

**ChatGPT said:**

That's a great approach, Mahpara! Working with topic modelling in the context of a news corpus like the *al-Jazeera Gaza corpus* offers rich opportunities for **distant reading**—seeing broad patterns and discourses across large text datasets. I’ll walk you through everything you’ve asked, step by step, starting with the **core concept** of topic modelling and what to expect in the dataset.

**🔍 1. What is Topic Modelling? (In the context of text analysis and distant reading)**

**Topic modelling** is an **unsupervised machine learning technique** used to uncover abstract "topics" that occur in a collection of documents. It helps researchers find **patterns of co-occurring words** across many texts.

Think of it this way:

* In close reading, you analyze a few documents in detail.
* In **distant reading**, you're interested in patterns across *many* texts—topic modelling is a tool for that.

**✳️ Example (from your context):**

If you're analyzing news articles on Gaza, topic modelling might discover topics like:

* Humanitarian aid, with top words like *relief, hospital, children, UN*.
* Military operations, with top words like *strike, army, tunnel, air*.
* International response, with *ceasefire, negotiation, US, UN*.

Each "topic" is a **cluster of words** that frequently appear together, and each document is seen as a **mixture of topics**.

**🗂️ 2. How Are Topic Modelling Datasets Usually Generated?**

The process typically includes:

1. **Preprocessing text**: Remove punctuation, lowercase, remove stopwords, lemmatize/stem words.
2. **Vectorization**: Convert text into a format a model can understand (e.g., bag-of-words or TF-IDF vectors).
3. **Apply topic model**: Most commonly **LDA (Latent Dirichlet Allocation)**, but other models like NMF (Non-negative Matrix Factorization) are also used.
4. **Export results**:  
   You might get several kinds of CSV files:
   * topics.csv: List of topics and their top words.
   * document\_topics.csv: Shows how much of each topic is in each document.
   * topic\_keywords.csv: Each row = a topic, with top keywords.
   * topic\_distributions.csv: Each row = a document, each column = % contribution of a topic.

**🐼 3. Exploring Topic Modelling Data in pandas—Where to Start**

Assuming you have a document\_topics.csv or similar, here’s your strategy:

**✅ First things to check:**

* **Column names**: Are they numeric topic IDs like topic\_0, topic\_1?
* **Document labels**: Is there a document\_id, title, or date column?
* **Topic proportions**: Are the values per row (document) summing to ~1? That means you have topic distributions.

**🔍 Look at:**

1. **Top topics by frequency**: Which topics dominate across the corpus?
2. **Topic composition of individual articles**: How many topics are present in each article? Is it mostly one, or a mix?
3. **Topic trends over time**: If you have dates, how do topics rise and fall chronologically?

**📊 4. Ideas for Meaningful Visualisations**

Since this is news content, you want visualisations that help show **narrative patterns** and **shifts in focus**.

Here are some good ones:

**a) Topic Distribution Histogram**

* Show how prevalent each topic is across all documents.
* Useful to identify dominant themes in the corpus.

**b) Top Words per Topic (Bar Charts)**

* For each topic, show its top 10 words.
* Helps interpret what each topic actually represents.

**c) Time Series of Topics**

* Plot topic proportions over time (if articles have publication dates).
* Insight: Did humanitarian coverage rise after a particular event?

**d) Topic Heatmap**

* Rows = documents, Columns = topics.
* Shows which topics are most strongly associated with each article.

**e) Topic Correlation Matrix**

* Shows which topics often co-occur across documents.
* Useful if some topics seem related, like *“ceasefire”* and *“diplomacy.”*

**⚠️ 5. Common Misinterpretations to Avoid**

Here are a few conceptual traps that many students fall into:

**🚫 Topic ≠ Label**

* LDA generates topics mathematically—it doesn't "know" they represent *war* or *peace*. The human analyst interprets them. Don’t over-trust the labels you give.

**🚫 Top words = meaning**

* Just because the top 10 words are clear doesn’t mean the *entire* topic is coherent. Always check example articles with high % of a topic.

**🚫 Topic proportions are not absolute truths**

* A document showing 40% topic A and 30% topic B doesn't mean it literally discusses 40% of A—it’s a probabilistic estimate.

**🚫 Topic overlap is common**

* Documents often belong to multiple topics; don't force them into one "category."

**📘 Summary: What You Might See in Your CSV**

| **document\_id** | **topic\_0** | **topic\_1** | **topic\_2** | **...** |
| --- | --- | --- | --- | --- |
| doc\_001 | 0.12 | 0.66 | 0.22 | ... |
| doc\_002 | 0.03 | 0.01 | 0.96 | ... |

OR

| **topic\_id** | **keyword\_1** | **keyword\_2** | **keyword\_3** | **...** |
| --- | --- | --- | --- | --- |
| 0 | strike | tunnel | air | ... |
| 1 | aid | child | hospital | ... |

You’ll interpret this by:

* Grouping and summarizing topic prevalence.
* Connecting topic presence to time, geography, or article themes.
* Choosing visual tools that help explain your findings.

When you're ready, we can go deeper into how to interpret a specific file or discuss how your findings compare with your teammates' TF-IDF or n-gram patterns. Let me know how you’d like to proceed next!

**You said:**

Now I’ve opened the topic modelling CSV file in pandas and used .columns to inspect the structure. Can you help me figure out: What kind of columns I should expect in a topic modelling dataset? How to interpret values like topic numbers, or keywords in the dataset? What kinds of filtering or grouping I can try to explore patterns over time, or differences between articles? How to connect individual articles back to topics, what strategies might help in identifying dominant themes or shifts over time? I’m not asking for full code just advice, strategy, and interpretation tips so I can build my analysis independently.

**ChatGPT said:**

That’s exactly the right mindset, Mahpara! Now that you’ve opened the CSV and looked at .columns, let’s unpack what you’re seeing and how to move forward.

**🧱 1. What Columns Should You Expect in a Topic Modelling Dataset?**

Depending on the export structure, you might see two broad kinds of datasets:

**🧾 A. Document-Topic Distribution File**

Each row = one document  
Each column = metadata + topic probabilities

**Typical columns:**

* document\_id or filename
* date or published\_date
* title or headline
* topic\_0, topic\_1, ..., topic\_n (each value = probability that topic appears in that document)

These help you explore **which topics are present in each document**.

**🗂️ B. Topic-Keywords File**

Each row = one topic  
Each column = keywords or weights

**Typical columns:**

* topic\_id (e.g., 0, 1, 2, ...)
* top\_words (sometimes a string of comma-separated words)
* OR multiple columns like word\_1, word\_2, ..., word\_10

These help you **interpret the meaning of each topic**.

**🧠 2. How to Interpret Topic Numbers and Keywords**

**🔢 Topic Numbers (topic\_0, topic\_1, ...)**

* These are **arbitrary labels** assigned by the model.
* Topic 3 isn't "more important" than topic 2; the numbers just refer to the order in which topics were generated.

**🗝️ Keywords**

* Keywords are **the words most associated** with a topic.
* They're often ordered by weight (importance), so the first few words give the clearest clue to what the topic is about.
* You’ll **assign a human-readable label** to each topic *after* looking at keywords and sample documents. This is called **topic interpretation**.

**🔍 3. Filtering or Grouping for Deeper Exploration**

If your dataset has **dates** or **categories**, here’s what you can do:

**📅 a. Group by Time Period**

If you have a date column:

* Convert it to datetime format.
* Group by month, week, or day.
* Then calculate the **average contribution of each topic over time**.

➡️ This helps you see **which topics rise or fall**—useful for identifying **news cycles**, **major events**, or **narrative shifts**.

**📰 b. Compare Topics Across Article Types**

If there’s a category, section, or source column:

* Group by this category.
* Compare **average topic weights** across groups.

➡️ Great for seeing how **different outlets or sections** (e.g., opinion vs. news) cover topics differently.

**🔢 c. Filter for High Topic Presence**

You can:

* Filter documents where a topic’s value > 0.5, or whatever threshold makes sense.
* This shows you **articles where the topic is dominant**.
* Reading a few examples will help you understand what the topic *actually* means in context.

**🧭 4. Connecting Individual Articles to Topics & Tracing Themes Over Time**

Here’s a **strategy roadmap**:

**1. Identify Dominant Topic per Document**

* For each article, find the topic with the highest value (using .idxmax() row-wise).
* Add a new column like dominant\_topic.

➡️ Helps group documents by their "main" theme.

**2. Map Topic Numbers to Human Labels**

* After exploring top keywords and dominant articles per topic, make a dictionary like:

Python

**# Conversation 1**

Copy code

topic\_labels = {

0: "Ceasefire & Diplomacy",

1: "Civilian Impact",

...

}

➡️ This makes your analysis more readable and interpretable.

**3. Time-based Patterns**

* Plot the **frequency** or **mean contribution** of topics over time.
* You’ll likely find peaks around real-world events (e.g., major escalations, UN statements).

**4. Cross-compare Topics with TF-IDF or N-grams**

* Ask your teammates to share their top TF-IDF terms or common n-grams per time slice.
* See if certain *topics* match *vocab patterns*.
* This will help in **triangulating your results** and making a stronger argument in your write-up.

**🧠 Summary: Interpretation First, Code Second**

Your guiding question at each step should be:

*What does this distribution of topics reveal about how the Gaza conflict is reported in the corpus?*

Use pandas just as a tool to extract that story. Focus on:

* What topics are most prevalent?
* How they change over time or across categories?
* Which articles “anchor” each topic?
* What does that say about media framing or narrative structure?

**#Conversation 2**

So far in my script, I’ve loaded the topic-model.csv dataset and cleaned it by removing unassigned topics and filtering out meaningless topic keywords (like stopwords). I also created readable topic labels and explored the top 10 most frequent topics. For example, I’ve printed all article titles where topic\_1 is "bank" and even extracted the most common co-occurring topic words (topic\_2, topic\_3, topic\_4). What I want to do now is just randomly select 5 article titles where topic\_1 is "bank" so I can read them and understand the topic better. I'm not sure which function to use for this—something that lets me randomly sample a few rows. Any suggestion?

**ChatGPT said:**

You're on exactly the right track, Mahpara! What you need here is the **.sample()** function in pandas—it’s perfect for selecting a **random subset of rows** from a DataFrame.

**✅ How to Use .sample() for Your Case**

If you already filtered your DataFrame for articles where topic\_1 is "bank", you can apply .sample(n=5) to that subset:

python

Copy code

df[df['topic\_1\_label'] == 'bank'].sample(n=5, random\_state=42)

**📝 Notes:**

* n=5 → selects 5 random rows.
* random\_state=42 → ensures reproducibility (you’ll get the same 5 articles every time unless you change the seed).
* If your topic label is in another column (like 'dominant\_topic\_label'), just adjust the column name accordingly.

**Optional: Show Only Titles**

If you just want the titles for reading and interpretation:

python

Copy code

df[df['topic\_1\_label'] == 'bank'].sample(n=5, random\_state=42)['title']

This will return a neat list of 5 titles for manual reading.

**# Convesation 3**

I want to explore the script more, instead of just raw article counts per topic each month, I want to calculate relative frequency or what percentage of that month’s total articles each topic represents. I’m trying to figure out the best approach to calculate this. I have an idea that I’ll need to calculate the total number of articles per month and then compare each topic’s count to that total. But I’m a little unsure What’s the best way to structure this? Should I calculate totals before or after filtering for the top 5 topics? Is there a good method you recommend for making sure the percentages add up correctly per month? Guide me through any knowledge to help me understand the logic and structure. I’d like to learn the reasoning behind the method, not just the code directly

**ChatGPT said:**

You're thinking like a true data analyst, Mahpara—focusing on **structure, reasoning, and validation**, not just results. Let's walk through the logic of calculating **relative topic frequencies per month**, step by step.

**🎯 Goal Recap:**

You want to see, **for each month**, what **percentage of articles** are mainly about each **top 5 topic**. This helps you **compare topic salience** over time while **controlling for monthly volume differences**.

**🧱 Step-by-Step Reasoning**

**✅ 1. Start With the Full Dataset**

You want to calculate the **total number of articles per month**, so don't filter for specific topics yet.

* Why? If you filter too early, your denominator (monthly totals) will be incomplete or biased.
* You want the **true full count of all articles per month** so your percentages are meaningful.

**🧱 2. Make Sure You Have:**

* A date column (preferably in datetime format).
* A column like dominant\_topic\_label or topic\_1\_label that assigns one topic per article.

If your date is in full format (YYYY-MM-DD), you'll want to extract the month:

python

Copy code

df['month'] = df['date'].dt.to\_period('M')

**📊 3. Calculate Monthly Totals**

This gives you the **denominator** for each month:

python

Copy code

monthly\_totals = df.groupby('month').size()

Each value = total number of articles published in that month.

**🔢 4. Count Articles per Topic per Month**

This gives you the **numerator**:

python

Copy code

monthly\_topic\_counts = df.groupby(['month', 'dominant\_topic\_label']).size()

Now you know how many articles in a given month were dominated by each topic.

**➗ 5. Calculate Relative Frequencies**

Now divide the topic counts by total monthly counts, grouped by month.

Here’s the logic:

* For each (month, topic) pair:
  + Divide topic count by total articles in that month.
* This gives you a **fraction or percentage**.

**🛠 6. Should You Filter to Top 5 Topics Before or After?**

Filter **after** calculating frequencies. Why?

* If you filter before, you won’t get a full picture—you’ll miss months where a topic had **zero articles** (and might mistakenly exclude that month altogether).
* By filtering after, you still retain **zeros** and get consistent time-series coverage.

So:

1. Do the full relative frequency calculation.
2. Then filter your final output to include only your top 5 topics (based on overall frequency or importance).

**✅ 7. Final Structure**

Once you're done, you'll have a DataFrame like:

| **month** | **topic** | **rel\_freq** |
| --- | --- | --- |
| 2023-10 | Ceasefire Talks | 0.28 |
| 2023-10 | Civilian Casualties | 0.22 |
| 2023-10 | Diplomacy | 0.17 |
| 2023-11 | Ceasefire Talks | 0.10 |
| ... | ... | ... |

Each row: one topic’s share of coverage in one month.

You can **pivot** or reshape this for plotting (e.g., time-series line plots for each topic).

**🔁 Cross-Check: Do Percentages Add to 100%?**

If you're assigning **only one dominant topic per article**, then yes:

* The total topic percentages for each month **should sum to 1.0 (or 100%)**.

If you're working with **topic probabilities (soft assignments)** across multiple topics, then no:

* An article may contribute partially to multiple topics, and monthly percentages may exceed 100%.

In your case, since you’re using dominant\_topic\_label, **percentages should sum to 100% per month**, assuming no missing values.

**🧠 Summary of Strategy**

| **Step** | **What You’re Doing** | **Why It Matters** |
| --- | --- | --- |
| 1 | Keep all articles at first | To get accurate totals |
| 2 | Group by month | For time series |
| 3 | Count total articles/month | Your denominator |
| 4 | Count articles per topic/month | Your numerator |
| 5 | Divide counts by totals | Normalize for comparison |
| 6 | Filter to top topics at the end | Avoid skewing totals or hiding months |
| 7 | Visualize or interpret | Trace topic salience over time |