SESSION 3 REFERENCE GUIDE

Session 3: Break It on Purpose

AI + Research Level 2 — Supplementary Material

Concept: DATA CLEANING AND FEATURE ENGINEERING
Space: Silly Phrase Finder with Cleaning
Model: `valhalla/distilbart-mnli-12-3` (same as Session 1)
Pre-built fallback: Have a version with cleaning deployed under profplate/ before class.

Time Breakdown (2 hours)

0:00–0:10 — Show-and-Tell

Ask: "Did anyone try swapping a model into their Space?"

If yes: share it. What model did they find? Did it work? Did it break?

If no: quickly show a model you tried between sessions. Keep it to 2 minutes.

Transition: "Today we're going back to our original Silly Phrase Finder. But instead of changing the model, we're going to break it — on purpose."

0:10–0:35 — Break It

Open the Session 1 Space (original Silly Phrase Finder, no cleaning). Start typing adversarial inputs. Have students suggest inputs too.

Pre-prepared adversarial inputs:

Input	Category	What happens
`□□□ THIS IS THE BEST DAY EVER 😂😂😂 I literally can't even rn`	Emoji + slang	Model may ignore emoji, struggle with "rn"
`THIS IS ABSOLUTELY THE MOST RIDICULOUS THING I HAVE EVER SEEN IN MY ENTIRE LIFE AND I AM NOT HAPPY ABOUT IT`	ALL CAPS	Model may read it differently than mixed case
`Rep. Johnson and Dr. Smith met with Gov. Williams at St. Mary's hospital.`	Abbreviations	Sentence splitter may break on periods in abbreviations
`The food was really good`	Extra whitespace	May not break the model, but it's messy input
`soooooooo booooooored nothing ever happens in this townnnnnnn`	Repeated characters	Model may not recognize stretched words
`no caps no punctuation just vibes honestly the whole thing was wild and nobody even noticed`	No punctuation/ caps	Sentence splitter can't find sentence boundaries
`Lo más increíble es que nadie dijo nada. Everyone just stood there.`	Mixed languages	English model won't understand the Spanish
`K`	Single character	Too short — the function should catch this
`The entirety of chapter one of a long novel pasted here...` (paste a real long paragraph)	Huge input	May be slow; tests the model's limits
`I'm not NOT having a good time (if you know what I mean 😉)`	Sarcasm + double negative	Model can't detect sarcasm

For each input, ask students:
1. What did you expect the model to do?
2. What did it actually do?
3. Why do you think it failed?

Write the failure categories on the shared screen as they emerge:
- **Noise** — emoji, extra spaces, repeated characters
- **Ambiguity** — sarcasm, double negatives, context-dependent meaning
- **Domain mismatch** — slang, mixed languages, formats the model hasn't seen

- **Insufficient signal** — too short, no sentence boundaries
- **Adversarial input** — deliberately confusing text

0:35–0:50 — Name the Failures

Go through the list of failures on screen. Give each category a name.

Talking points:
- "These aren't bugs in our code. The code works fine. The *input* is the problem."
- "Real-world text is messy. People don't type in perfect sentences."
- "Some of these we can fix. Some we can't. Let's figure out which is which."

Draw two columns:

We CAN fix	We CAN'T fix
Extra spaces	Sarcasm
Repeated characters	Cultural context
Abbreviations	Meaning / intent
Emoji	Mixed languages (without a multilingual model)
ALL CAPS	Ambiguity

0:50–1:15 — Fix It: Add clean_text()

Open `app.py` in the Files tab. Add the `clean_text()` function above `find_silliest()`.

Build it step by step, explaining each piece:

Step 1: Strip whitespace

```
text = text.strip()
```

"The simplest fix. Remove junk from the beginning and end."

Step 2: Collapse multiple spaces

```
text = re.sub(r' {2,}', ' ', text)
```

"Turn five spaces into one space."

Step 3: Limit repeated characters

```
text = re.sub(r'(.)\1{2,}', r'\1\1', text)
```

"Turn 'sooooo' into 'soo.' The model might understand 'soo' but definitely not 'sooooooo.'"

Step 4: Expand abbreviations

```
abbreviations = {"Rep.": "Representative", "Dr.": "Doctor", ...}
for abbr, full in abbreviations.items():
    text = text.replace(abbr, full)
```

"Remember when 'Rep.' broke our sentence splitter? Now it won't."

Step 5: Remove emoji

```
text = re.sub(r'[\U0001F600-\U0001F64F...]+', ' ', text)
```

"The model doesn't know what ▯ means. Strip it out so it can focus on the words."

Step 6: Normalize ALL CAPS

```
if caps_count > 3:
    text = text.title()
```

"ALL CAPS might change the model's reading. Normalize it."

Then add one line to `find_silliest()`:

```
cleaned = clean_text(text)
```

And use `cleaned` instead of `text` for the rest of the function.

Commit and rebuild.

1:15–1:35 — Test the Fix

Run the same adversarial inputs from earlier. Compare results.

Questions for students:
- Which inputs work better now?
- Which ones are still broken?
- What's in the "can't fix" column that no amount of cleaning will help with?

Key insight: "Data cleaning is a real job. Data scientists spend a huge amount of time cleaning data before models ever see it. Garbage in, garbage out."

1:35–1:50 — CLEAR Framework

Introduce the CLEAR Framework for prompting AI coding assistants:

Letter	Meaning	Example
C	Context	"I have a Gradio app that uses a zero-shot classifier..."
L	Language	"The code is in Python, using the transformers library..."
E	Explain	"When I paste text with emoji and ALL CAPS, the model gives bad results..."
A	Ask	"Can you add a text cleaning function that..."
R	Requirements	"It should handle emoji, repeated characters, and abbreviations."

Live demo: Open Claude or ChatGPT. Paste the Space code. Write a CLEAR prompt asking it to add input cleaning. Show students the response.

Example CLEAR prompt:

> ***Context:*** *I have a Hugging Face Space that uses a zero-shot classifier to find the silliest phrase in a text passage.*
>
> ***Language:*** *Python, using gradio, transformers, and re.*
>
> ***Explain:*** *When users paste messy text (emoji, ALL CAPS, repeated characters like "sooooo", abbreviations like "Dr."), the model gives unreliable results.*
>
> ***Ask:*** *Add a `clean_text()` function that preprocesses the input before it reaches the model.*
>
> ***Requirements:*** *The function should strip whitespace, collapse repeated characters, expand common abbreviations, remove emoji, and normalize ALL CAPS.*

Say: "This is how you talk to an AI coding assistant. You'll use this a lot in upcoming sessions."

1:50–2:00 — Notebook Time

Share the Colab link in the Zoom chat.

Walk through together:
1. Run the setup cell and load the model
2. Run the "before cleaning" cell together — see the messy input results
3. Run the `clean_text()` definition cell
4. Run the "after cleaning" cell — compare the difference

Say: "The notebook has the same `clean_text()` function we built. Try the experiments — edit the code in the cells to try your own messy inputs."

Notebook skill being introduced: Editing code in a cell (changing the text variable) and re-running

GitHub skill being introduced: "Upload this notebook to your `my-ai-portfolio` repo."

2:00 — Wrap Up

Share the between-session challenge. Encourage them to use CLEAR to ask Claude/ChatGPT for help.

Say: "Next week we're going to put two models head-to-head and see which one is actually better. We'll need to figure out how to keep score."

What Could Go Wrong

Problem	Fix
Editing `app.py` in HF browser editor is fiddly	Have the complete code ready to paste. Show students how to select all → paste.
Regex is confusing for students	Don't explain regex syntax in detail. Just say "this pattern finds repeated characters" and move on. The point is what it does, not how.
Students want to fix sarcasm detection	Great instinct! Explain that sarcasm is an active research problem. "Even humans disagree on sarcasm."
CLEAR demo produces different code than yours	That's fine and even useful. "AI assistants give different answers each time. That's why you need to understand what the code does."
Space rebuild fails after editing	Check for syntax errors. Most common: missing closing parenthesis, indentation errors.

Key Vocabulary (introduce casually)

- **Data cleaning** — preprocessing text to remove noise before the model sees it
- **Noise** — stuff in the input that confuses the model (emoji, extra spaces, weird formatting)
- **Adversarial input** — text deliberately designed to confuse or break a model
- **Feature engineering** — transforming raw input into something a model can work with better
- **Preprocessing** — any transformation applied to data before the model processes it
- **CLEAR Framework** — a structure for writing good prompts to AI coding assistants