#for code completion and introspection/ I didn't know about it till colab errored on my second cell.

!pip install jedi

#Install dependencies ( all the external packages and tools needed to run this project)

!apt-get update -qq

!apt-get install -y -qq openjdk-17-jre-headless

!python -m pip install --upgrade pip setuptools wheel

!python -m pip install --quiet stanza pandas numpy matplotlib seaborn scipy statsmodels scikit-learn openpyxl python-Levenshtein wordfreq language-tool-python

# I put everything in one script: data, functions, analysis, and saves.

import re, math, json, statistics, sys, os

# basic utilities for regex, math functions, saving JSON results, and file management.

from collections import Counter

# counts word and POS frequencies efficiently.

from pathlib import Path

# manages file paths safely across OSes.

import stanza

# Stanford NLP toolkit for tokenization, POS tagging, lemmatization, and syntax parsing.

import pandas as pd, numpy as np

# for structured data handling, numeric computation, and building dataframes.

import matplotlib.pyplot as plt, seaborn as sns

# for plotting (boxplots, heatmaps) and visualizing results clearly.

from scipy.stats import friedmanchisquare, wilcoxon, f\_oneway, kruskal

# provides statistical tests (within- and between-subject comparisons).

from statsmodels.stats.multitest import multipletests

# corrects for multiple statistical comparisons (Bonferroni correction).

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

# for dimensionality reduction (combining syntactic metrics into a single composite).

import Levenshtein

# computes text edit distance to measure how much participants’ drafts differ.

from wordfreq import zipf\_frequency

# gives word frequency scores for lexical sophistication measures.

# LanguageTool init with fallback

# I'm using 'language\_tool\_python' to automatically detect grammar and spelling errors.

# It requires Java to work properly (that’s why I installed openjdk earlier).

# Sometimes in Colab or some systems Java might not connect right, so I also add a fallback:

# If LanguageTool fails to start, I’ll still run a simple "heuristic" error detector

# that catches rare/typo-like words and weird punctuation patterns.

# https://github.com/jxmorris12/language\_tool\_python

# Daniel Naber (2003). A Rule-Based Style and Grammar Checker. Diplomarbeit, University of Bielefeld.

# https://www.danielnaber.de/languagetool/download/style\_and\_grammar\_checker.pdf

has\_languagetool = False

tool = None

try:

import language\_tool\_python

try:

tool = language\_tool\_python.LanguageTool('en-US')

has\_languagetool = True

print("LanguageTool initialized.")

except Exception as e:

print("LanguageTool init failed; will fallback. Error:", e)

has\_languagetool = False

except Exception as e:

print("language\_tool\_python import failed; will fallback. Error:", e)

has\_languagetool = False

#optional but recommended for persistence)

# I’m doing this because when I run code in Google Colab,

# the session runs on a temporary virtual machine (VM).

#

# If I save files only to "/content", they disappear when the session ends.

# So I mount my Google Drive to keep all results permanently.

#

# That way, all my output tables, plots, and JSON summaries

# are stored safely in a Drive folder, not lost when the runtime resets.

#

# I can later open them directly from Google Drive,

# share them, or analyze them again in Excel, SPSS, or R if I want.

#

#Bisong, E. (2019). Google Colaboratory. In Building Machine Learning and

#Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners.

#Apress. https://doi.org/10.1007/978-1-4842-4470-8\_7

# Try to mount drive if in Colab; fallback to local folder if not

DEFAULT\_OUT = Path("/content/syntax\_results")

try:

from google.colab import drive

try:

drive.mount('/content/drive', force\_remount=True)

OUT\_DIR = Path("/content/drive/MyDrive/syntax\_results")

OUT\_DIR.mkdir(parents=True, exist\_ok=True)

print("Google Drive mounted. Outputs:", OUT\_DIR)

except Exception as e:

print("Drive mount failed — falling back to local /content. Error:", e)

OUT\_DIR = DEFAULT\_OUT

OUT\_DIR.mkdir(parents=True, exist\_ok=True)

print("Outputs (local):", OUT\_DIR)

except Exception as e:

OUT\_DIR = DEFAULT\_OUT

OUT\_DIR.mkdir(parents=True, exist\_ok=True)

print("Not in Colab or drive import failed; using local output:", OUT\_DIR)

PNG\_DIR = OUT\_DIR / "plots"

PNG\_DIR.mkdir(exist\_ok=True)

print("Outputs: ", OUT\_DIR)

# Stanza setup

# I’m using Stanza — an open-source NLP library from the Stanford NLP Group as mentioned earlier.

#

# It lets me tokenize, tag parts of speech (POS), lemmatize, and analyze dependency syntax in English.

#

# This is essential because I want to measure syntactic complexity.

print("Downloading Stanza english model (one-time, may take ~30-60s)...")

try:

stanza.download('en', verbose=False)

nlp = stanza.Pipeline('en', processors='tokenize,pos,lemma,depparse', use\_gpu=False, verbose=False)

stanza\_ready = True

print("Stanza ready.")

except Exception as e:

stanza\_ready = False

nlp = None

print("Stanza download/init failed; syntax parsing will be skipped. Error:", e)

# All participant texts (embedded)

text1\_1 = ''' It was so disappointing when I found out my phone is out of service in the middle of city that was supposed to be much better than I was expected! I tried to explore the city a bit and reach out to the citizens and asking if they can help me with my phone’s singnal but I realized they can barely speak English and the worse part was that I didn’t have any internet to use translator. finally I could find a police there and he could help me with his broken English and the first thing I did after finding service was to buy return ticket. By far that was the worse mysterious trip that I have ever taken in my life. '''

text1\_2 = '''When the plane landed, I expected the usual announcement welcoming us to a familiar destination. Instead, the captain’s voice came through with an unsettling vagueness: “We have arrived. Enjoy your stay.” No city name, no further details—just that. Curious but slightly uneasy, I reached for my phone to check my location, only to find that I had no signal. No service, no WiFi—completely disconnected. Hoping to find help, I wandered through the airport and approached locals, but communication proved nearly impossible. Few spoke English, and without internet access, I couldn’t even use a translation app. My attempts at gestures were met with polite smiles or blank stares, making the situation all the more frustrating. Eventually, I found a police officer and asked for assistance. He squinted at me, trying to understand, and after a few moments, he managed to explain in broken English that my phone simply wouldn’t work there. His only advice? “Wait.” Wait for what, exactly? He didn’t say. The moment I managed to get a weak signal, I didn’t hesitate—I booked the first flight out. Even now, I can’t shake the feeling of that strange, unmarked city. No signs, no recognition on any map, and for a brief moment, it felt as if it existed in complete isolation from the rest of the world. '''

text1\_3 = ''' Losing one's passport after landing in a new place might have its own perks. After finding out such a thing has happened to me or someone else, one might need to contact the airport police ASAP and ask for help. In case the embassy was closed, I would ask forfinger prints and face recognition in order to confirm my identity and if possible get back to my country after that confirmation. '''

text2\_1 = '''my phone stops working exactly when i needed it the most, i am very frightened as i couldn't catch what city we have landed when the captain mentioned it few minutes ago, i don't know was it because of the poor quality of the microphone or the pagers or speaking incomprehensibly is among the responsibilities of every captain in this world. i have been to Dubai's airport many times before, where i was supposed to have my connection flight for 2 hours, but here looks like anything but Dubai! the only thing they have in common is that both of them are in the middle of the dessert but what frightens me is that if our flight has been hijacked cause here looks like an abandoned airport with no staff, nor other flights. besides missing my next flight that departs for Rome, i don't know what will happen for our safety and everybody else seems as confused as i am and none of the flight attendants are present to help or respond to our questions. here is very chaotic but everyone is talking in their maternal language and the man sitting near me couldn't speak any other language but arabic, so even though he was trying his best to explain to me but he left me in more confusion. after some fake head nodding i zoned out after hearing the word 'Yemen' which is in war at the time.'''

text2\_2 = '''My phone stops working exactly when i need it the most. I am very frightened and confused, like i just woke up from a nightmare. I didn't catch the city name when the captain announced it a few minutes ago. Was it because of the poor quality of the microphone? Or perhaps speaking incomprehensibly is just part of a captain's job everywhere in the world? Two hours ago, I was supposed to have my connection flight in Dubai. But now i don't even know where i am! This place looks nothing like Dubai. The only thing they commonly share is that probably both are in the middle of desert. I am afraid that our flight has been hijacked. Here looks like an abandoned airport. No staff. No other planes. Endless desert and just silence. I don't know what will happen to us. Inside the cabin is very chaotic and as far as i can see, there are no flight attendants to help us or respond to our questions. A man sitting behind me gestures frantically, speaking rapid Arabic which i don't understand. i could only catch one word: 'Yemen'. My stomach drops. Yemen? The country at war? I nod, pretending to understand, but my mind races.'''

text2\_3 = '''Finally, after many years, I arrived at Tokyo which means that my childhood dream became the new reality, but also at the same time my childhood nightmare decided to become the reality in this competition. I lost my passport, not like I care about that shit colour piece of paper, but the real question is how I can go back home? I can’t say I panicked just because my arrogance doesn’t let me to show that even to myself. Afterall, I have a powerful self-gaslighting ability to remain calm, so I decided to walk back all the places I went today. I walked the same path three times. My mom says, ‘Third is the charm!’. No trace of it anywhere. I saw a police officer on the way and asked for his help, but he couldn’t speak English and my only knowledge in Japanese is ‘Konnichiwa’. I walked past by him with a deep frustration, and I decided to go to the police station, maybe somebody there knows English. At this point I had lost my hope completely, but my appetite seems to be unbothered. It was almost 3 p.m. and I had a complete lunch at a local restaurant two hours ago. I must have done a marathon to be this hungry. I got some miso soup and onigiri from a wending machine in the station. I sat there eating in a parking lot like a homeless man and watched the trains passing in front of my eyes. I arrived at the main police department in the same neighbourhood that my hotel was located and hopefully there was a very helpful young guy who helped me out filing a report. He was very handsome. I wasn’t thinking about my passport at all while I was talking to him. I couldn’t even initiate to flirt because I was struck by his energy. He must have not noticed this at all and thought that I may be shocked because I lost my passport the same day. I walked back towards the hotel I stayed in thinking about that police officer. I walked in the lobby and as I was looking at the new arrivals checking in, I realised I might have just left my passport in the reception. So, I walked faster and asked the receptionist in a rushed manner if I have left my passport with them? I can’t forget her terrible smile. She shook her head in the most awkward manner possible and I was stunned about how stupid I was today.'''

text3\_1 = '''I would initially look for a map of the area so I can guide myself. Since my phone doesn't have reception, I would probably look for a call center or public phones so I can contact a friend or parent. If everything goes as planned, I can stay there and enjoy the trip, if not, I would try to buy a return ticket as soon as possible.'''

text3\_2 = '''If I found myself in an unfamiliar place without phone reception, here’s what I would do. First, I would look for a map of the area to find my way around or ask someone where I am. Since my phone wouldn’t have reception, I’d probably look for a call center or a public phone to contact a friend or a parent. If everything went as planned, I’d stay and enjoy the trip, maybe visit some places, try local food, and have fun. Otherwise, I’d try to get a return ticket as soon as possible because, unfortunately, being in an unknown place comes with many risks, and sometimes it’s safer to leave.'''

text3\_3 = '''I will probably wait for the embassy to reopen, explain what happened and hope they can help me so I can go home. Until then I will visit the city, because I am there for that purpose, but I will always be anxious because I don't know if they found my passport or not, and I don't know if I can go home or not. So, feeling anxious and worried, i will "enjoy" the city hoping for the best.'''

text4\_1 = '''As my plane landed, the captain announced a city I had never heard of, an unfamiliar destination and definitely not my intended one. In a mix of confusion and curiosity and trying to understand the reason of this unannounced plan change, panic set in me as soon as I checked my phone seeing that I had no signal. Seeking for some explanations, I approached a confused flight attendant that, after a brief check with the captain, explained me that it was a necessary rerouting due weather conditions. As I disembarked, waiting for a rebooking with the other passengers, I found myself thinking about that this unexpected delay could be an opportunity to explore a new place.'''

text4\_2 = '''As my plane landed, the captain announced a city I had never heard of, an unfamiliar destination and definitely not my intended one. In a mix of confusion and curiosity, I. struggled to understand the reason behind this unannounced change of plan. Panic washed over me as soon as I checked my phone seeing that I had no signal to contact anyone or figure out what was going on. Seeking explanations, I approached a confused flight attendant who, after a brief check with the captain, explained to me that it was a necessary rerouting due to weather conditions. As I disembarked, waiting for a rebooking with the other passengers, I found myself thinking that this unexpected delay might be an opportunity to explore a new place'''

text4\_3 = '''I always wanted to visit Tokyo. Finally, I bought a ticket and arrived on a sunny Friday afternoon with the intention to spend an entire week exploring around. Everything felt perfect, until the moment I realized something was missing: my passport. Standing right in front of the exit of the airport, I checked all my bags, pockets, and even the airport floor. I couldn’t find it anywhere, It was gone. I felt panic in my chest. Trying to stay calm, I checked with airport security, lost and found, the airline, but nothing. Without a passport, I probably couldn’t check into a hotel and for sure I couldn’t fly home. I went to the embassy right away, but it was already closed. A sign on the door said it would open again on Monday morning. I took a deep breath and tried to calm down. I still had my wallet, phone, and ID card from my country. I called the hotel that I booked to explain my situation and fortunately they accepted other documents and let me stay. Then, I went to the police station to report the lost passport. They gave me a paper that said I had reported it. I took a photo of it with my phone, just in case. I spent the weekend carefully. I stayed close to the hostel, visited a few places nearby, and kept all my things very safe. I couldn’t enjoy as much as I wanted these first days of my vacation and, on Monday, I went to the embassy very early. They helped me and gave me a temporary passport so I could travel home. Form that moment on I could enjoy better the vacation I planned, but I also learned something really important: stay calm, think clearly, and always keep your documents in a safe place.'''

text5\_1 = '''If my plane lands in a different cuty I will panic and ask aanother passanger whose phone work for calling my family for some suggestion. It also depends where it lands, if it lands in a beautiful place I will ask for money from my family and try to explore the city. At the same time I'll try to buy a tourist simcard so I am able to contact my family from my phone and have internet. '''

text5\_2 = '''My first instinct is to check my phone in panic if my plane lands in a city I've never heard of. I will ask someone if I can use their phone to call my family and get suggestions if there is no signal. I will attempt to stay in the city if it's pleasant. So that I can travel, I'll ask my family to send me some money. Additionally, I'll attempt to buy a tourist SIM card so that I can call them using my personal phone. Even though it's a little challenging, I'll try to remain calm and find the positive side of things.'''

text5\_3 = '''If my plane lands and I recognize that I have lost my oassport, I will be stressed and try to talk ti the police officers at the airport. It might be challenging since I need to show that I am not an illigal immigrant or a terrorist. After some checks, I might need a lawyer who speaks my language and can talk to the embassy to solve the problem. Overall, this is going to be a very scary situation and I need to control my emotions.'''

text6\_1 = '''As I tried to comprehend the situation, everyone started freaking out. I quickly unfastened my seatbelt and jumped out of my seat in the aisle, to take a look outside the window. I couldn’t see anything, there were dirt and dust as far as it could be seen as if we were led there by a tornado. I tried to listen carefuly but it was impossibele to hear the captain’s buzz through the chaos. I finally got up and tried to find a way between those narrow aisels like it was supposed to lead to somewhere! I kept pushing people to find a way when I bumped into someone. Before I could apologize to him, I heard a familiar voice froze every single cell in my body “I thought I would never see you again ...”. I finally got the courage to look into his big doe eyes and mumbled: “wha ..at.. ar..re...you... do..doing here?” “Finding you!” he answered with the same old smirk on his lips. My first crush, highschool sweetheart, and later my biggest regret, Damien Brody, who broke my heart by moving to another state without saying goodbye. He grabbed me by my arm without any hesitation and dragged me to the nearest emergency exit. While I was standing there processing the biggest shock of my life, he finally kicked open the door and jumped out of the plane while holding me! The soil we walked on was red and shiny as if we just landed on Mars! Still in shock I asked: Where are we? –“Area 81”. “Ok very funny, I have heard about area 51, but there is no such thing as area 81!” “You havent heard about it because it did not exist until last week” he answered. Whithin 10 minutes everyone was off the plane, wondering around in disbelief. “Where are we? Why arent we in an airport?” an old lady asked. We were so distracted by the name of the city that we have almost forgotten about the fact that a plane should land in an airport, not barren lands! “Stay close to me” Damien whispered to me, while some sort of millitary army were encompassing us. Suddenly we heard a deafening noise and after that, total blackout. I woke up feeling nausous and in pain. I saw a pair of blue eyes as I was opening my eyes which disappeared at the second. Were am I? What is happening? I couldn’t move my arms or legs as if I was paralysed. A couple of meters away from me there was Damien, also lying on a table. He turned his head towards me and whispered: we were broght here as chosen ones. –“What ... What the hell are you talking about?” He continued: havent you always asked yourself why I left all of a sudden? I was with them! They took me! -“Them? Who are they?” He tried to elaborate: “I tried to stay away from you, but they know about you... your genes ... our genes...”. My blood froze, how did he know I had his baby? I didn’t even know I was pregnant when he left, how could he know? It was impossible. “ I know Rachel ... “ As if he has read my thoughts. I couldn’t hold my tears and burst into tears: No you won’t ever take her from me! “Lower your voice, I am trying to save your ... our daughter!” he shushed me. He continued: “We have the rarest gene called Alfa-Botinol-S, a gene that has been created as a result of a rare mutation, which normally is silenced, but by triggering it, you can reprogram every single cell in your body, you become immortal do you understand?” he continued: “Do you know how you can trigger this gene naturally? By having two species with this gene to create an offspring!” “Lilly ...” I gasped. Damien approved by nodding his head. “No it can’t be ... it can’t be ... she is just a normal child!” I whispered. A man entered the room before we could finish our converstion. I started screaming as I saw my little Lilly being dragged into the room by two people dressed totally in white. “No ... no you cant take my daughter ... you can’t take her ...” I screamed as I woke up from my terrible nightmare. “Dear passengers please remain in your seets as we are landing in Octaone City” the captain announced. Wait why haven’t I heard about this city before?'''

text6\_2 = '''As I struggled to comprehend the situation, panic erupted around me. Passengers gasped, shouted, and clutched their phones—only to realize they had no signal. My pulse quickened. I unfastened my seatbelt in a rush, bolting from my seat and stepping into the aisle. I needed to see where we had landed. Pressing my face against the nearest window, my breath hitched. There was nothing but a vast expanse of dust and dirt stretching endlessly, as if we had been swept into this place by a storm. The eerie stillness outside sent a chill down my spine. I tried to focus on the captain’s announcement, but his voice was drowned out by the rising chaos. Heart pounding, I pushed my way through the narrow aisle, as if moving forward would somehow lead me to answers. I weaved through the panicked passengers, shoving past bodies in my desperate attempt to find clarity—until I slammed into someone. Before I could utter an apology, a voice stopped me cold. “I thought I’d never see you again…” My breath caught in my throat. Every muscle in my body tensed. That voice—familiar yet distant—froze me in place. Slowly, I forced myself to look up, locking eyes with the last person I ever expected to see. Damien Brody. My first crush. My high school sweetheart. And my greatest regret. The same boy who had shattered my heart by vanishing without a word, moving to another state as if I had never existed. He smirked—the same knowing, cocky smirk I had once fell for. I swallowed hard, my voice barely a whisper. “Wha… what are you doing here?” “Finding you,” he answered simply. My stomach twisted. Before I could process his words, he grabbed my arm and pulled me toward the emergency exit. My mind reeled, still trying to understand what was happening, let alone why he was here. Then—before I could resist—he kicked open the exit door and leaped out, dragging me with him. We hit the ground hard. A strange metallic scent filled the air. Dazed, I looked down. The soil beneath us wasn’t normal—it was red and glistening, like molten glass. It looked like we had just landed on Mars. Still in shock, I turned to Damien. “Where the hell are we?” He didn’t hesitate. “Area 81.” I let out a dry laugh. “Very funny. I’ve heard of Area 51, but there’s no such thing as—” “You haven’t heard of it because it didn’t exist until last week.” Before I could press him further, the ground trembled. Silhouettes emerged from the dust, armored figures moving with unnatural precision. The deafening screech of an alarm split the air. Then—blackness.'''

text6\_3 = '''I frantically checked all the pockets, disappointed because my passport was gone. Panic struck my chest and I froze, looking around at the crowded airport. The embassy would not be open until Monday. My phone had no signal. I made myself inhale and swallowed. The police first. I would get a proper record and report the loss. I would then ring the embassy and explain the situation over the phone and by emails. I would ask my family for scanned copies of my records or for emergency money in case. Despite the chaos, I would not give in to fear. Seeking assistance, indicators, and a path ahead, I scoured the airport. To be able to tour the city I had dreamed of without this wreck ruining it all, I would find a temporary solution, like an emergency ID or travel permit. I kept looking out for someone who would be willing to help me cut through the crowd, whether it was an enthusiastic fellow traveler, a member of airport personnel, or even another lost tourist. Every little hint was a lifeline. I was telling myself that this hassle was part of the process and that, with determination and planning, I would be able to turn this failure into something that I would be recounting for decades. I finally located a peaceful place and devised a plan. I would consider what I would do next, place calling the embassy in the priority category for Monday, and collect any necessary documents in the meantime. I felt to be in control of every small thing that I was doing, and fear started to recede a little with each action. I realized that I could still find my way through this crisis, even with uncertainty, and maybe even learn something new about the city along the way.'''

text7\_1 = '''As the plane landed, I did not know where I was. My phone does not show any signals, it reminded of a movie called "Lost". The plane was landed in the middle of nowhere. At first I thought I was in a dream! I do not have any memory where this trip started. Now, I know I am awake in a city I do not know anything about, I do not know any of the passengers and the flight attendants do not respond to my questions. I need to have a plan. I will get off the plane and reach to the information desk to ask how I can contact my family and friends; that is the priority. Although this a strange situation, I can enjoy it! Maybe this is a rare opportunity to discover a city I did not know anything about before for a day till I can get back to my actual destination. I have contacted my family and everything is calmer now. I got my visa on arrival and now I am going to the city. The city is astonishing! It is a combination of a modern and ancient city. I have asked the citizens about a good place to have long with body language! It is hard to get there when I do not know the city but it is a kind of adventure. I only have about half a day to return to airport because the airport agency booked me a ticket for my actual destination. The food was perfect! I feel that this was a great chance to see this place. This trip taught me I can enjoy unpredictable happening too. I have learned many thing from this trip and enjoyed every moment of it. Now I have a story to narrate for my friends!'''

text7\_2 = '''As the wheels of the plane hit the the wonderland, I was hearing my heartbeat! I was in the middle of nowhere. I was looking around and all the passengers was confused like mine. Flight attendants were wondering around. I was glazing at my phone and the screen was showing no signal sign. I was in the middle of nightmare, the situation reminded me of the series called Lost. Meanwhile I thought maybe this is an opportunity I can discover a new city. As I get off the plane, I went right through the airport to talk to information desk and the immigration office to get my visa on arrival to explore the city. After a long discussion I finally got into the city. The city has the sense of a modern architecture with sign of ancient ruins. The city was astonishing, the smell of cherry blossom was in the air. I asked natives to find a perfect place for lunch. It was hard but I used a body gesture by robing my stomach to transfer the meaning of hungry. The adventure was perfect. I learned I can enjoy unpredictable happenings too. No I have a story to narrate for mu friends.'''

text7\_3 = '''At first, since I am an anxious person, I predict that I will panic. Therefore, the first thing I should manage is my health condition and reassure myself that nothing catastrophic is going to happen. It is very common that embassies are closed at the weekend and obviously I can not do anything to get my new passport during the weekend as a normal citizen of my country. I suppose the best choice in this situation is to enjoy the city I have always wanted to visit. The only thing I need to avoid is to visit places where need a valid ID. Therefore, to resolve this situation I will go to my hotel and plan a magnificent visit of the city for the whole weekend. Meanwhile, I notify my family and friends about my situation, in case any emergency happens so they can contact the embassy. In addition, I would not worry about staying in the country with valid tourist visa, because most tourist visas are valid for about three months. So, after enjoying my visit, I will go to the embassy as soon as possible after the weekend before my reservation ends at the hotel; in case anything happens, I will extend my stay at the hotel for one to two more nights. In conclusion, the situation can be managed easily, an I can enjoy sightseeing the city at the weekend.'''

text8\_1 = '''Well that is something new, did not see that coming tho. The plane force landed in an unkonwn place located right in the middle of nowhere and now they are asking everyone to leave the plane! what a brilliant idea! I got to be honest about this, if it was up to me, I would rather to stay here in the plane, safe and sound cause that's usually where all the trouble starts, LEAVING THE SAFE SPOT! I don't want to sound too dismissive but we all know how it will go, or should I say "how it went"? okay, bare with me a few more lines. I ain't gonna start over and do a recap for you. the rest is the same old scripts of armagedon movies in which some men get locked in an unknown wicked iseland struggling to survive through series of apocoliptical events and acyion scenes getting murdered one by one up to the final scene where the boy hugging her lover watching her bleeding to death whilst in the background you can hear kesington singing "There I'm all alone again...". the end'''

text8\_2 = '''The plane shuddered, screamed, and fell. No warning. No grace. Just a mechanical beast losing its war against gravity. Gasps turned to prayers, prayers turned to silence. And then—impact. A groan of twisted metal, the earth swallowing us whole. When the dust settled, we were somewhere unknown, stranded in the middle of nowhere. That’s when the captain told us to leave. Brilliant idea. If it were up to me, I’d chain myself to this seat and declare squatter’s rights. Think about it—when was the last time a guy who stayed put got mauled by some cosmic horror? No, it’s always the idiots who say, “Let’s split up and explore” that end up with their intestines decorating a tree. And now they want me to step out, unarmed, unprepared, and—let’s be honest here—probably the first in line to get devoured? No, thanks. But let’s not pretend we don’t know how this goes—scratch that, how it already went. The music swells, time folds in on itself, and suddenly we’re caught in some Nolan-esque fever dream. The wreckage flickers like déjà vu, a limping Cillian Murphy mutters something cryptic, and Leonardo DiCaprio stumbles past, whispering about layers of reality. Then, in the distance, a scream—cut short, swallowed by something unseen. And now, here we are, trapped in a story we never chose to be part of. Then the killing starts. The first guy vanishes, torn apart so fast his shadow still lingers. The second explodes from the inside out, his own blood painting the sky. Someone tries to run—bad move. A clawed hand snatches him mid-stride, splits him clean in half like a meat puzzle. “Well, that’s a right bloody mess,” Butcher grumbles, flicking a cigarette onto a pile of what used to be a person. More carnage, more screams, more bodies reduced to shredded remnants of what was once called human. “Fucking diabolical.” And then, the final scene. The boy cradles his lover in the mud, her lifeblood seeping into the earth like ink spilled on an unwritten page. A gust of wind brushes past, indifferent to her last breath, to his whispered pleas, to the warmth fading between his fingers. Kensington hums in the background, “There, I’m all alone again…” as the camera pulls back—slowly, deliberately. The wreckage, the bodies, the island swallowing them whole. The boy doesn’t cry. He doesn’t scream. He just sits there, holding onto something already gone. The shot lingers. Then, the screen fades to black. Silence.'''

text8\_3 = '''Well, that's a buzz killer I guess. The very first thing to do Before getting started with the process, is to go to a local police station and make a declaration on the passport being lost just in case if someone finds it then to look for a place to stay without a passport. I guess I can get some help online looking for a host in one of those social media platforms but first and foremost I got to find a place to eat something cause it ain't promising to do it with an empty stomach. Having that all settled, I can start with searching online for solutions. Let's see how it went for others with similar problems, what are the suggestions by the embassy in those sort of situations, how to get some help from the local government and figure out the best procedure to follow up. Then being patient is crucial, I mean come on! I got some days to waste in my favorite city and the paperwork can not be done in a faster way! Hopefully by the start of the week, there will be some silver lining and the problem will get resolved one way or another! '''

text9\_1 = '''If I am at a city I don't know that the airplane land, I call the flight attendant and I will ask what this city is and why I am taken to this city by mistake. I shall analyze the situation and see if there is any solution. I have never had this experience that is why thinking about it is difficult'''

text9\_2 = '''If I don’t recognize the city after I land, I stay calm and talk to a flight attendant to ask where we are. After that, I turn off flight mode and see if I have signal. If there is no signal, I would feel nervous and unsettled. It is important to think logically and solve the problem. After that, I find a solution to get back to my city. I think it is important that people help each other. '''

text9\_3 = '''If I lose my passport after landing in my destination several feelings will come to my mind and experiencing all of them can be unsetteling. I would ask for help by talking to other people as they have any solutions just because the embassy is closed and don't open till monday morning. I would try to make friends and. see if I can stay with them because I don't have passport to show to my hotel. I will remain hopeful that the embassy solve my problem fastly and that I can enjoy my travel.'''

text10\_1 = '''The plane landed softly and the captain announced that we had arrived at Tikaka. ' What kind of joke is that?!' I asked. I checked other passengers' faces! they were so relaxed! it seems like they have expected that distinction! I wasn't still sure what was happening so I reached for my phone to check the location. 'NO SIGNALS'. The phone was practically useless. I asked one of the crew about the situation. She replied with a warm smile that was in Tikaka and went away! Where the hell is this place!!!! what the hell!!! '''

text10\_2 = '''The plane landed softly, and the captain announced, “We have arrived at Tikaka.” “What kind of joke is that?!” I asked. I checked the other passengers' faces—they were so relaxed! It seemed like they had expected this distinction. I still wasn't sure what was happening, so I reached for my phone to check the location. No signal. The phone was practically useless. I asked one of the crew about the situation. She replied with a warm smile, “You are in Tikaka,” and went away. Where the hell is this place? What the hell?! Suddenly, someone tapped me on the shoulder and kindly said, “We both know where we are.” I was shocked. As I got off the plane, I asked, “Where are we? What is this place?” “I will tell you,” he said. After a few moments of awkward silence, he added, “People arrive by accident, but leaving isn’t simple—you have to understand why you were brought here.” “What do you mean?!” I asked. “I will explain this later. Wake up.” And suddenly, I woke up.'''

text10\_3 = '''I arrived in Barcelona on a bright Saturday morning. I had always dreamed of visiting this city, and I was feeling very excited. After I had walked out of the airport, I checked my bag. Suddenly, I realized that my passport was missing. My heart was beating fast because without it I could not check into my hotel. I went to the police station near the airport and explained my problem. The officer listen carefully but he told me the embassy had already closed for the weekend. I felt nervous, however; I was trying to stay calm. I decided to call my friend Samaneh in the US. She suggested that I should keep my passport photocopy and my ID card with me at all times. She also told me to stay in a safe place until the embassy opened. I had booked a small hostel using my ID and the copy. I was lucky that he owner was friendly and said it was fine! He accepte the picturs of my documents! During the weekend, I was walking through some streets of Barcelona and was taking many pictures. I kept thinking about my passport! On Monday morning I went to the embassy and finally have them give me the new one! :))'''

text11\_1 = '''I have always loved adventure and visiting new places.nothing can stop me enjoy the moment even if the destination is not what I planned for. I am sure with the travel insurance purchase, I am gauranteed a return of my money after getting back to my city. So, why not to enjoy this moment? From Africa, to America, All cities have their own beuties and can teach me new thing. I can learn a new wxperience and meet new people. The only thing that may bother me is thinking about landing in an unsafe place which hopefully is not the case here. To recao, I'll make the best out of every situation and won't ruin the day. I will just contact my boyfriend and family after making sure about my safety from someone else's phone as mine doesn't have signal.'''

text11\_2 = '''I have always loved adventure and discovering new places. Nothing can stop me from enjoying the moment, even if the destination isn’t what I had planned. Since I have travel insurance, I know I’ll get my money back once I return home. So, why not make the most of this unexpected journey? From Africa to America, every city has its own beauty and something new to teach me. I can have unique experiences and meet interesting people along the way. The only thing that worries me is landing in an unsafe place, which I hope isn’t the case here. To recap, I will make the best out of every situation and won’t let anything ruin my day. I’ll contact my boyfriend and family after making sure I’m safe, using someone else’s phone since mine doesn’t have a signal.'''

text11\_3 = '''I always enjoyed adventuring and exploring. No matter what happens or whether things go my way or not, I will always maximize my moment regardless of what happens. It's comforting to know that I can get my money back if I go home since I have travel insurance. Why not take advantage of this surprise situation? Each city, from Africa to America, has its own beauty and has something to learn. Even though this holiday isn't really the one I was hoping for, I am still able to go on great adventures and make new friends. Arriving somewhere that's dangerous is the only thing which actually scares me, and I really hope that's not the case here.'''

text12\_1 = '''Well, isn’t this just my kind of luck? Captain starts rattling off some city name I swear I’ve never heard before, right as we’re landing. And yeah, my phone picks now to go totally dead—no signal, zilch. So, great, apparently I live here now? Honestly, it freaks me out for a sec. But hey, I keep my cool. I mean, weird places keep life interesting, right? Guess this is my shot to go full Dora the Explorer. Outside the window, I’m staring at these streets and buildings that look like they belong in another timeline. Something about it kinda excites me, though. Who knows what weird snacks I’ll find, or who I might bump into? Adrenaline’s there, but so is that little itch to dive in. First off, gotta not die. Seems important. I’ll hunt down someone who speaks enough English to point me in the direction of “civilization,” let my family and boyfriend know I’m, you know, still on planet Earth. After that? Bring on the accidental tourist moments—photos, getting lost on purpose, maybe making some stories worth telling. Not how I pictured today going, but screw it, I’m rolling with it. check the text for any improvements'''

text12\_2 = '''Well, isn’t this just my kind of luck? The captain starts rattling off some city name I swear I’ve never heard before—right as we’re landing. And, of course, my phone picks now to go totally dead. No signal. Zilch. So, great. Apparently, I live here now? Honestly, it freaks me out for a sec. But hey, I keep my cool. Weird places keep life interesting, right? Guess this is my shot to go full Dora the Explorer. Outside the window, I’m staring at streets and buildings that look like they belong in another timeline. Something about it excites me, though. Who knows what weird snacks I’ll find or who I might bump into? Adrenaline’s there, but so is that little itch to dive in. First off, gotta not die. Seems important. I’ll hunt down someone who speaks enough English to point me toward “civilization” and let my family and boyfriend know I’m, you know, still on planet Earth. After that? Bring on the accidental tourist moments—photos, getting lost on purpose, maybe making some stories worth telling. Not how I pictured today going, but screw it. I’m rolling with it.'''

text12\_3 = '''Well, this isn't exactly how I thought I'd be starting out on my adventure. I finally arrive in the city that I'd been dreaming of, and then I recall my passport. Instantly, all my excitement turns to panic. And to add insult to injury, the embassy is shut for the whole weekend, so I know I'll be having to sort it out myself for the moment. At first, I’d probably freak out a little, but then I’d force myself to calm down. Stress won’t bring my passport back. The first step is to retrace my steps and check the airport, my bags, or even ask lost and found if someone turned it in. If that doesn’t work, I’d head to the nearest police station to file a report. At least it would make life easier later on with the embassy. Since I won't be able to acquire a new passport at once, I'd simply focus on remaining safe and making the most of my weekend. I'd take the police report and photocopy of my travel insurance with me wherever I go, just in case I need to present evidence of my identity. I'd also tell my family and boyfriend sooner rather than later so they're aware of what had happened and maybe receive some guidance or counsel. Of course, I wouldn't get too far without a passport, but that doesn't mean I would be confined to my hotel room the whole time. I'd still go into the city some, take photographs, eat the cuisine, and enjoy myself as much as possibl, just more carefuly than usual. When Monday comes, I’ll go straight to the embassy with my documents and the police report to ask for an emergency passport or travel document. Problem solved or at least, eventually. For now, I’ll try not to let this ruin my trip. After all, adventures aren’t always perfect, and sometimes the best stories come from the unexpected.'''

# Building texts dictionary

texts = {

"participant1": {"1": text1\_1, "2": text1\_2, "3": text1\_3},

"participant2": {"1": text2\_1, "2": text2\_2, "3": text2\_3},

"participant3": {"1": text3\_1, "2": text3\_2, "3": text3\_3},

"participant4": {"1": text4\_1, "2": text4\_2, "3": text4\_3},

"participant5": {"1": text5\_1, "2": text5\_2, "3": text5\_3},

"participant6": {"1": text6\_1, "2": text6\_2, "3": text6\_3},

"participant7": {"1": text7\_1,"2": text7\_2,"3": text7\_3},

"participant8": {"1": text8\_1,"2": text8\_2,"3": text8\_3},

"participant9": {"1": text9\_1,"2": text9\_2,"3": text9\_3},

"participant10": {"1": text10\_1,"2": text10\_2,"3": text10\_3},

"participant11": {"1": text11\_1,"2": text11\_2,"3": text11\_3},

"participant12": {"1": text12\_1,"2": text12\_2,"3": text12\_3},

}

# Utilities & measures (same as before, but compacted)

# In this section, I'm defining small helper functions that measure

# different aspects of lexical diversity and syntactic complexity.

# Tokenization:

# I first define a simple tokenizer to split text into words,

# ignoring digits and punctuation. This helps make my measures consistent.

#

# I use a group of lexical diversity and sophistication measures to quantify

# how rich and varied a participant’s vocabulary is.

#

# These measures are common in computational linguistics and second language writing research.

#

# They give me an idea of how advanced or repetitive the vocabulary is across drafts (texts 1–3).

#

# These metrics come from L2 writing and computational linguistics research.

#

# Full biblio will be orovided in the abstract.

WORD\_RE = re.compile(r"\b[^\d\W]+\b", re.UNICODE)

# I define a regular expression that matches alphabetic word tokens

# (ignores digits and punctuation). This forms the basis of tokenization.

def tokenize\_words(text):

# I split the text into lowercase word tokens using the regex above.

# This ensures consistent tokenization for all lexical measures.

return WORD\_RE.findall((text or "").lower())

def types\_tokens(tokens):

# I return two counts:

# (1) number of unique word types

# (2) total number of word tokens

# These are the foundation of type–token ratio metrics.

return len(set(tokens)), len(tokens)

def ttr(tokens):

# Type–Token Ratio (TTR): proportion of unique words in the text.

# A basic measure of lexical diversity — higher TTR = more varied vocabulary.

a,b = types\_tokens(tokens)

return a/b if b else 0

def cttr(tokens):

# Corrected Type Token Ratio (CTTR): adjusts TTR for text length bias.

# Formula: types / √(2 × tokens)

# I use this because raw TTR decreases with longer texts.

a,b = types\_tokens(tokens)

return a/math.sqrt(2\*b) if b else 0

def hapax\_rate(tokens):

# Hapax Legomena Rate: proportion of words that occur only once.

# Indicates how much of the vocabulary is “non-repetitive.”

# Higher hapax\_rate = richer, more varied lexical usage.

if not tokens:

return 0

c = Counter(tokens)

return sum(1 for w,count in c.items() if count==1)/len(tokens)

def mtld(tokens, ttr\_threshold=0.72):

# Measure of Textual Lexical Diversity (MTLD):

# A robust, length-independent measure of lexical diversity.

#

# It computes how many words are needed before TTR drops below 0.72,

# and averages this over forward and backward passes of the text.

if not tokens:

return 0

def one(seq):

types=set(); token\_count=0; factors=0.0; types\_count=0

for w in seq:

token\_count+=1; types.add(w); types\_count=len(types)

if token\_count>0 and (types\_count/token\_count) <= ttr\_threshold:

factors += 1; types.clear(); token\_count=0

if token\_count>0:

if (types\_count/token\_count) != 0:

factors += (1 - (types\_count/token\_count) / ttr\_threshold)

return len(seq)/factors if factors!=0 else len(seq)

return (one(tokens) + one(list(reversed(tokens))))/2.0

def avg\_zipf\_freq(tokens):

# Average Zipf Frequency: I use the wordfreq library to find how common

# each word is in general English (on a 1–7 Zipf scale).

#

# Lower average Zipf values = rarer vocabulary (more advanced lexis).

if not tokens:

return None

vals=[zipf\_frequency(w,'en') for w in tokens]

vals=[v for v in vals if (v is not None and not math.isinf(v))]

return sum(vals)/len(vals) if vals else None

def lexical\_sophistication(tokens, rare\_threshold\_zipf=3.5):

# Lexical Sophistication: proportion of "rare" words in the text

# (words below a certain Zipf frequency threshold, default 3.5).

# This quantifies advanced, less frequent word use.

if not tokens:

return None

rare = sum(1 for w in tokens if (zipf\_frequency(w,'en') < rare\_threshold\_zipf))

return rare/len(tokens)

# Stanza helpers

# In this part, I use the syntactic analysis produced by Stanza

# (tokenization, POS tagging, and dependency parsing)

# to extract key measures of syntactic complexity.

def mean\_sentence\_length(doc):

# I calculate the average number of words (excluding punctuation)

# per sentence in the parsed Stanza document.

# Longer sentences often indicate more syntactic complexity.

#

#I also acknowledge that: Sentence length is a surface proxy —

#a long sentence can still be syntactically simple (e.g., lists).

#

#Therefore, I combine it with other metrics like subordinate clause count,

#dependency distance, and tree depth to ensure that “longer = more complex”

#holds only when supported by underlying structure.

lengths = []

for s in doc.sentences:

word\_count = sum(1 for w in s.words if w.upos != 'PUNCT')

if word\_count > 0:

lengths.append(word\_count)

return statistics.mean(lengths) if lengths else 0

def count\_subordinate\_clauses(sentence):

# I count subordinate clauses (dependent structures like advcl, ccomp, etc.)

# using the dependency relations assigned by Stanza.

# More subordination = higher syntactic embedding and complexity.

return sum(1 for w in sentence.words if w.deprel in ('advcl', 'ccomp', 'xcomp', 'acl', 'acl:relcl'))

def mean\_clause\_length(doc):

# I calculate the average number of words per clause.

# This gives another perspective on sentence density and syntactic load.

lengths = [len([w for w in s.words if w.upos != 'PUNCT'])

for s in doc.sentences if any(w.deprel != 'punct' for w in s.words)]

return statistics.mean(lengths) if lengths else 0

def mean\_dependency\_distance(sentence):

# For each dependency link in a sentence, I measure how far apart

# the dependent and its head are (in token positions).

#

# Greater average distance reflects more syntactic complexity and embedding.

distances = [abs(int(w.id) - int(w.head)) for w in sentence.words if w.head != 0]

return statistics.mean(distances) if distances else 0

def tree\_depth(sentence):

# I calculate the maximum depth of the dependency tree for a sentence,

# which captures how hierarchically nested the syntactic structure is.

# Deeper trees = more complex syntax.

#

# A dense sentence encodes more ideas, arguments, or modifiers in fewer sentences. For example:

# Simple: “She left.” (1 clause, 2 words → low density)

# Complex: “She left the meeting early because she felt sick and had forgotten her medicine.” (3 clauses, multiple modifiers → high density)

# This reflects the writer’s ability to manage multiple grammatical relationships within a single structure.

# Metrics that capture it: Mean Clause Length — more words per clause = higher density.

#

# Subordinate Clauses per Sentence — more embedded clauses = denser structure.

# Clause-to-Sentence Ratio — often used in complexity research (Lu, 2011).

#Syntactic load refers to how structurally and cognitively demanding a sentence is — the amount of grammatical “weight” carried by its syntax.

# More on Syntax load : In my analysis, I use mean dependency distance and tree depth as indicators of syntactic load.

#Longer dependency distances and deeper syntactic trees mean words are more spread out and hierarchically nested,

#which increases processing effort and grammatical complexity.For example:

#“The book that the teacher who won the award recommended was fascinating.” → high syntactic load (deep embedding).

#“The teacher recommended a fascinating book.” → low syntactic load.

#So, higher syntactic load reflects more advanced structural organization and greater linguistic control, but also greater cognitive demand.

#(Lu, 2011; Kyle & Crossley, 2018)

children = {int(w.id): [] for w in sentence.words}

for w in sentence.words:

hid = int(w.head); wid = int(w.id)

if hid != 0:

children[hid].append(wid)

def depth(node):

if not children.get(node):

return 1

return 1 + max(depth(c) for c in children[node])

roots = [int(w.id) for w in sentence.words if int(w.head) == 0]

return max(depth(r) for r in roots) if roots else 0

# Text similarity & uptake measures

def normalized\_edit\_distance(a, b):

# Measures how different two texts are by calculating Levenshtein edit distance

# (the number of single-character insertions, deletions, or substitutions required

# to turn one text into the other).

#

# I normalize the distance by the length of the longer text so the score

# ranges from 0 (identical) to 1 (completely different).

#

# I use this to quantify the magnitude of revision between drafts —

# higher values mean more substantial changes.

a\_s = (a or "").strip()

b\_s = (b or "").strip()

if not a\_s and not b\_s:

return 0.0

try:

d = Levenshtein.distance(a\_s, b\_s)

except Exception:

# fallback simple DP edit distance (character-level) if Levenshtein missing

def ed(x,y):

m = len(x); n = len(y)

dp = [[0]\*(n+1) for \_ in range(m+1)]

for i in range(m+1): dp[i][0] = i

for j in range(n+1): dp[0][j] = j

for i in range(1,m+1):

for j in range(1,n+1):

cost = 0 if x[i-1]==y[j-1] else 1

dp[i][j] = min(dp[i-1][j]+1, dp[i][j-1]+1, dp[i-1][j-1]+cost)

return dp[m][n]

d = ed(a\_s, b\_s)

return d / max(len(a\_s), len(b\_s), 1)

def ngrams\_tokens(s, n=3):

# Extracts all contiguous n-grams (default = 3 words) from a text.

# This captures local word-sequence patterns and helps measure

# how much phrasing is repeated or reused across texts.

toks = tokenize\_words(s)

return set(tuple(toks[i:i+n]) for i in range(len(toks)-n+1)) if len(toks) >= n else set()

#Chat gpt's first response to generate a corrected version for each participant in their interaction

gpt\_first = {

"participant1 GPT": """As the plane touched down, I expected to hear the usual “Welcome to [city name]” from the captain. Instead, his voice came through the speakers with an unsettling announcement: “We have arrived. Uhh… enjoy your stay.” No city name. No explanation. Just that. I grabbed my phone to check my location, but there was no signal—no WiFi, no bars, nothing. I looked out the window, expecting to see a familiar skyline or at least an airport with some recognizable signs. Instead, I saw a place that looked stuck in time. Stepping out of the airport, I tried asking people for help. “Excuse me, do you know where I can find a phone signal?” Blank stares. Some shook their heads. Others just walked past as if I didn’t exist. And the worst part? No internet meant no translator app. I tried gesturing with my phone, pointing to the sky like some lost tourist praying to the signal gods. Still nothing. Just as I was about to panic, I spotted a police officer. Finally! Someone official! I rushed over and asked for help. He squinted at me, struggling to understand, before finally managing in broken English: “Phone… no work here. You… need… wait.” Wait? Wait for what?! I didn’t care. The second I found a weak signal, I did the only logical thing—I booked the first ticket out. Mystery or not, this was one adventure I had no interest in continuing. As my new plane took off, I glanced out the window one last time at the strange city below. No name, no memory of it on any map, and no trace of it on my phone. For all I knew, it never existed at all.""",

"participant2 GPT": """Great start! You’ve captured the tension and confusion well. Now, let's refine it to make it even more engaging. A few things to consider: • Clarity & Flow: Some sentences could be broken down or restructured for better readability. • Grammar & Punctuation: Minor fixes, like capitalising "desert" and adjusting sentence structures. • Emphasising Fear: You could enhance the emotional impact by showing more physical reactions (e.g., heart pounding, sweaty palms). • Dialogue & Inner Thoughts: Maybe add a short internal monologue or quick exchanges with another passenger to make it more immersive. Would you like me to suggest a revised version, or do you want to tackle specific parts first?""",

"participant3 GPT": """I would initially look for a map of the area to guide myself. Since my phone wouldn’t have reception, I would probably look for a call center or a public phone to contact a friend or a parent. If everything went as planned, I would stay there and enjoy the trip; if not, I would try to buy a return ticket as soon as possible.""",

"participant4 GPT": """Here are some suggestions to improve your text: 1. Grammar & Clarity: Instead of "panic set in me," consider "panic set in" or "panic washed over me" for smoother phrasing. 2. Conciseness: Instead of "Seeking for some explanations," use "Seeking explanations" to make it more natural. 3. Flow & Readability: "Explained me" should be "explained to me." 4. Word Choice: "Thinking about that this unexpected delay could be an opportunity..." could be more fluid as "thinking that this unexpected delay might be an opportunity..." 5. Tension & Emotion: Consider expanding on your emotions—what exactly made you panic? Was it just the phone signal, or the unfamiliarity of the place? Adding a small sensory detail could enhance immersion. Your text already has a great structure! These refinements will make it smoother and more engaging.""",

"participant5 GPT": """If my plane lands in a different city, I will panic at first and ask another passenger whose phone works if I can use it to call my family and ask for advice. It also depends on where the plane lands—if it’s a beautiful place, I will ask my family to send me some money so I can explore the city. At the same time, I’ll try to buy a tourist SIM card so I can contact my family from my own phone and have internet access.""",

"participant6 GPT": """As I struggled to comprehend the situation, panic erupted around me. Passengers gasped, shouted, and clutched their phones—only to realize they had no signal. My pulse quickened. I unfastened my seatbelt in a rush, bolting from my seat and stepping into the aisle. I needed to see where we had landed. Pressing my face against the nearest window, my breath hitched. There was nothing but a vast expanse of dust and dirt stretching endlessly, as if we had been swept into this place by a storm. The eerie stillness outside sent a chill down my spine. I tried to focus on the captain’s announcement, but his voice was drowned out by the rising chaos. Heart pounding, I pushed my way through the narrow aisle, as if moving forward would somehow lead me to answers. I weaved through the panicked passengers, shoving past bodies in my desperate attempt to find clarity—until I slammed into someone. Before I could utter an apology, a voice stopped me cold. “I thought I’d never see you again…” My breath caught in my throat. Every muscle in my body tensed. That voice—familiar yet distant—froze me in place. Slowly, I forced myself to look up, locking eyes with the last person I ever expected to see. Damien Brody. My first crush. My high school sweetheart. And my greatest regret. The same boy who had shattered my heart by vanishing without a word, moving to another state as if I had never existed. He smirked—the same knowing, cocky smirk I had once fallen for. “Finding you,” he answered simply. Before I could react, he grabbed my arm and pulled me toward the emergency exit. My mind reeled, still trying to process his presence, let alone his intentions. And then—before I could resist—he kicked open the exit door and leaped out, dragging me with him. We hit the ground hard. A strange metallic scent filled the air. Dazed, I looked down. The soil beneath us wasn’t normal—it was red and glistening, like molten glass. It looked like we had just landed on Mars. Still in shock, I turned to Damien. “Where the hell are we?” He didn’t hesitate. “Area 81.” I let out a dry laugh. “Very funny. I’ve heard of Area 51, but there’s no such thing as Area 81.” “You’ve never heard of it because it didn’t exist—until last week.” My stomach dropped. Within minutes, the other passengers had been herded off the plane, wandering in a daze. A woman’s voice cut through the confusion. “Where are we? Why aren’t we at an airport?” Her words sent another wave of realization crashing over me. She was right. Planes didn’t just land in the middle of barren wastelands. “Stay close to me,” Damien whispered. A sudden rumbling sound filled the air. A moment later, armored figures emerged, their movements calculated, their weapons drawn. A deafening noise pierced the sky. Then—nothing. Darkness. I woke up nauseous, my limbs heavy, my head throbbing. My surroundings were a blur. Blinking against the harsh white lights, I struggled to move—only to realize I couldn’t. My wrists. My ankles. They were restrained. Panic surged through me. A pair of piercing blue eyes flickered into my vision—Damien. He was strapped to a table just a few feet away. His face was twisted with urgency as he turned toward me. “We were brought here as the chosen ones,” he whispered. A cold dread settled in my stomach. “What the hell are you talking about?” He took a slow breath. “Haven’t you ever wondered why I left so suddenly? Why I disappeared?” My throat tightened. I had spent years asking myself that question. “They took me,” he said. “I was with them.” Them. I swallowed hard. “Who are they?” His expression darkened. “The ones who know about us. About our genes.” My blood ran cold. My genes. Our genes. How did he—? No. No, it wasn’t possible. My heart pounded in my ears as a sickening realization washed over me. I had never told Damien. Never told him I was pregnant when he left. He couldn't know. He couldn’t possibly know. And yet… “I know, Rachel,” he murmured. His voice was barely a whisper, yet it shattered my world. Tears welled in my eyes. “You can’t take her from me,” I choked out. “You will never take her from me.” “Lower your voice,” he warned. “I’m trying to save our daughter.” I could barely breathe. “Our genes…” Damien continued. “We carry something rare. Something called Alpha-Botinol-S. It’s a mutation—normally dormant. But if activated, it can reprogram every single cell in the body.” He exhaled. “It makes you immortal.” My head spun. “The only way to naturally trigger this gene…” He hesitated. “Is by having two carriers produce an offspring.” My entire body went numb. “Lilly…” I gasped. Damien’s silence confirmed it. “No,” I whispered, shaking my head. “No. She’s just a normal child.” The door swung open before I could demand more answers. My breath caught in my throat. Two figures, dressed head to toe in white, stepped inside—dragging a small, trembling girl between them. My daughter. “NO!” I screamed. “You can’t take her! You CAN’T TAKE HER—” My own voice jolted me awake. Gasping, I clutched the armrests of my seat, my heart hammering against my ribs. A voice crackled through the intercom. “Dear passengers, please remain seated as we prepare to land in Octaone City.” Wait. Octaone City? I had never heard of that place before. Why did it feel like I had?""",

"participant7 GPT": """As the plane landed, I realized I had no idea where I was. My phone showed no signal, reminding me of the movie Lost. The plane had landed in the middle of nowhere. At first, I thought I was dreaming! Strangely, I couldn't even recall where this trip had started. Now, fully aware that I was awake in an unfamiliar city, I looked around. I didn’t recognize any of the passengers, and the flight attendants ignored my questions. I needed a plan. My first priority was to get off the plane and find the information desk to ask how I could contact my family and friends. Despite the strange situation, I decided to embrace it. Maybe this was a rare opportunity to explore a city I had never heard of before. After finally managing to reach my family, I felt calmer. Luckily, I received a visa on arrival, allowing me to step into the city. What I saw amazed me—it was a stunning blend of ancient and modern architecture. I wandered through the streets, using body language to ask locals for a good place to eat. Navigating the city without knowing its layout was challenging, but it felt like an adventure. I only had half a day before I needed to return to the airport, as the airline had booked me a ticket to my actual destination. The food was incredible! Every moment in this unexpected city felt like a gift. This experience taught me that I can enjoy unpredictable events too. I learned so much from this trip and cherished every second of it. Now, I have a great story to share with my friends!""",

"participant8 GPT": """Well, that’s something new—I didn’t see that coming. The plane just force-landed in some unknown place, right in the middle of nowhere, and now they’re asking everyone to leave. Brilliant idea! I’ve got to be honest—if it were up to me, I’d rather stay inside the plane, safe and sound. Because let’s be real—that’s usually when all the trouble starts: LEAVING THE SAFE SPOT! I don’t want to sound too dismissive, but we all know how this is going to go. Or should I say, how it went? Okay, bear with me for a few more lines—I’m not about to start over and do a recap for you. The rest? It’s the same old script from every Armageddon-style movie: A bunch of people trapped on some wicked, unknown island, struggling to survive through a series of apocalyptic events and action-packed sequences. One by one, they get murdered until we reach the final scene—the boy hugging his lover as she bleeds to death, while in the background, Kensington sings, "There, I’m all alone again..." The end.""",

"participant9 GPT": """If the airplane landed in a city I didn’t recognize, I would be very confused. I’d probably call over a flight attendant right away to ask where we were and why we landed there. With no signal on my phone, I’d start to worry a little—but I’d try to stay calm and figure out what’s going on. Maybe there was an emergency or a detour. I’ve never experienced anything like this before, so I’m not sure how I would react exactly, but I’d try to stay alert and look for answers.""",

"participant10 GPT": """The plane shuddered as it touched down. The captain’s voice came over the intercom, calm and unnerving: “Welcome to Tikaka.” Tikaka? I blinked at the words, at the passengers who, shockingly, seemed unfazed. Their faces were masks of serene expectation, like they had been waiting for this moment all their lives. My chest tightened. This wasn’t supposed to happen. I scrambled for my phone. No signal. Nothing. My thumbs hovered over the screen as if sheer will could summon a connection. “Excuse me,” I said to a nearby flight attendant, voice trembling. “Where are we, really?” She smiled—a smile too slow, too knowing—and simply said, “You’re home,” before gliding away like a ghost. Home? Panic clawed at me. I stumbled down the aisle, heart hammering, scanning faces for an ally. Everyone moved with quiet purpose, whispering words I didn’t understand. The world felt wrong. Outside, the city stretched endlessly, shimmering with an unnatural light. Streets curved impossibly, buildings twisted skyward as if reaching for some unseen force. Shadows moved of their own accord, following me, closing in, whispering my name in voices that weren’t my own. I ran. Doors slammed shut before I could reach them. Windows showed not sky, but gray voids. A whispering wind seemed to laugh at my desperation. My legs burned, my lungs ached, but there was no escape—Tikaka had swallowed me whole. Then, just as a shadow reached for my face, the world cracked. The city blurred, the passengers’ eyes stretched into black hollows, and the air pressed down like water. My heart froze. I screamed. And then—suddenly—I woke up. My room was quiet. Sunlight streamed through the curtains. The plane, Tikaka, the shadows—it had all been a dream. My hands shook as I reached for my phone. Signal bars flickered, solid and ordinary. But even as relief washed over me, a chill lingered. In the corner of my room, for just a heartbeat, a shadow shifted, and I could have sworn I heard a whisper: Welcome home.""",

"participant11 GPT": """I have always loved adventure and discovering new places. Nothing can stop me from enjoying the moment, even if the destination is not what I had planned. Since I purchased travel insurance, I am guaranteed to get my money back after returning to my city. So, why not enjoy this moment? From Africa to America, every city has its own beauty and can teach me new things. I can have a new experience and meet new people. The only thing that might bother me is the thought of landing in an unsafe place, which hopefully is not the case here. To recap, I will make the best out of every situation and won’t let it ruin my day. I will just contact my boyfriend and family after making sure I am safe, using someone else’s phone since mine has no signal.""",

"participant12 GPT": """Well, isn’t this just my kind of luck? The captain drones some city name that sounds like it was invented by a fantasy novelist—right as we’re touching down. And, of course, my phone decides this is its moment to die a tragic, silent death. No signal. Nothing. Zilch. So… welcome to my new life, apparently? For a second, I panic. Okay, maybe more than a second. But then I shrug and remind myself: weird places are the spice of life. Maybe this is my chance to go full Dora the Explorer—backpack, map, and all. Peeking out the window, the streets and buildings look like they belong in a parallel universe. Time travel vibes, but make it fashion. My heart races a little—not just from adrenaline, but from that irresistible itch to dive in. Who knows what bizarre snacks I’ll stumble upon? Or which strange, fascinating humans I’ll accidentally befriend? Step one: survive. Seems important. I’ll track down someone who speaks enough English to point me toward “civilization,” then fire off a quick “still alive” message to my family and my boyfriend. Step two: embrace the chaos. Snap photos, wander into alleys I shouldn’t, maybe get gloriously lost on purpose. Not the day I imagined, sure. But hey… adventure doesn’t care about plans. I’m rolling with it. And honestly? I might just fall in love with this accidental detour."""

}

# Running analyses: compute metrics for each participant × text,

# plus uptake of ChatGPT feedback into revisions (text2 and text3).

results\_all = {}

print("\nRunning analyses on texts with embedded ChatGPT feedback...")

# Define count\_automated\_errors before using it

def count\_automated\_errors(text):

# Prefer LanguageTool; I use LanguageTool because it’s an open-source grammar

# and style checker that automatically identifies grammar, spelling, and punctuation errors in text.

#It’s built on a large rule-based and statistical system (Naber, 2003) and is reliable for surface-level

# accuracy in learner writing — especially for things like: subject–verb agreement article misuse misspellings extra or missing punctuation

#incorrect capitalization, etc.

if has\_languagetool and tool is not None:

try:

matches = tool.check(text)

num = len(matches)

# count number of flagged issues

tokens = tokenize\_words(text)

#e rrors per 100 words

per100 = num / (len(tokens)/100) if tokens else num

# count by rule type

types = Counter([m.ruleId for m in matches])

return {"num\_suggestions": num, "per\_100\_words": per100, "by\_rule": dict(types)}

except Exception as e:

# falback to heuristic if LanguageTool crashes

print("LanguageTool failed on text; using heuristic. Error:", e)

# FALLBACK HEURISTIC (if LanguageTool not available) ---

# I estimate surface errors using simple signals of potential mistakes:

# 1) Very rare words (zipf < 2.5) → possible typos or misspellings

# 2) Repeated punctuation sequences ("!!", "...", ",,,") → style/grammar issues

# 3) Letter duplication ("soooo", "noooo") → informal spelling or error

tokens = tokenize\_words(text)

if not tokens:

return {"num\_suggestions": 0, "per\_100\_words": 0.0, "by\_rule": {}}

rare = []

try:

rare = [w for w in tokens if zipf\_frequency(w,'en') < 2.5]

except Exception:

rare = []

rp = len(re.findall(r'([!?.,]{2,})', text))

weird = len([w for w in tokens if re.search(r'(.)\1{3,}', w)])

num = len(rare) + rp + weird

per100 = num / (len(tokens)/100) if tokens else 0

return {"num\_suggestions": int(num), "per\_100\_words": per100, "by\_rule": {"rare\_tokens\_est": len(rare), "repeated\_punct": rp, "weird\_dup": weird}}

# Running analyses

# This is the main loop that runs every metric I’ve defined earlier

# (syntactic, lexical, and error-based) on every participant × text.

# For each participant, I go through their three texts and extract

# linguistic measures using Stanza and my helper functions.

results\_all = {}

print("\nRunning analyses on texts...")

for pid, writings in texts.items():

results\_all[pid] = {}

for tid, raw\_text in writings.items():

raw\_text = (raw\_text or "").strip()

if not raw\_text:

results\_all[pid][tid] = {

"raw\_text": "", "sentence\_count":0, "mean\_sentence\_length":0, "subordinate\_clauses":0,

"avg\_clause\_length":0, "pos\_counts":{}, "dependency\_distance":0, "tree\_depth":0,

"tokens":0,"types":0,"ttr":0,"cttr":0,"hapax\_rate":0,"mtld":0,"avg\_zipf":None,

"lexical\_sophistication":None,"lexical\_density":None,"langtool":{"num\_suggestions":0,"per\_100\_words":0.0}

}

continue

# 1) parsing with Stanza (if available)

# I create a parsed document so I can access sentences, words, and dependency relations.

if stanza\_ready and nlp is not None:

try:

doc = nlp(raw\_text)

except Exception as e:

print(f"Stanza parsing failed for participant {pid} text {tid}; error:", e)

doc = None

else:

doc = None

# 2) syntactic metrics

if doc:

syntactic\_sentence\_count = len(doc.sentences)

mean\_sent\_len = mean\_sentence\_length(doc)

total\_subordinate = sum(count\_subordinate\_clauses(s) for s in doc.sentences)

avg\_clause\_len = mean\_clause\_length(doc)

pos\_count = Counter([w.upos for s in doc.sentences for w in s.words])

dep\_distances = [mean\_dependency\_distance(s) for s in doc.sentences if len(s.words)>0]

dep\_dist = statistics.mean(dep\_distances) if dep\_distances else 0

tree\_depths = [tree\_depth(s) for s in doc.sentences if len(s.words)>0]

tree\_d = max(tree\_depths) if tree\_depths else 0

else:

syntactic\_sentence\_count = 0

mean\_sent\_len = 0

total\_subordinate = 0

avg\_clause\_len = 0

pos\_count = Counter()

dep\_dist = 0

tree\_d = 0

# 3) for lexical metrics

tokens = tokenize\_words(raw\_text)

t\_types, t\_tokens = types\_tokens(tokens)

lex\_ttr = ttr(tokens); lex\_cttr = cttr(tokens); lex\_hapax = hapax\_rate(tokens)

lex\_mtld = mtld(tokens); lex\_avg\_zipf = avg\_zipf\_freq(tokens); lex\_soph = lexical\_sophistication(tokens)

# lexical density = proportion of content words (NOUN, VERB, ADJ, ADV, PROPN)

lex\_density = None

if pos\_count:

content = sum(pos\_count.get(tag,0) for tag in ('NOUN','VERB','ADJ','ADV','PROPN'))

total\_pos = sum(pos\_count.values()) - pos\_count.get('PUNCT',0)

lex\_density = content/total\_pos if total\_pos else None

# 4) Accuracy metrics

# I use LanguageTool (or heuristic fallback) to estimate grammatical error rates.

lt\_err = count\_automated\_errors(raw\_text)

results\_all[pid][tid] = {

"raw\_text": raw\_text, "sentence\_count": syntactic\_sentence\_count,

"mean\_sentence\_length": mean\_sent\_len, "subordinate\_clauses": total\_subordinate,

"avg\_clause\_length": avg\_clause\_len, "pos\_counts": dict(pos\_count),

"dependency\_distance": dep\_dist, "tree\_depth": tree\_d,

"tokens": t\_tokens, "types": t\_types, "ttr": lex\_ttr, "cttr": lex\_cttr,

"hapax\_rate": lex\_hapax, "mtld": lex\_mtld, "avg\_zipf": lex\_avg\_zipf,

"lexical\_sophistication": lex\_soph, "lexical\_density": lex\_density,

"langtool": lt\_err

}

print("Text-level analyses complete.")

# Participant-level edit distances and deltas, I want to see how much each participant’s texts change between versions.

# I want to see how much each participant’s texts change between drafts.

# This tells me about the extent of revisionn(surface-level change)

# and also about developmental improvement(lexical and syntactic gains).

# I use two main kinds of comparisons here:

# 1) Normalized Edit Distance — measures textual change.

# 2) Delta Metrics (MTLD, Mean Sentence Length) — measure linguistic improvement.

for pid, writings in results\_all.items():

a = writings.get("1", {}).get("raw\_text", "")

b = writings.get("2", {}).get("raw\_text", "")

c = writings.get("3", {}).get("raw\_text", "")

# Textual Revision Magnitude (Levenshtein edit distance)

# I calculate normalized edit distance between drafts:

# - edit\_1\_2 → how much the text changed from version 1 → 2

# - edit\_2\_3 → how much it changed from version 2 → 3

# A higher score = more rewriting, a lower score = more similarity.

results\_all[pid]["edit\_1\_2"] = normalized\_edit\_distance(a,b) if (a is not None and b is not None) else None

results\_all[pid]["edit\_2\_3"] = normalized\_edit\_distance(b,c) if (b is not None and c is not None) else None

# Lexical Diversity Change (MTLD)

# I check if lexical diversity (MTLD) increased between drafts.

# Positive change = richer vocabulary use.

try:

m1 = results\_all[pid]["1"]["mtld"]; m2 = results\_all[pid]["2"]["mtld"]

results\_all[pid]["mtld\_change\_1\_2"] = (m2 - m1) if (m1 is not None and m2 is not None) else None

except Exception:

results\_all[pid]["mtld\_change\_1\_2"] = None

# Syntactic Complexity Change (Mean Sentence Length)

# I also track whether sentences got longer on average between drafts,

# which may signal higher syntactic elaboration or denser structure.

try:

s1 = results\_all[pid]["1"]["mean\_sentence\_length"]; s2 = results\_all[pid]["2"]["mean\_sentence\_length"]

results\_all[pid]["mean\_sent\_len\_change\_1\_2"] = (s2 - s1) if (s1 is not None and s2 is not None) else None

except Exception:

results\_all[pid]["mean\_sent\_len\_change\_1\_2"] = None

# Save JSON, After calculating all lexical, syntactic, and error metrics for every participant and every text,

# I want to save those results permanently so I can reuse or analyze them later.

# I keep two output formats:

# JSON: the full nested dictionary structure (best for programmatic reuse)

# Excel: a “long” flat table (best for manual inspection and statistical analysis)

json\_path = OUT\_DIR / "results\_all.json"

json\_path.write\_text(json.dumps(results\_all, ensure\_ascii=False, indent=2), encoding="utf8")

print("Saved results JSON to", json\_path)

# Build long dataframe and save Excel

# I flatten the nested results (participant → text → metrics)

# into a single table where each row = one participant × one text.

# This is the standard “tidy data” format for analysis in Pandas, R, or SPSS.

rows=[]

for pid, texts\_metrics in results\_all.items():

pid\_num = int(re.search(r'(\d+)', str(pid)).group(1)) if re.search(r'(\d+)', str(pid)) else pid

for text\_id in ['1','2','3']:

m = texts\_metrics.get(text\_id,{})

row = {"participant": pid\_num, "text": int(text\_id)}

row["raw\_text"] = m.get("raw\_text","")

for col in ["sentence\_count","mean\_sentence\_length","subordinate\_clauses","avg\_clause\_length","dependency\_distance","tree\_depth","tokens","types","ttr","cttr","hapax\_rate","mtld","avg\_zipf","lexical\_sophistication","lexical\_density"]:

row[col]=m.get(col,None)

lt = m.get("langtool", None)

row["langtool\_num\_suggestions"] = (lt.get("num\_suggestions") if lt else None)

row["langtool\_per\_100w"] = (lt.get("per\_100\_words") if lt else None)

pos = m.get("pos\_counts", {})

for tag,cnt in pos.items(): row[f"pos\_{tag}"]=cnt

rows.append(row)

df\_long = pd.DataFrame(rows).fillna(0).sort\_values(["participant","text"]).reset\_index(drop=True)

df\_long['participant'] = df\_long['participant'].astype(int); df\_long['text'] = df\_long['text'].astype(int)

long\_xlsx = OUT\_DIR / "syntax\_metrics\_long.xlsx"

with pd.ExcelWriter(long\_xlsx, engine="openpyxl") as writer:

df\_long.to\_excel(writer, index=False, sheet\_name="metrics\_long")

raw\_sheet = pd.DataFrame([{"participant": int(re.search(r'(\d+)', p).group(1)), "text": int(t), "raw\_text": texts[p][t]} for p in texts for t in texts[p]])

raw\_sheet.to\_excel(writer, index=False, sheet\_name="raw\_texts")

print("Saved long xlsx to", long\_xlsx)

# Build wide table

# I'm converting the per-participant × per-text nested results into a single wide table.

# For each metric (e.g., mean\_sentence\_length, mtld) I create columns like mean\_sentence\_length\_1, mean\_sentence\_length\_2, mean\_sentence\_length\_3.

# This makes it easy for me to compute participant-level deltas, run between-subject models, or export a single-row summary per participant.

# Then I save the wide table to Excel and CSV for sharing or manual inspection.

main\_metrics = ["mean\_sentence\_length","subordinate\_clauses","avg\_clause\_length","dependency\_distance","tree\_depth","mtld","ttr","hapax\_rate","langtool\_num\_suggestions"]

wide\_frames = []

for metric in main\_metrics:

if metric in df\_long.columns:

tmp = df\_long.pivot\_table(index="participant", columns="text", values=metric)

if tmp.shape[1]>0:

tmp.columns = [f"{metric}\_{int(c)}" for c in tmp.columns]

wide\_frames.append(tmp)

if wide\_frames:

df\_wide = pd.concat(wide\_frames, axis=1).reset\_index().sort\_values("participant")

wide\_path = OUT\_DIR / "syntax\_metrics\_wide.xlsx"

df\_wide.to\_excel(wide\_path, index=False)

print("Saved wide xlsx to", wide\_path)

else:

print("No wide metrics created (check columns).")

# PCA:

# In this step, I create a single "syntactic complexity composite score"

# using Principal Component Analysis (PCA).

# WHY I DO THIS:

# Each syntactic metric (mean sentence length, subordinate clauses, clause length,

# dependency distance, tree depth) measures a slightly different aspect of

# syntactic complexity, but they are often correlated.

#

# Rather than interpret five separate measures, I combine them into one index

# that summarizes overall syntactic elaboration per text.

# - It takes all standardized syntactic features and finds the main dimension

# (the first principal component, PC1) that explains the largest amount of variance.

# - This PC1 score serves as a weighted combination of all those features.

# For example, texts with long sentences, more subordination, and deeper trees

# will get higher PC1 (more complex syntax).

pca\_cols = [c for c in ["mean\_sentence\_length","subordinate\_clauses","avg\_clause\_length","dependency\_distance","tree\_depth"] if c in df\_long.columns]

if pca\_cols:

scaler = StandardScaler(); X = df\_long[pca\_cols].fillna(0)

Xs = scaler.fit\_transform(X)

pca = PCA(n\_components=1)

df\_long["syntax\_pc1"] = pca.fit\_transform(Xs).flatten()

loadings = dict(zip(pca\_cols, pca.components\_[0].tolist()))

print("PC1 loadings:", loadings)

# Within-subject Friedman on mean\_sentence\_length

# The Friedman test is a non-parametric alternative to repeated-measures ANOVA.

# It doesn’t assume normal distributions and works well for small samples.

# Each participant contributes three related measures — one per text version.

# The test checks if there’s a consistent change in mean\_sentence\_length

# across the three conditions.

# If the Friedman test is significant, I run \*pairwise Wilcoxon signed-rank tests\*

# to pinpoint where the difference lies (e.g., between Text 1–2, 1–3, or 2–3).

#

# Then I apply a Bonferroni correction to control for multiple comparisons.

def within\_subject\_test(metric="mean\_sentence\_length"):

if metric not in df\_long.columns:

print("Metric missing:", metric); return None

vals={1:[],2:[],3:[]}; included=[]

for pid,g in df\_long.groupby("participant"):

s=g.set\_index("text")

if all(t in s.index for t in (1,2,3)):

vals[1].append(float(s.loc[1,metric])); vals[2].append(float(s.loc[2,metric])); vals[3].append(float(s.loc[3,metric]))

included.append(pid)

if len(vals[1])<2:

print("Not enough complete participants."); return None

stat,p = friedmanchisquare(vals[1], vals[2], vals[3])

print(f"Friedman {metric}: chi2={stat:.4f}, p={p:.6g}, n={len(vals[1])}")

pairs=[(1,2),(1,3),(2,3)]; pvals=[]

for a,b in pairs:

try: pv = wilcoxon(vals[a], vals[b]).pvalue

except Exception: pv = float('nan')

pvals.append(pv)

nonnan=[pv for pv in pvals if not (isinstance(pv,float) and np.isnan(pv))]

if nonnan:

rej,p\_adj,\*\_ = multipletests(nonnan, method="bonferroni")

print("Pairwise raw p-values:", pvals)

print("Bonferroni adj (non-NaNs):", p\_adj)

return {"friedman\_p": p, "pairwise\_raw": pvals, "participants": included}

print("\n--- Within-subject test (mean\_sentence\_length) ---")

within\_subject\_test("mean\_sentence\_length")

# Between-subject example (ANOVA/Kruskal)

# Here I compare participants against each other — not across their drafts,

# but as independent individuals — to see if there are meaningful differences

# in their overall syntactic complexity.

# - If the data roughly meet normality assumptions, I use a \*one-way ANOVA\*

# (Analysis of Variance) to test for differences among participants.

# - If the data are non-normal or I have few samples, I use the

# \*Kruskal–Wallis H test\*, which is the non-parametric equivalent.

# - Both tests check if at least one participant differs significantly

# from the others in the given metric.

if "mean\_sentence\_length" in df\_long.columns:

grouped = df\_long.groupby("participant")["mean\_sentence\_length"].apply(list)

lists = [lst for lst in grouped if len(lst)>0]

if len(lists) >= 2:

try:

fstat,pval = f\_oneway(\*lists); print(f"One-way ANOVA mean\_sentence\_length: F={fstat:.4f}, p={pval:.6g}")

except Exception:

kstat,kp = kruskal(\*lists); print(f"Kruskal-Wallis mean\_sentence\_length: H={kstat:.4f}, p={kp:.6g}")

# Plots: saved to PNG\_DIR

# After running all my lexical, syntactic, and accuracy analyses,

# I visualize the results using Seaborn and Matplotlib.

# Each plot helps me see overall trends, variation, and group-level patterns

# that are harder to interpret from tables alone.

#

# I save every figure as a PNG file into my plots folder (PNG\_DIR)

# so I can easily include them in reports or papers.

sns.set(style="whitegrid", rc={"figure.figsize":(8,5)})

if "mean\_sentence\_length" in df\_long.columns:

plt.figure(); sns.boxplot(data=df\_long, x="text", y="mean\_sentence\_length"); sns.stripplot(data=df\_long,x="text",y="mean\_sentence\_length",color="k",size=4,alpha=0.6,jitter=0.2)

plt.title("Mean sentence length by Text (all participants)")

plt.savefig(PNG\_DIR / "box\_mean\_sentence\_length\_by\_text.png", dpi=200, bbox\_inches="tight"); plt.close(); print("Saved plot mean\_sentence\_length")

if "mtld" in df\_long.columns:

plt.figure(); sns.boxplot(data=df\_long, x="text", y="mtld"); plt.title("MTLD by Text"); plt.savefig(PNG\_DIR / "box\_mtld\_by\_text.png", dpi=200, bbox\_inches="tight"); plt.close(); print("Saved mtld plot")

if "syntax\_pc1" in df\_long.columns:

plt.figure(); sns.boxplot(data=df\_long, x="text", y="syntax\_pc1"); plt.title("Syntax composite (PC1) by Text"); plt.savefig(PNG\_DIR / "box\_syntax\_pc1\_by\_text.png", dpi=200, bbox\_inches="tight"); plt.close(); print("Saved syntax\_pc1 plot")

if "langtool\_num\_suggestions" in df\_long.columns:

plt.figure(); sns.boxplot(data=df\_long, x="text", y="langtool\_num\_suggestions"); plt.title("Grammar suggestions by Text"); plt.savefig(PNG\_DIR / "box\_langtool\_suggestions\_by\_text.png", dpi=200, bbox\_inches="tight"); plt.close(); print("Saved langtool plot")

# Correlation heatmap of averaged metrics

metrics\_for\_wide = [m for m in ["mean\_sentence\_length","mtld","ttr","hapax\_rate","subordinate\_clauses","avg\_clause\_length","dependency\_distance","tree\_depth","langtool\_num\_suggestions"] if m in df\_long.columns]

if metrics\_for\_wide:

df\_feat = df\_long.groupby("participant")[metrics\_for\_wide].mean()

plt.figure(figsize=(9,7)); sns.heatmap(df\_feat.corr(), annot=True, cmap="vlag", center=0); plt.title("Correlation of averaged metrics"); plt.savefig(PNG\_DIR / "metric\_correlations.png", dpi=200, bbox\_inches="tight"); plt.close(); print("Saved correlations heatmap")

# Change summary 1->2 and save

# I take the mean of each metric per participant (across their three texts),

# then I create a correlation matrix and visualize it as a heatmap.

# I want to see whether improvements or strengths in one area

summary=[]

for pid,g in df\_long.groupby("participant"):

r=g.set\_index("text"); out={"participant": pid}

if 1 in r.index and 2 in r.index:

out["mtld\_change\_1\_2"] = float(r.loc[2,"mtld"] - r.loc[1,"mtld"]) if ("mtld" in r.columns) else None

out["mean\_sent\_len\_change\_1\_2"] = float(r.loc[2,"mean\_sentence\_length"] - r.loc[1,"mean\_sentence\_length"]) if ("mean\_sentence\_length" in r.columns) else None

out["langtool\_change\_1\_2"] = float(r.loc[2,"langtool\_num\_suggestions"] - r.loc[1,"langtool\_num\_suggestions"]) if ("langtool\_num\_suggestions" in r.columns) else None

summary.append(out)

pd.DataFrame(summary).to\_excel(OUT\_DIR / "change\_summary\_1\_2.xlsx", index=False)

print("Saved change summary.")

print("\nAll outputs saved to:", OUT\_DIR)

print("I measured basic lexical diversity (TTR, MTLD, hapax), syntactic complexity (mean sentence length, clause counts, dependency distance, tree depth), and automated error counts (LanguageTool if available or heuristic). I do this because:")

print("- Lexical measures tell me about vocabulary richness and rare-word use.")

print("- Syntactic measures give signals of sentence complexity and planning.")

print("- Edit distances show how much participants changed between drafts.")

print("- Error counts indicate form/grammar issues that might correlate with proficiency or revision quality.")

print("\nIf you'd like a README.txt describing the files, or the script broken into nicer Colab cells, say which and I'll add it.")

# ADD GPT-CORRECTED TEXTS, PLOT + COMPARISONS (single cell)

# To seperately analyze the participants whose 2nd answer is missing

import math, os, json, statistics, warnings

from pathlib import Path

import pandas as pd, numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from scipy.stats import friedmanchisquare, wilcoxon, f\_oneway, kruskal

from statsmodels.stats.multitest import multipletests

from collections import Counter

warnings.filterwarnings("ignore")

sns.set(style="whitegrid")

fig\_dpi = 200

# Config

OUT\_DIR = Path(OUT\_DIR) if 'OUT\_DIR' in globals() else Path.cwd() / "syntax\_results"

PNG\_DIR = Path(PNG\_DIR) if 'PNG\_DIR' in globals() else OUT\_DIR / "plots"

PNG\_DIR.mkdir(parents=True, exist\_ok=True)

# JSON helper to avoid TypeError: int64 not serializable

def make\_json\_serializable(obj):

"""Recursively convert numpy/pandas objects into json-serializable Python builtins."""

import math

from datetime import date, datetime

if obj is None:

return None

if isinstance(obj, (str, bool)):

return obj

if isinstance(obj, (np.integer,)):

return int(obj)

if isinstance(obj, (np.floating,)):

if math.isnan(obj) or math.isinf(obj):

return None

return float(obj)

if isinstance(obj, (np.bool\_,)):

return bool(obj)

if isinstance(obj, (np.ndarray,)):

return [make\_json\_serializable(v) for v in obj.tolist()]

if isinstance(obj, (pd.Timestamp, datetime)):

return obj.isoformat()

if isinstance(obj, date):

return obj.isoformat()

if isinstance(obj, (pd.Series,)):

return make\_json\_serializable(obj.tolist())

if isinstance(obj, (pd.DataFrame,)):

return make\_json\_serializable(obj.to\_dict(orient='records'))

if isinstance(obj, dict):

return {str(k): make\_json\_serializable(v) for k,v in obj.items()}

if isinstance(obj, (list, tuple, set)):

return [make\_json\_serializable(v) for v in obj]

# try item/tolist fallback

try:

if hasattr(obj, "item"):

return make\_json\_serializable(obj.item())

if hasattr(obj, "tolist"):

return make\_json\_serializable(obj.tolist())

except Exception:

pass

# last resort

return str(obj)

# Loading df\_long if missing

if 'df\_long' not in globals():

xlsx\_path = OUT\_DIR / "syntax\_metrics\_long.xlsx"

if xlsx\_path.exists():

df\_long = pd.read\_excel(xlsx\_path, sheet\_name="metrics\_long")

print("Loaded df\_long from", xlsx\_path)

else:

raise RuntimeError("df\_long not found in memory and no syntax\_metrics\_long.xlsx in OUT\_DIR")

# Ensure types

df\_long['participant'] = df\_long['participant'].astype(int)

df\_long['text'] = df\_long['text'].astype(int)

# Helper to compute metrics for a raw text (uses your earlier functions)

def compute\_metrics\_for\_text(raw\_text):

raw\_text = (raw\_text or "").strip()

out = {}

out["raw\_text"] = raw\_text

if not raw\_text:

# return NaNs for numeric fields

numeric\_keys = ["sentence\_count","mean\_sentence\_length","subordinate\_clauses","avg\_clause\_length",

"dependency\_distance","tree\_depth","tokens","types","ttr","cttr","hapax\_rate",

"mtld","avg\_zipf","lexical\_sophistication","lexical\_density"]

for k in numeric\_keys:

out[k] = np.nan

out["langtool"] = {"num\_suggestions":0,"per\_100\_words":0.0,"by\_rule":{}}

return out

# stanza parse if available

doc = None

if 'stanza\_ready' in globals() and stanza\_ready and 'nlp' in globals() and nlp is not None:

try:

doc = nlp(raw\_text)

except Exception as e:

print("Warning: stanza parse failed:", e)

doc = None

if doc:

sentence\_count = len(doc.sentences)

mean\_sent\_len = mean\_sentence\_length(doc)

total\_subordinate = sum(count\_subordinate\_clauses(s) for s in doc.sentences)

avg\_clause\_len = mean\_clause\_length(doc)

pos\_count = Counter([w.upos for s in doc.sentences for w in s.words])

dep\_distances = [mean\_dependency\_distance(s) for s in doc.sentences if len(s.words)>0]

dep\_dist = statistics.mean(dep\_distances) if dep\_distances else np.nan

tree\_depths = [tree\_depth(s) for s in doc.sentences if len(s.words)>0]

tree\_d = max(tree\_depths) if tree\_depths else np.nan

else:

sentence\_count = np.nan; mean\_sent\_len = np.nan; total\_subordinate = np.nan

avg\_clause\_len = np.nan; pos\_count = Counter(); dep\_dist = np.nan; tree\_d = np.nan

tokens = tokenize\_words(raw\_text)

t\_types, t\_tokens = types\_tokens(tokens)

lex\_ttr = ttr(tokens); lex\_cttr = cttr(tokens); lex\_hapax = hapax\_rate(tokens)

lex\_mtld = mtld(tokens)

try:

lex\_avg\_zipf = avg\_zipf\_freq(tokens)

except Exception:

lex\_avg\_zipf = np.nan

try:

lex\_soph = lexical\_sophistication(tokens)

except Exception:

lex\_soph = np.nan

lex\_density = np.nan

if pos\_count:

content = sum(pos\_count.get(tag,0) for tag in ('NOUN','VERB','ADJ','ADV','PROPN'))

total\_pos = sum(pos\_count.values()) - pos\_count.get('PUNCT',0)

lex\_density = content/total\_pos if total\_pos else np.nan

lt = {"num\_suggestions":0,"per\_100\_words":0.0,"by\_rule":{}}

if 'count\_automated\_errors' in globals():

try:

lt = count\_automated\_errors(raw\_text)

except Exception:

lt = {"num\_suggestions":0,"per\_100\_words":0.0,"by\_rule":{}}

out.update({

"sentence\_count": sentence\_count, "mean\_sentence\_length": mean\_sent\_len, "subordinate\_clauses": total\_subordinate,

"avg\_clause\_length": avg\_clause\_len, "pos\_counts": dict(pos\_count),

"dependency\_distance": dep\_dist, "tree\_depth": tree\_d,

"tokens": t\_tokens, "types": t\_types, "ttr": lex\_ttr, "cttr": lex\_cttr,

"hapax\_rate": lex\_hapax, "mtld": lex\_mtld, "avg\_zipf": lex\_avg\_zipf,

"lexical\_sophistication": lex\_soph, "lexical\_density": lex\_density,

"langtool": lt

})

return out

# Building GPT mapping (participant number -> GPT text)

gpt\_map = {}

if 'gpt\_first' in globals():

import re

for k,v in gpt\_first.items():

m = re.search(r'(\d+)', k)

if m:

pid = int(m.group(1))

gpt\_map[pid] = v

else:

print("No gpt\_first found; GPT texts won't be included.")

# Building extended df (texts 1, GPT as 2, original 2 -> 3, original 3 -> 4)

participants = sorted(df\_long['participant'].unique())

df\_long\_base = df\_long.copy()

new\_rows = []

for pid in participants:

# trying to fetch raw texts from 'texts' dict if available

raw1 = raw2 = raw3 = ""

key = f"participant{pid}"

if 'texts' in globals() and isinstance(texts, dict) and key in texts:

raw1 = texts[key].get("1","") or ""

raw2 = texts[key].get("2","") or ""

raw3 = texts[key].get("3","") or ""

# fallback to df\_long\_base raw\_text

if not raw1:

t1 = df\_long\_base[(df\_long\_base.participant==pid) & (df\_long\_base.text==1)]

raw1 = t1["raw\_text"].iloc[0] if not t1.empty else ""

if not raw2:

t2 = df\_long\_base[(df\_long\_base.participant==pid) & (df\_long\_base.text==2)]

raw2 = t2["raw\_text"].iloc[0] if not t2.empty else ""

if not raw3:

t3 = df\_long\_base[(df\_long\_base.participant==pid) & (df\_long\_base.text==3)]

raw3 = t3["raw\_text"].iloc[0] if not t3.empty else ""

gpt\_text = gpt\_map.get(pid, "")

m1 = compute\_metrics\_for\_text(raw1)

mg = compute\_metrics\_for\_text(gpt\_text)

m2 = compute\_metrics\_for\_text(raw2)

m3 = compute\_metrics\_for\_text(raw3)

def mkrow(pid, version, mdict):

lt = mdict.get("langtool", {"num\_suggestions":0,"per\_100\_words":0.0})

return {

"participant": int(pid),

"text": int(version),

"raw\_text": mdict.get("raw\_text",""),

"sentence\_count": mdict.get("sentence\_count", np.nan),

"mean\_sentence\_length": mdict.get("mean\_sentence\_length", np.nan),

"subordinate\_clauses": mdict.get("subordinate\_clauses", np.nan),

"avg\_clause\_length": mdict.get("avg\_clause\_length", np.nan),

"dependency\_distance": mdict.get("dependency\_distance", np.nan),

"tree\_depth": mdict.get("tree\_depth", np.nan),

"tokens": mdict.get("tokens", np.nan),

"types": mdict.get("types", np.nan),

"ttr": mdict.get("ttr", np.nan),

"cttr": mdict.get("cttr", np.nan),

"hapax\_rate": mdict.get("hapax\_rate", np.nan),

"mtld": mdict.get("mtld", np.nan),

"avg\_zipf": mdict.get("avg\_zipf", np.nan),

"lexical\_sophistication": mdict.get("lexical\_sophistication", np.nan),

"lexical\_density": mdict.get("lexical\_density", np.nan),

"langtool\_num\_suggestions": lt.get("num\_suggestions", np.nan),

"langtool\_per\_100w": lt.get("per\_100\_words", np.nan)

}

new\_rows.append(mkrow(pid, 1, m1))

new\_rows.append(mkrow(pid, 2, mg))

new\_rows.append(mkrow(pid, 3, m2))

new\_rows.append(mkrow(pid, 4, m3))

df\_long\_ext = pd.DataFrame(new\_rows)

df\_long\_ext['participant'] = df\_long\_ext['participant'].astype(int)

df\_long\_ext['text'] = df\_long\_ext['text'].astype(int)

# Save extended sheet

xlsx\_ext = OUT\_DIR / "syntax\_metrics\_long\_withGPT.xlsx"

with pd.ExcelWriter(xlsx\_ext, engine="openpyxl") as writer:

df\_long\_ext.to\_excel(writer, sheet\_name="metrics\_long\_withGPT", index=False)

print("Saved extended metrics with GPT as text-2 to", xlsx\_ext)

# Plot grouped bars: participants on x, bars = texts

numeric\_cols = df\_long\_ext.select\_dtypes(include=[np.number]).columns.tolist()

numeric\_cols = [c for c in numeric\_cols if c not in ("participant","text")]

preferred = ["tokens","types","ttr","cttr","mtld","hapax\_rate","avg\_zipf","lexical\_sophistication","lexical\_density",

"sentence\_count","mean\_sentence\_length","subordinate\_clauses","avg\_clause\_length","dependency\_distance","tree\_depth",

"langtool\_num\_suggestions","langtool\_per\_100w"]

metrics = [m for m in preferred if m in numeric\_cols] + [m for m in numeric\_cols if m not in preferred]

participants = sorted(df\_long\_ext['participant'].unique())

texts\_found = sorted(df\_long\_ext['text'].unique())

n\_texts = len(texts\_found)

def savefig(fig, name):

path = PNG\_DIR / name

fig.savefig(path, dpi=fig\_dpi, bbox\_inches="tight")

plt.close(fig)

print("Saved:", name)

# Cleaning up impossible negatives for metrics that should be non-negative

nonneg = {"ttr","mtld","hapax\_rate","mean\_sentence\_length","tokens","types","subordinate\_clauses","avg\_clause\_length","dependency\_distance","tree\_depth","langtool\_num\_suggestions","langtool\_per\_100w"}

for m in metrics:

if m in df\_long\_ext.columns and m in nonneg:

bad = df\_long\_ext[m] < 0

if bad.any():

print(f"Warning: negative values in {m}; setting to NaN for plotting.")

df\_long\_ext.loc[bad, m] = np.nan

# --- plotting with pastel palette + smart vertical labels when overlapping ---

# Use seaborn's pastel palette (you can replace with a list of hex colors if desired)

palette = sns.color\_palette("pastel", n\_colors=max(3, n\_texts))

for metric in metrics:

if df\_long\_ext[metric].dropna().empty:

print("Skipping", metric, "— all NaN")

continue

pivot = df\_long\_ext.pivot\_table(index='participant', columns='text', values=metric, aggfunc='mean')

pivot = pivot.reindex(participants)

x = np.arange(len(pivot.index))

total\_width = 0.8

width = total\_width / max(1, n\_texts)

fig, ax = plt.subplots(figsize=(max(8, len(participants)\*0.6), 5))

bars = []

for i, t in enumerate(texts\_found):

vals = pivot.get(t, pd.Series(index=pivot.index, data=np.nan))

bar = ax.bar(x + (i - (n\_texts-1)/2)\*width, vals.values,

width=width\*0.95, label=f"Text {t}",

color=palette[i % len(palette)], edgecolor='black', linewidth=0.3)

bars.append(bar)

ax.set\_xticks(x)

ax.set\_xticklabels([f"P{p}" for p in pivot.index], rotation=45, ha='right')

ax.set\_ylabel(metric)

ax.set\_title(f"{metric} — grouped bars per participant (texts = {texts\_found})")

ax.legend(title="Text")

ax.grid(axis='y', linestyle='--', alpha=0.5)

# draw to get a renderer (needed to measure distances in pixels)

fig.canvas.draw()

renderer = fig.canvas.get\_renderer()

# collect bar centers in display (pixel) coordinates so we can detect overlap

centers\_disp = []

for rect in ax.patches:

cx = rect.get\_x() + rect.get\_width()/2

cx\_disp, \_ = ax.transData.transform((cx, 0))

centers\_disp.append(cx\_disp)

# compute minimal center distance (in pixels)

min\_center\_dist = np.inf

if len(centers\_disp) >= 2:

centers\_disp\_sorted = np.sort(np.array(centers\_disp))

diffs = np.diff(centers\_disp\_sorted)

if len(diffs) > 0:

min\_center\_dist = diffs.min()

# estimate typical text width (in pixels) for a formatted number like "1234.56"

test\_txt = f"{np.nanmax(df\_long\_ext[metric].fillna(0)):.2f}"

txt = ax.text(0, 0, test\_txt, fontsize=7) # temporary text to measure

bb = txt.get\_window\_extent(renderer=renderer)

est\_text\_width = bb.width

txt.remove()

# decide whether to rotate labels vertically: if center spacing is smaller than text width \* ~1.1 -> rotate

rotate\_vertical = (min\_center\_dist < est\_text\_width \* 1.1)

# annotate bars — if rotating, place text vertically above bar, else standard horizontal

for rect in ax.patches:

h = rect.get\_height()

if not np.isnan(h) and abs(h) > 1e-8:

cx = rect.get\_x() + rect.get\_width()/2

cy = h

label = f"{h:.2f}"

if rotate\_vertical:

# place vertical text; rotate 90 degrees; offset slightly above the bar

ax.annotate(label,

xy=(cx, cy),

xytext=(0, 3), textcoords="offset points",

ha='center', va='bottom',

fontsize=7, rotation=90)

else:

# default horizontal small labels

ax.annotate(label,

(cx, cy),

ha='center', va='bottom', fontsize=7)

savefig(fig, f"groupedbar\_allparticipants\_{metric}.png")

# Within-subject Friedman + pairwise tests

within\_results = []

for metric in metrics:

# collecting participants who have non-NaN values for all texts\_found

complete\_participants = []

for pid, group in df\_long\_ext.groupby("participant"):

s = group.set\_index("text")

if all(t in s.index and pd.notna(s.loc[t, metric]) for t in texts\_found):

complete\_participants.append(pid)

# Building matrices for friedman: each element is a vector of participant values per text

if len(complete\_participants) >= 2:

matrices = []

for t in texts\_found:

col = [float(df\_long\_ext[(df\_long\_ext.participant==pid) & (df\_long\_ext.text==t)][metric].values[0]) for pid in complete\_participants]

matrices.append(col)

try:

stat,p = friedmanchisquare(\*matrices)

except Exception as e:

stat,p = np.nan, np.nan

# pairwising wilcoxon (adjust for multiple comparisons)

pairs = []

pvals = []

for i in range(len(texts\_found)):

for j in range(i+1, len(texts\_found)):

a = np.array(matrices[i]); b = np.array(matrices[j])

try:

pv = wilcoxon(a,b).pvalue

except Exception:

pv = np.nan

pairs.append((texts\_found[i], texts\_found[j])); pvals.append(pv)

pvals\_nonan = [pv for pv in pvals if not (isinstance(pv,float) and np.isnan(pv))]

if pvals\_nonan:

rej, adj\_pvals, \_, \_ = multipletests(pvals\_nonan, method="bonferroni")

# map adjusted back into full list

adj = []

it = iter(adj\_pvals)

for pv in pvals:

if not (isinstance(pv,float) and np.isnan(pv)):

adj.append(next(it))

else:

adj.append(np.nan)

else:

adj = [np.nan]\*len(pvals)

within\_results.append({

"metric": metric, "friedman\_stat": float(stat) if not np.isnan(stat) else None, "friedman\_p": float(p) if not np.isnan(p) else None,

"pairwise\_pairs": pairs, "pairwise\_p\_raw": pvals, "pairwise\_p\_adj": adj,

"n\_participants\_complete": len(complete\_participants)

})

else:

within\_results.append({

"metric": metric, "friedman\_stat": None, "friedman\_p": None,

"pairwise\_pairs": [], "pairwise\_p\_raw": [], "pairwise\_p\_adj": [], "n\_participants\_complete": len(complete\_participants)

})

# Saving within-subject results (JSON safe)

with open(OUT\_DIR / "within\_subject\_friedman\_results.json", "w", encoding="utf8") as fh:

json.dump(make\_json\_serializable(within\_results), fh, indent=2, ensure\_ascii=False)

print("Saved within-subject Friedman + pairwise results to JSON.")

# Between-subject tests (ANOVA or Kruskal on participant groups)

between\_results = []

for metric in metrics:

if metric not in df\_long\_ext.columns:

continue

# lists per participant

groups = [group[metric].dropna().values for \_, group in df\_long\_ext.groupby("participant") if len(group[metric].dropna())>0]

if len(groups) >= 2 and all(len(g)>0 for g in groups):

try:

fstat, p = f\_oneway(\*groups)

between\_results.append({"metric": metric, "test": "ANOVA", "F": float(fstat), "p": float(p)})

except Exception:

try:

kstat, kp = kruskal(\*groups)

between\_results.append({"metric": metric, "test": "Kruskal", "H": float(kstat), "p": float(kp)})

except Exception as e:

between\_results.append({"metric": metric, "error": str(e)})

else:

between\_results.append({"metric": metric, "test": "insufficient\_data"})

with open(OUT\_DIR / "between\_subject\_results.json", "w", encoding="utf8") as fh:

json.dump(make\_json\_serializable(between\_results), fh, indent=2, ensure\_ascii=False)

print("Saved between-subject results to JSON.")

# Per-participant deltas / percent changes

delta\_rows = []

for pid, group in df\_long\_ext.groupby("participant"):

s = group.set\_index("text")

row = {"participant": int(pid)}

for metric in metrics:

vals = {t: (float(s.loc[t, metric]) if (t in s.index and pd.notna(s.loc[t, metric])) else np.nan) for t in texts\_found}

def pdiff(a,b):

if np.isnan(a) or np.isnan(b): return np.nan

return b - a

def pcent(a,b):

if np.isnan(a) or np.isnan(b) or a == 0: return np.nan

return (b-a)/abs(a)\*100.0

# fill values for up to 4 texts (1..4)

for t in texts\_found:

row[f"{metric}\_val\_{t}"] = vals.get(t, np.nan)

# deltas consecutive

for i in range(len(texts\_found)-1):

a\_t = texts\_found[i]; b\_t = texts\_found[i+1]

row[f"{metric}\_delta\_{a\_t}\_{b\_t}"] = pdiff(vals.get(a\_t, np.nan), vals.get(b\_t, np.nan))

row[f"{metric}\_pct\_{a\_t}\_{b\_t}"] = pcent(vals.get(a\_t, np.nan), vals.get(b\_t, np.nan))

delta\_rows.append(row)

df\_deltas = pd.DataFrame(delta\_rows)

deltas\_path = OUT\_DIR / "per\_participant\_metric\_deltas.csv"

df\_deltas.to\_csv(deltas\_path, index=False)

print("Saved per-participant deltas to", deltas\_path.name)

# Highlights: largest absolute changes per metric

highlights = []

for metric in metrics:

cols = [c for c in df\_deltas.columns if c.startswith(f"{metric}\_delta\_")]

out = {"metric": metric}

for c in cols:

series = df\_deltas[c].abs().dropna()

if not series.empty:

top\_idx = series.idxmax()

out[c + "\_max"] = float(series.max())

out[c + "\_participant"] = int(df\_deltas.loc[top\_idx,"participant"])

out[c + "\_signed"] = float(df\_deltas.loc[top\_idx, c])

else:

out[c + "\_max"] = np.nan; out[c + "\_participant"] = np.nan; out[c + "\_signed"] = np.nan

highlights.append(out)

pd.DataFrame(highlights).to\_csv(OUT\_DIR / "delta\_highlights.csv", index=False)

print("Saved delta highlights to delta\_highlights.csv")

# Save supporting tables

df\_long\_ext.to\_csv(OUT\_DIR / "syntax\_metrics\_long\_withGPT.csv", index=False)

pd.DataFrame(make\_json\_serializable(within\_results)).to\_csv(OUT\_DIR / "friedman\_summary.csv", index=False)

pd.DataFrame(make\_json\_serializable(between\_results)).to\_csv(OUT\_DIR / "between\_subject\_summary.csv", index=False)

print("Saved data CSVs and summaries.")

print("\nALL DONE. Outputs saved to:", OUT\_DIR)

print("- Grouped bar PNGs ->", PNG\_DIR)

print("- per-participant deltas ->", deltas\_path)

print("- Friedman JSON ->", OUT\_DIR / "within\_subject\_friedman\_results.json")

print("- Between-subject JSON ->", OUT\_DIR / "between\_subject\_results.json")

print("\nNext options I can add (tell me which):")

print(" - single PDF report concatenating all PNGs")

print(" - arrow annotations/highlights on grouped bars where deltas exceed a threshold")

print(" - a formatted significance table (LaTeX/Markdown) of pairwise tests")