Prompts used for Generating dataset

Initial Prompt for GPT to Generate Data Points

Generate 10 structured medical case reports in JSON format. Each report should follow the structure below:

```
{
    "sentence": "Generated sentence",
    "spans": [
        {"label": "LABEL_NAME", "text": "Extracted entity text"},
        {"label": "LABEL_NAME", "text": "Extracted entity text"}
    ]
}
```

Labels and their descriptions:

- PATIENT: Patient's name (e.g., John Doe)
- STAFF: Medical staff name (e.g., Dr. Emily Carter, Nurse Olivia Green)
- AGE: Patient's age (e.g., 45-year-old)
- BIRTHDATE: Patient's birthdate (e.g., March 12, 1978)
- DATE: Any date (admission, discharge, etc.) (e.g., June 20, 2023)
- PHONE: Contact number (e.g., +1-415-832-5678)
- ID: Unique identifier (e.g., PAT-564321, SURG-890123)
- EMAIL: Email address (e.g., emily.carter@greenfieldhosp.org)
- LOC: Location details (e.g., Colombo, Sri Lanka, New York, USA)
- HOSP: Hospital or medical institution (e.g., Greenfield Hospital, Apollo Medical Center)
- ORG: Organizations or institutions (e.g., World Health Organization, American Heart Association)
- URL: Web links to medical or research-related content (e.g., https://www.medicalupdates.com)
- TIMESTAMPS: Relative or exact timestamps (e.g., yesterday, three months ago, last Friday)

Special instructions:

1. Each case must be unique and should contain a mix of the above labels.
2. Ensure different starting points for each sentence (e.g., some start with the hospital, others with the patient, doctor, or timestamps).
3. Include real-world scenarios such as emergency cases, routine check-ups, surgeries, research studies, and public health reports.

4. Each case must contain at least two URLs related to medical research, hospital sites, or health news.
5. The dataset should have variations in sentence structure and order of information to ensure diversity.

Generate the output as a JSON array with 10 structured medical cases.

General Reminder to Ensure Variation in Sentence Structure

Before generating the dataset, follow these instructions carefully to avoid repetition:

1. Vary sentence structure and order of information
  - Do not follow the same format in every sentence.
  - Some sentences should begin with the hospital name, others with the doctor's name, date, patient details, or event description.
  - Change the way dates and timestamps are introduced (e.g., "Yesterday at 9 AM" vs. "On March 10, 2024").
  - Use a mix of short, concise statements and longer, detailed reports.

2. Use different sentence styles
  - Some should sound like formal hospital reports.
  - Some should be research summaries or patient narratives.
  - Others can resemble news articles or public health statements.

Generate 10 unique medical cases, ensuring no two follow the same writing pattern.

Preventing GPT from Missing Tags in Sentences

Each generated medical case must contain a sufficient number of entity labels.

1. Every case must include at least 8-12 labeled entities from the following:
  - PATIENT, STAFF, AGE, BIRTHDATE, DATE, PHONE, ID, EMAIL, LOC, HOSP, ORG, URL, TIMESTAMPS
2. No case should miss important categories (e.g., every case should have at least one DATE, one LOC, and one STAFF or HOSP).
3. Ensure correct placement of information so that all labels appear naturally within the sentence.
4. Do not generate incomplete sentences -all labels must be covered correctly.

Generate 10 structured cases and make sure all necessary labels are properly included.

Forcing GPT to Use Unique Names for Patients, Doctors, and Hospitals

To avoid repetition of names, each case must use different names for patients, doctors, and hospitals.

1. Use diverse patient names, ensuring no name is repeated.
2. Each case must introduce a different medical staff member (doctor, nurse, or surgeon).
3. Use different hospital names and locations (mixing well-known hospitals and fictional ones).
4. Include realistic global variation—patients and doctors should come from different countries and cities.

Example name variations:

- Patients: Daniel White, Olivia Brown, Aamir Hassan, Sophia Lin, Juan Rodriguez, Naomi Ishikawa
- Doctors: Dr. Henry Collins, Dr. Anjali Desai, Dr. Lee Cheng, Dr. Priya Patel, Dr. Samuel Okafor
- Hospitals: Apollo Medical Center, Colombo General Hospital, Mayo Clinic, Royal Health Institute, Sunrise Medical Center

Generate 10 structured cases, ensuring complete diversity in names.

Covering All Labels with Sufficient Data Points

Before generating the dataset, ensure that every label has a sufficient number of data points across all cases.

1. Each case should contain a mix of at least 8-12 labels.
2. Ensure different labels appear across cases, not just in a few cases.
   - Some cases should highlight AGE and BIRTHDATE, while others focus on ID and PHONE.
   - Some should include EMAIL and ORG, while others emphasize HOSP and LOC.
3. At the end of the dataset, all labels must have been used multiple times.

Generate 10 unique cases, ensuring balanced distribution of entity labels.

Covering the Maximum Possible Medical Scenarios

To ensure a diverse dataset, generate cases covering a wide range of medical scenarios:

1. Include different types of cases:

- Emergency treatments (e.g., heart attack, stroke)
- Routine check-ups (e.g., annual medical exams, vaccinations)
- Surgeries and procedures (e.g., knee replacement, bypass surgery)
- Hospital admissions & discharge (e.g., COVID-19 recovery, post-surgery care)
- Medical research & clinical trials (e.g., experimental cancer treatment)
- Public health announcements (e.g., flu season warning, vaccine rollout)
- Patient recovery stories (e.g., rehabilitation after an accident)

2. Each case must introduce a unique medical scenario, avoiding redundancy.

Generate 10 structured cases, ensuring that they represent different real-world situations.

Covering the Maximum Possible Medical Scenarios

To ensure a diverse dataset, generate cases covering a wide range of medical scenarios:

1. Include different types of cases:
- Emergency treatments (e.g., heart attack, stroke)
- Routine check-ups (e.g., annual medical exams, vaccinations)
- Surgeries and procedures (e.g., knee replacement, bypass surgery)
- Hospital admissions & discharge (e.g., COVID-19 recovery, post-surgery care)
- Medical research & clinical trials (e.g., experimental cancer treatment)
- Public health announcements (e.g., flu season warning, vaccine rollout)
- Patient recovery stories (e.g., rehabilitation after an accident)

2. Each case must introduce a unique medical scenario, avoiding redundancy.

Generate 10 structured cases, ensuring that they represent different real-world situations.

Example Combined Reminder Prompt

Before generating the dataset, follow these important rules:

1. Vary sentence structure—do not repeat the same format.
2. Ensure each case includes at least 8-12 entity labels.
3. Use different names for patients, doctors, and hospitals.

4. Cover all labels across the dataset—every entity type must appear multiple times.
5. Ensure diverse medical cases (e.g., emergencies, check-ups, surgeries, research).
6. Each case must include at least two URLs related to medical research or hospital sites.
7. Mix timestamps (both exact dates and relative times like "last Friday").

Now, generate 10 structured medical cases in JSON format.

Ensuring Variation in Phone Number Formats

Each generated case must contain a different phone number format to reflect global diversity. Do not use the same format repeatedly.

1. Include different country codes (e.g., +1 for the USA, +44 for the UK, +91 for India, +94 for Sri Lanka).
2. Use a mix of spacing, dashes, and parentheses:
   - "+1 415 832 5678"
   - "+44-7789-123456"
   - "(212) 555-0198"
   - "011-269-9999"
   - "+94 77 234 5678"
3. Ensure a mix of hospital, doctor, and emergency contact numbers.
4. Include both mobile and landline formats.

Generate 10 cases ensuring different phone number formats in each case.

Generating Diverse Email Formats

Each generated case must contain a different email structure. Avoid repeating the same format.

1. Vary domains (e.g., hospital, research institute, government, personal emails):
   - "dr.smith@apollohosp.org"
   - "contact@who.int"
   - "michael.stevens@medilab.com"
   - "info@healthcare.gov"
2. Use different structures:
   - FirstName.LastName (e.g., "john.doe@citymed.com")
   - Initials with domain (e.g., "j.smith@royalhospital.net")

- Numbers in email (e.g., "drlee1980@globalhealth.org")
- Hospital departments (e.g., "cardiology@greenfieldhosp.org")

3. Include different types of emails (doctor, hospital, support, emergency contact).

Generate 10 cases ensuring different email formats across the dataset.

Ensuring Variation in Medical IDs

Each generated case must contain unique and diverse medical IDs.

1. Use different ID formats based on context:
   - Patient IDs: "PAT-564321", "PX-987654"
   - Medical Report IDs: "MED-20240516", "LAB-098765"
   - Surgery IDs: "SURG-456789", "PROC-102938"
   - National ID variations: "NIC-983456789V", "SSN-123-45-6789"
2. Use different numbering styles:
   - Prefixes: "PAT-", "MED-", "SURG-", "LAB-", "EMR-"
   - Numeric-only IDs: "764382910"
   - Alphanumeric IDs: "RX-45A2B67"
   - Year-based IDs: "LAB-2024-567890"
3. Ensure variety across patients, hospitals, and reports.

Generate 10 cases, ensuring different ID structures in each.

Covering All Date Formats

Each generated case must contain a different date format to ensure variety.

1. Use different numeric formats:
   - "07/14/2023" (MM/DD/YYYY)
   - "14/07/2023" (DD/MM/YYYY)
   - "2023/07/14" (YYYY/MM/DD)
   - "07-14-2023" (MM-DD-YYYY)
   - "2023-07-14" (YYYY-MM-DD)
2. Include named month formats:
   - "July 14, 2023"
   - "14 July 2023"

- "2023 July 14"
3. Use ordinal dates:
- "July 14th, 2023"
- "14th of July, 2023"
4. Ensure time-stamped entries:
- "2023-07-14 14:30:00"
- "April 5, 2023, at 10:30 AM"

Generate 10 cases, ensuring different date formats in each case.

Covering All Birthdate Formats

Each generated case must contain a unique and diverse birthdate format.

1. Use different common formats:
- "March 12, 1978"
- "12/03/1978"
- "1978-03-12"
- "03-12-1978"
- "12 March 1978"
2. Include variations in century representation:
- "07/07/1980"
- "December 20, 1985"
- "January 1, 1975"
3. Include a mix of numeric and written formats.
4. Ensure birthdates are realistic (between 1920-2024).

Generate 10 cases, ensuring different birthdate formats in each case.

Covering All Relative Dates (TIMESTAMPS)

Each generated case must contain a unique and varied timestamp or relative date format.

1. Use different relative date formats:
- "three months ago"
- "last Friday"
- "six weeks ago"

- "a year ago"
- "yesterday at 8:30 AM"
- "last summer"

2. Ensure variety in how time references appear:
- "On May 3, 2024, at 14:45"
- "Two weeks before the surgery"
- "On the evening of March 1st"
- "Earlier this year"

Generate 10 cases, ensuring different timestamp formats in each case.

Challenges Encountered in Dataset Generation and Their Solutions

1. GPT is Not Accurate in Generating Start and End Indexes for Labels

Problem: GPT often fails to correctly determine the start and end indexes of entity labels in generated sentences.

Solution:

- Instead of relying on GPT for indexes, extract the labeled text from the generated output and compute the start and end indexes using a Python script.
- Use string matching techniques (e.g., str.find()) to locate entities within sentences after generation.

2. GPT Tends to Follow the Same Structure When Generating Sentences

Problem: GPT-generated sentences often follow a repetitive pattern, reducing dataset diversity.

Solution:

- **Use variation-enforcing instructions**: Direct GPT to rearrange sentence structures dynamically.
- **Use multiple perspectives**: Ask GPT to generate text from different viewpoints (e.g., doctor, patient, hospital, researcher).
- **Generate multiple outputs at once**: Request 5-10 outputs per query to ensure diversity.
- **Force random order**: Instruct GPT to shuffle information order rather than following a fixed template.

3. GPT Misses Some Tags in Sentences

Problem: Certain entity labels may not appear in some sentences, causing missing data points.

Solution:

- **Manual verification**: Ensure every generated entry includes a predefined set of required labels.
- **Reinforce mandatory labels in the prompt**: Explicitly instruct GPT to include specific entities in every output.
- **Post-process missing labels**: Run a script to check if expected labels are present, and regenerate missing data points.

4. GPT Tends to Reuse the Same Names for Patients, Doctors, and Hospitals

Problem: Limited variety in entity names reduces dataset realism.

Solution:

- **Provide a diverse set of names**: Use predefined lists of unique names for each category (patients, doctors, hospitals, etc.).
- **Randomize name selection**: Implement a script to replace repeated names with alternatives from a name pool.
- **Introduce cultural diversity**: Ensure names represent different regions and demographics.

5. Covering All Labels with Sufficient Data Points

Problem: Some entity types appear too frequently, while others are underrepresented.

Solution:

- **Track entity distribution**: Maintain a counter to ensure all labels appear in balanced proportions.
- **Generate different sentence types**: Structure prompts to emphasize different labels in different cases.
- **Set category-specific label requirements**: Ensure prompts for different scenarios focus on relevant entities.

6. Covering Maximum Possible Scenarios

Problem: The dataset may lack coverage of real-world variations and edge cases.

Solution:

- **Define a broad range of scenarios**: Create a predefined list of case types (e.g., emergency admissions, lab results, surgery reports, research papers, etc.).
- **Ensure scenario balance**: Generate an equal number of cases for each type.
- **Randomize generation prompts**: Modify instructions to force unique case studies.

7. Generating Large Datasets Efficiently

Problem: Manually generating a large dataset (e.g., 2000+ entries) is time-consuming and prone to errors.

Solution:

- **Batch generation**: Request GPT to generate 20-50 entries per prompt execution.
- **Automate post-processing**: Use scripts to validate, clean, and structure generated data.
- **Use distributed generation**: Run multiple independent queries with varied instructions to ensure a diverse dataset.
- **Implement quality control checks**: Verify entity distribution, remove duplicates, and validate label coverage programmatically.

Conclusion

By implementing these solutions, we can significantly improve the diversity, accuracy, and completeness of the generated dataset while ensuring it remains scalable for large-scale projects.