PharmaSUG 2026 - Paper AI-157

Tips and Considerations for Preparing Health Data for Efficient and Accurate AI and LLM Modelling

Louise S. Hadden, Cormac Corporation

ABSTRACT

Large language models (LLMs) and AI systems present new opportunities for pharmaceutical research and clinical insight generation from data derived from electronic medical record systems (EMRs). However, health-related data—especially data streams with open-ended narratives and diverse coding systems—requires rigorous preparation for use in compliant, accurate, and efficient AI workflows. This paper outlines a practical framework for preparing transcoded EMR data for AI models and LLM use within health analytics pipelines. Topics include data cleaning, normalization, de-identification, prompt engineering, and iterative refinement cycles. Three use cases are explored: (1) detecting behavioral health issues from free-text 'other specify' fields, categorizing adverse patient safety events from free-text ‘other specify’ fields, and (3) linking disparate medical coding systems (SNOMED, ICD-10, Common Formats, FHIR/HL7, etc.) Implementations using SAS, AWS AI tools, and open source software are discussed.

INTRODUCTION

The pharmaceutical industry increasingly leverages AI and natural language processing (NLP) to extract clinical insights from disparate data sources provided by sponsors, etc. Free-text narratives, clinician notes, and 'other specify' fields contain nuanced signals that structured data often omits. Preparing such data for large language and AI models requires a rigorous, auditable pipeline combining structured and unstructured data processing across multiple environments.

USE CASES

Our system is cloud-based (AWS), and a combination of SAS, SQL, Python, and AWS tools such as Comprehend Medical are being used. Tableau and Quarto are used for reporting. The current data stream has millions of records and hundreds of variables (stored in Oracle) so scalability is important. Due to highly confidential data, specific examples will not be presented.

USE CASE 1

The Agency for Healthcare Research and Quality (AHRQ) is a U.S. federal agency within the Department of Health and Human Services (HHS) focused on improving the safety, quality, and effectiveness of healthcare. It produces evidence-based research, tools, and data to help patients, clinicians, and policymakers make informed decisions. The AHRQ Patient Safety Organization Privacy Protection Center (PSOPPC) is a program created by the Agency for Healthcare Research and Quality (AHRQ) to help PSOs collect, secure, and analyze patient safety data. PSOPPC standardizes data on adverse patient safety events received from PSOs via AHRQ Common Formats to identify trends while protecting confidentiality. These data are collected as a series of data elements related to a number of discrete event types (i.e., injurious falls, pressure injuries, etc.). Events that aren’t classified into a specific type are coded as “Other”. There are several very large open text fields that are used to collect information about these “Other” events, and these fields are being run through LLM processing in the attempt to classify them. However, text fields are notoriously messy, and these fields are full of PHI/PII and cannot be analyzed prior to rigorous cleaning and deidentification. Additionally, some data

elements are stored in coding systems such as SNOMED CT and ICD-10, providing challenges to harmonizing information.

USE CASE 2

A specific search / classification task was to collect information on Behavioral Health, which is not currently defined in the Common Formats. Manual inspection of the data collected above found the presence of multiple behavioral health indicators in the open-ended EMR fields,

USE CASE 3

AHRQ's Common Formats for Event Reporting (CFER) are a set of standardized definitions and answer codes that streamline the collection of data from disparate PSOs, specifically targeting patient safety events. Answer codes may include other coding systems, as well as open text fields. CFER is used in different environments (Hospitals, Nursing Homes, Community Pharmacies, and Diagnostic Safety), and has different versions (V1.2 and V2.0) and updates. An effort was undertaken to associate discrete data elements to elements from other health coding systems (FHIR, USCDI, USCDI+, HL7, etc.).

In an effort to safely and securely perform categorization on this data, we follow the following steps.

DATA PREPARATION AND CLEANING

Health-related data contains inconsistencies, typographical errors, and protected health information (PHI). A multi-step data cleaning pipeline ensures compliance and analytical quality.

If you send raw, identifiable free text straight into an LLM, you are taking on avoidable risk. The safest pattern on AWS is to minimize -> de-identify -> validate -> then model. We attempt a practical, production-oriented approach tailored to adverse event narratives.

Define scope and risk upfront – before you touch the data:

- Classify what you have (PHI/PII, clinical notes, event narratives)
- Map obligations (e.g., HIPAA, contractual terms)
- Decide the minimum necessary fields for categorization (often just the narrative and a few non-identifying attributes – note that it can be handy to have a sequence number to match to if you want to pull in additional data later)
- Make sure to store originating and resulting data securely (e.g., Amazon S3 with SSE-KMS)

Much of our PSOPPC data is collected via transcription from EMR data. EMR data often contain system-generated or keystroke-triggered placeholder entries such as 'AMA', 'Injury', or 'Other'. These artifacts arise when electronic health record systems auto-populate fields based on partial input or when users skip optional fields. Such entries can cause difficulties for natural language models, inflating noise and false-positive rates. Therefore, pre-cleaning should explicitly screen for and remove or reclassify these placeholders before downstream processing.

Key steps include:

- Data extraction and profiling in SAS and SQL to detect missingness, duplication, and outliers.
- Text normalization: case folding, spelling correction using clinical dictionaries, and abbreviation expansion. For example:
    - Fix encoding, normalize whitespace, remove control characters
    - Standardize dates/units ("03/01/24"->ISO, expand common abbreviations

    - Remove boilerplate templates and EMR autopopulated junk (e.g., “Other/AMA/Unknown” placeholders (making sure these aren’t valid answers)
    - Basic spell-correction for common medical terms
    - Build a dictionary of clinical synonyms (e.g. “fall”, “slip”,”found on floor”) to standardize before modeling
    - Watch for embedded identifiers in narratives – these are common misses
    - Remove auto-populated EMR phrases early – they add noise and can carry identifiers
    - You can do this in AWS Glue or Amazon EMR(Spark)
    - Encoding normalization and sentence segmentation for long narratives.
- De-identification using AWS Comprehend Medical and Python regex-based masking routines.
    - Use automated PHI/PII detection plus rule-based passes:
        - Automated detection (Amazon Comprehend Medical for clinical PHI, Amazon Comprehend for general PII)
        - Rule-based / regex layer (complements ML)
    - Transformations (replace with tokens i.e., [PATIENT NAME], generalize rather than drop i.e. exact date - > month/year or a relative time)
- IMPORTANT: Validate de-identification quality!
    - Sampling + human review of redacted text (privacy officer / clinical reviewer)
    - Measure recall of PHI detection (misses are the main risk)
    - Add “canary” tests (seed known PHI patterns and ensure they are removed)
    - Iterate rules where misses occur (e.g. unusual facility names, nicknames)
    - If you are dealing with rare conditions, rare events, small geographies, perform secondary risk reduction such as k-anonymity style checks, suppress/generalize outliers
- Audit trail creation in SAS metadata tables to document every transformation.

SPECIAL CONSIDERATIONS FOR USE CASE 3

Disparate medical terminologies—such as SNOMED CT, ICD-10, AHRQ Common Formats, and FHIR/HL7—pose challenges for data harmonization. SAS and SQL can be used to store and index mapping tables, while Python scripts perform contextual disambiguation and version management.

Recommended mapping process:

- Normalize and version all incoming codes using SAS PROC FORMAT and custom mapping tables.
- Apply deterministic mappings where possible, flagging ambiguous cases for manual review.
- Employ Python-based rule engines to handle probabilistic mappings and context-sensitive conversions.
- Store mapping provenance (source, rule, confidence) in a unified AWS RDS schema.

PROMPT ENGINEERING FOR HEALTH TEXT ANALYSIS

Effective prompt design ensures accurate and interpretable LLM outputs. In SAS-integrated AI workflows, prompts can be generated dynamically based on patient data subsets. Optimal prompts include context, output schema, and examples while remaining concise.

Example prompt template (behavioral health detection):

Input: "Patient reports difficulty sleeping and frequent sadness."

Expected Output: { 'behavioral_health_flag': 'yes', 'categories': ['depression', 'insomnia'], 'confidence': 0.91 }

ITERATIVE MODEL REFINEMENT

The refinement cycle is a cornerstone of responsible AI deployment in health analytics. Each iteration improves model precision and reduces hallucination risk through human-in-the-loop validation. SAS can store and compare model metrics, while Python handles retraining and evaluation automation. With our models, we have had many interations and refinements of prompts, with as many validation cycles.

Cycle steps:

1. Generate and test initial prompts.
2. Review model outputs with subject matter experts.
3. Alter prompts and pre-processing scripts to correct observed weaknesses.
4. Re-test and record metrics until convergence thresholds are reached.

VALIDATION SAMPLING

Data resulting from our LLM runs was sampled for validation using SAS PROC SURVEYSELECT. SURVEYSELECT provides the option for a serpentine sort, which maximizes the chance of getting at least one of every record type. Our cleaned, de-identified files contain a minimal number of fields – a sequence number, binaries for each of 18 event types, weights for each present event type, and the cleaned, deidentified text field. The sample was broken into 4 separate files for 4 separate reviewers, each with 200 records (we used colors). Each reviewer was assigned different event types, and once reviews were complete, reviewers met to go over together, and make recommendations for re-engineered prompts, etc.

Figure 1: PROC SURVEYSELECT Code

```
*****************************************************************************;
*** Use PROC SURVEYSELECT with a Serpentine Sort to Output Samples      ***;
*** 4 replicates of 200 for 800 total                                   ***;
*** Purpose is for Review of Classification                             ***;
*****************************************************************************;

%macro runcolor(color=green,cvars=binwt1 binwt2 binwt3 binwt4 binwt5);

proc surveyselect data=output_&color
    out=Class_&color._200
    method=seq          /* Sequential sampling */
    sampsize=200        /* Total sample size of 800 */
    /*reps=4               Divide into 4 groups/replicates */
    seed=12345;
    control PSO_ID &cvars;  /* Serpentine sort variables */
run;

%mend;

%runcolor(color=green,cvars=binwt1 binwt2 binwt3 binwt4 binwt5);
%runcolor(color=purple,cvars=binwt6 binwt7 binwt8 binwt16);
%runcolor(color=red,cvars=binwt11 binwt12 binwt13 binwt15);
%runcolor(color=blue,cvars=binwt9 binwt10 binwt14 binwt17);
```

It's very important that this review be done in a secured space: although we'd used AWS Comprehend Medical and Python tools to redact our data fields, we found that we needed to run through the tools repeatedly because consecutive names (i.e. first and last, Dr. Last) were not always caught. Word to the wise!

EVALUATION AND GOVERNANCE

Evaluation metrics include precision, recall, F1-score, and inter-annotator agreement for labeled datasets. Governance frameworks must ensure reproducibility, traceability, and ethical oversight. AWS security controls and SAS metadata management provide the necessary audit layers. As a result of our LLM models, we are able to recommend new event types and revised instructions for completing PSOPPC submissions.

CONCLUSION

Integrating SAS, SQL, AWS tools, and Python enables an efficient, auditable pipeline for preparing our PSOPPC data for large language models. Through robust cleaning, thoughtful prompt design, and iterative refinement, data scientists can responsibly leverage LLMs for insights into adverse patient safety events and coding systems without compromising data security concerns.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Louise S. Hadden
louisesquibbhadden@gmail.com