HiLabs Hackathon: Smart Contract Language Tagging for Negotiation Assist

Organized By: HiLabs in collaboration with IIT Kharagpur

Problem Statement: Intelligent Language Classification for Healthcare Contracts

Introduction

Healthcare provider contracts define the payment terms, rules, and responsibilities between:

- **Insurance companies (payers)**, and
- **Healthcare providers** like hospitals, clinics, and physicians.

Even small differences in contract language can lead to significant financial, compliance, and operational impacts, including:

- Overpayment or underpayment for services
- Provider dissatisfaction or disputes
- Network disruptions (providers leaving the network)
- Legal and regulatory issues due to unclear or incorrect terms

Traditionally, reviewing these contracts is manual and time-consuming, requiring legal, actuarial, and network teams to read through each document line-by-line, identify important clauses, and determine if the language aligns with organizational policies.

HiLabs' **Negotiation Assist** platform aims to automate and simplify this process using AI-powered analytics, making healthcare contract negotiations:

- **Faster** – reducing manual effort
- **Smarter** – improving accuracy with automated checks
- **Consistent** – applying the same rules every time

This hackathon challenges participants to build a beginner-friendly version of this capability. Your solution will automatically read healthcare contracts, extract important clauses, and classify them, so teams can quickly review and compare contracts.

The Challenge

You will build a system that processes healthcare contracts and classifies each clause into one of two categories:

- **Standard** – Language that perfectly matches the organization's preferred wording
- **Non-Standard** – Language that is different, risky, or requires attention during negotiation

Your system should:

1. Extract 5 attributes from masked contracts of **two states** (dataset provided).
2. Extract the same 5 attributes from a standard template for each state.
3. Compare the extracted contract clauses with the corresponding state's standard clauses.

Classify each clause as **Standard** or **Non-Standard** based on similarity.

Key Focus Areas

1. **Contract Text Parsing & Clause Extraction**
    - Parse contracts in PDF or text format.
    - Identify section headers, sub-sections, and clauses.
    - Extract only the relevant parts for analysis (ignore extra text like signatures or metadata).
2. **Language Comparison & Classification Algorithm**
    - Compare each extracted clause from a contract with the corresponding standard template clause.
    - Decide whether each clause is Standard or Non-Standard.

Suggested Criteria for Language Comparison:

Criteria	**Description**
Exact Match	Clause matches the standard wording completely (word-for-word).

Semantic Similarity	Meaning is the same even if words differ (can use NLP or synonyms).

Examples of Standard vs. Non-Standard Classification Rules

1. **Exact Structural Match (Standard)**
   - Contract text matches the standard template in structure, phrasing, and intent.
   - Placeholders (e.g., XX%, [Fee Schedule]) are replaced with actual values.
   - Example:
     - [(XX%)] → 100% ✓ Classify as Standard
2. **Value Substitution (Still Standard)**
   - Percentage values, fee schedule names, or similar placeholders differ but follow the same formula/intent.
   - Example:
     - Standard: “[(XX%)] of the Fee Schedule”
     - Extracted: “95% of the Fee Schedule” ✓ Classify as Standard
3. **Minor Wording Differences (Standard)**
   - Stylistic or language changes without altering meaning.
   - Example:
     - Standard: "in effect on the date of service"
     - Extracted: "as in force at the time services are rendered" ✓ Classify as Standard
4. **Structural or Conditional Changes (Non-Standard)**
   - Additional conditions, carve-outs, or exceptions not in the standard.
   - Reimbursement tied to something other than the specified Fee Schedule.
   - Example:

- "Shall be 100% of the Fee Schedule except for cardiology services, which will be 80%." ✕ Classify as Non-Standard

5. **Reference to Different Methodologies (Non-Standard)**

   - ✕ Classify as Non-Standard

What we provide

You will be given contracts from two states, TN and WA. Contracts for each state are present in their respective folder. You will also receive **one standard template per state**. The extracted data from TN contracts must be compared with the TN template, and the same applies for WA

Folder: HiLabsAIQuest_ContractsAI.zip

1. **Masked Contracts Dataset (Folder: "Contracts")**
   - Several anonymized healthcare contracts from two different markets
   - Data has realistic structure and language but no confidential information
2. **Standard Template Dataset (Folder: "Standard Templates")**
   - One standard template for each market
   - Serves as the gold standard for comparison
3. **Clause Attributes Reference (Attribute Dictionary.xlsx)**
   - A guide to 5 key terms that need to be extracted along with section headers, examples, and definitions. THESE ARE THE ONLY 5 ATTRIBUTES THAT NEED TO BE EXTRACTED FOR THIS HACKATHON.

Rules

- Use **open-source libraries/models only** – no proprietary tools or paid APIs
- All data processing must happen **locally**, using the provided datasets
- Code must be **well-documented and easy to follow**
- Approach should be **modular and flexible**, so new clauses or rules can be easily added later

Tips for Success

- **Start small:** Work with one contract and one template first. Once it works, scale to multiple files
- **Keep it simple:** A basic keyword or rule-based system is acceptable; advanced NLP methods are optional
- **Maintain modularity:** Example file structure:
    - extract_clauses.py → for parsing
    - compare_clauses.py → for matching and scoring
    - main.py → to run the entire pipeline

Submission Format

Your submission must include:

1. **Analytics Platform**
    - Working backend code for parsing, comparing, and classifying clauses
2. **Summary Metrics**
    - Total clauses by category (Standard, Non-Standard)
    - Number of contracts with at least one Non-Standard clause
3. **README File**
    - GitHub repo link (must be in the first line)
    - Explanation of approach and architecture decisions
    - Clear setup and run instructions
4. **Demo Video or Script**
    - Short video or script showing your solution working on the provided data
    - A simple UI dashboard clearly categorizing each document and attribute with its language classification is recommended

Optional Bonus

- **Dockerize your solution** for portability and consistency (bonus points)

Evaluation Criteria

Criteria	Weightage
Analytical Accuracy (Correctness of classification)	30%
Extraction Accuracy (Correct parsing of clauses)	25%
Technical Implementation (Performance, clean code)	20%
Innovation & Creativity (Unique features, workflow improvements)	15%
Completeness & Documentation	10%

High-Level Architecture

Your solution should demonstrate three layers:

1. **Data Processing Layer**
   - Load and clean contract data
   - Extract relevant clauses
2. **Classification Engine**
   - Compare clauses with standard templates
   - Assign categories and calculate similarity scores
3. **Documentation & Reporting**
   - Output results and summary metrics
   - Provide clear setup instructions for users