Patient Notes to ICD 10 Code Mapping.

Introduction:

Medical coding maps clinical notes to ICD-10 codes, a process crucial for billing, insurance claims, and research. Manual coding is time-consuming, error-prone, and inconsistent. This project aims to automate ICD-10 classification for patient notes, focusing on 12 specific chronic disease codes across cardiovascular, respiratory, and endocrine disorders. The key challenge is that labels are absent, requiring NLP-based labeling and classification.

- **Reduces Manual Effort** – Automates coding, saving time for medical professionals.
- **Improves Accuracy** – Ensures consistency, reducing human errors.
- **Optimizes Billing** – Minimizes claim denials by ensuring correct coding.
- **Enhances Clinical Insights** – Helps healthcare providers analyze conditions efficiently.

Commercial Tools: https://www.getfreed.ai/

Data Understanding

Dataset Overview

1. The dataset consists of **unstructured patient notes** with no predefined labels.
2. Each patient note may be associated with **multiple ICD-10 codes**, making it a **multi-label classification** problem.

Challenges

1. **Lack of labeled data** – Required generating weak labels using an LLM-based approach.
2. **Unstructured text** – Necessitated **NLP-based preprocessing** to extract relevant information.
3. **Variability in terminology** – Different ways of describing the same disease required **robust keyword identification** to improve labeling accuracy.

Approach & Methodology

Step 1: Data Preparation

Generating Labels:

1. Used OpenAI GPT-4o Mini to predict ICD-10 codes for patient notes.
    a. Fine-tuned prompts based on manual evaluation to improve accuracy.

```
SYSTEM_PROMPT_v2 = '''
Given the following clinical note, extract the most specific and relevant ICD-10 codes
while adhering to medical coding guidelines.

Guidelines for ICD-10 Code Selection:
Primary Diagnosis (Main Condition Requiring Treatment):
Identify the most definitive diagnosis as the code with type "Primary" (e.g., a named
disease rather than symptoms).
If the primary condition has an underlying cause, ensure the cause is also coded.
Secondary Diagnoses (Related or Contributing Conditions):
Include coexisting conditions that impact management but are not the primary reason
for treatment.
Do not list symptoms separately if they are inherent to the primary diagnosis.
Coding Hierarchy Considerations:
Prefer specific codes over general ones (e.g., use P74.1 for neonatal dehydration
rather than E86.0).
Use combination codes when applicable instead of separate codes for diagnosis and
cause.
Ensure proper sequencing (e.g., if a condition is caused by another, the underlying
cause should be coded first when required).
Additional Considerations:
Genetic findings should be included only if they affect clinical care.
Exclude resolved conditions unless relevant to ongoing treatment.
'''

ICD10_DETECTION_PROMPT_v2 = """
  Clinical Note for coding:
  {clinical_note}
  """
```

3. Generated structured outputs containing:

- Code
- Code Type
- Description
- Confidence Score

```python
from pydantic import BaseModel, Field
from typing import List, Optional, Literal, Union
from enum import Enum

class MedicalEntity(BaseModel):
   text: str
   label: str

class ICD10Code(BaseModel):
   code: str
   code_type: Literal["Primary", "Secondary"]
   description: str
   confidence: float
   supporting_keyword_evidence: str


class ICD10Response(BaseModel):
   identified_codes: List[ICD10Code]

class SummaryResponse(BaseModel):
   summary: str
```

Exploring positive Labels with Keyword Search:

1. Identified frequently associated terms for each condition. Filtered records based on the keyword bank and Applied LLM filtering on these cases to extract positive samples.

3. Created a final dataset with 3,000 patient notes:

- 1,200 positive samples

- 1,200 negative samples

Step 2: Feature Engineering & Text Processing

- **Text Cleaning:**
    - Removed stopwords, special characters, and extra spaces.
- **Handling Class Imbalance:**
    - Used **balanced weights** in Logistic Regression to improve recall, ensuring rare conditions were properly detected.

Step 3: Model Selection

Baseline Model (For Benchmarking): Logistic Regression

- Framed as a **multi-label classification** task.
- Implemented **One-vs-Rest Logistic Regression** as an initial benchmark.

Classes	precision	recall	f1-score	support
E03	0.54	0.88	0.67	8
E10	0.26	0.63	0.37	8
E11	0.75	0.92	0.82	48
E66	0.64	0.82	0.72	11
E78	0.53	0.73	0.62	22
I10	0.76	0.89	0.82	46
I25	0.65	0.87	0.74	23
I48	0.82	0.90	0.86	20
I50	0.52	0.85	0.64	20
J44	0.54	0.88	0.67	8
J45	0.50	0.86	0.63	7
J84	0.53	0.67	0.59	12
Overall				
micro avg	0.63	0.85	0.72	233
macro avg	0.59	0.82	0.68	233
weighted avg	0.65	0.85	0.73	233
samples avg	0.34	0.41	0.35	233

A low Hamming Loss but moderate accuracy suggests that while individual labels are being predicted well, the model often misses at least one correct label in multi-label classification.

1.

Advanced Model: Clinical BERT

1. Implementation

- Using Bio_ClinicalBERT, a domain-specific variant of BERT, to improve ICD-10 code classification accuracy.

2. Challenges

- Lengthy Patient Notes: Clinical notes are often long, making it difficult to fit them within the token limit of BERT models.

- Chunking & Aggregation: Need a strategy to split long notes while ensuring meaningful context is retained when aggregating predictions.

- Alternative Approach: Instead of chunking, explore running BERT on summarized notes to improve efficiency.

Exploring Summarization

1. Hypothesis

- Summarization may improve classification accuracy by removing irrelevant details and focusing on key medical information.

2. Approach

- T5-based Summarization:

  - Use a pretrained T5 model to generate concise summaries of patient notes.

  - Compare classification performance with and without summarization to assess its impact.

3. Next Steps

- Experiment with different summarization strategies (extractive vs. abstractive).

- Analyze whether summarization helps or distorts crucial information.

- Fine-tune ClinicalBERT on summarized vs. full notes to find the optimal approach.

Reference:

1. HF Clinical BERT model
2. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1244/final-projects/ArjunJainDevanshuLadsariaRishiRajVerma.pdf
3. https://www.kaggle.com/code/aleksandrapestka/extract-entities-linked-to-umls
4. https://github.com/amzn/pecos
5.

Metrics Appendix

Since this is a multi-label classification problem where each patient note can have multiple ICD-10 codes, the choice of evaluation metrics depends on capturing both class-wise performance and overall model effectiveness.

1. Micro F1-score
   - Aggregates true positives, false positives, and false negatives across all labels before computing precision and recall.
   - Best when class imbalance is a concern, as it gives more weight to frequent labels.
   - Use case: If the goal is to ensure overall model performance across all ICD-10 codes.

2. Macro F1-score

- Computes F1-score for each label separately and then averages them.
- Gives equal importance to rare and frequent classes.
- Use case: If detecting all diseases is equally important, even the rare ones.

3. Hamming Loss
- Measures the fraction of incorrectly classified labels over all instances.
- Use case: If penalizing any incorrect label prediction (false positive or false negative) is critical.

4. Jaccard Similarity (Intersection over Union - IoU)
- Measures the overlap between predicted and actual labels per instance.
- Use case: If partial correctness matters (e.g., predicting 2 out of 3 correct ICD codes for a note).

Which One to Use?
- If class imbalance is high: Use Micro F1-score.
- If all classes are equally important: Use Macro F1-score.
- If minimizing mistakes is the top priority: Use Hamming Loss.
- If partial correctness is acceptable: Use Jaccard Similarity.

In this case, since recall is more critical in medical applications (missing a condition is worse than a false positive), Macro F1-score and Micro F1-score are the most relevant. Jaccard Similarity can also be used to evaluate how well predicted codes overlap with actual labels.