# KREL: Automatic Medical Coding via Knowledge-Guided Reasoning over Clinical Evidence with LLMs

**Anonymous ACL submission**


## Abstract

Automatic Medical Coding (AMC), which assigns standardized International Classification of Diseases (ICD) codes to clinical notes, is essential for medical reimbursement, quality reporting, and clinical research. Existing pre-trained language model (PLM)-based methods typically formulate AMC as an extreme multi-label classification problem over a predefined code set, while recent large language model (LLM)-based approaches instead frame it as generation or multi-step reasoning. However, key challenges remain, including the extreme length of clinical notes that hinders effective interpretation, the vast ICD label space, and complex coding rules that are not explicitly captured by LLMs. In this work, we propose Knowledge-Guided Reasoning over Clinical Evidence with LLMs (KREL), a framework that leverages LLMs for clinical text understanding and reasoning while integrating external ICD coding guidelines as structured knowledge. This design enables tight coupling between domain knowledge and LLM reasoning, reducing hallucinations and improving compliance with coding standards. Experiments on benchmark datasets show that KREL consistently outperforms strong PLM-based and state-of-the-art LLM-based baselines.


## 1 Introduction

Medical coding is the process of assigning relevant diagnosis codes to each clinical note, which is essential in public health and healthcare systems (Ji et al., 2024; Gao et al., 2024). This task is generally performed by experienced human annotators who follow the World Health Organization's ICD coding guidelines when assigning codes. This manual process is notoriously time-consuming and error-prone due to complex rules and the requirement of domain-specific knowledge (Yan et al., 2022; Motzfeldt et al., 2025).

The high cost of medical coding has motivated extensive research on AMC, which aims to automatically assign ICD codes to each clinical note. Existing methods can be broadly categorized into traditional machine learning-based and LLM-based approaches. The former formulates AMC as an extreme multi-label classification problem, where neural encoders or pre-trained language models (PLMs) directly map clinical notes to codes within a predefined label space (typically 50–100 labels, which is much smaller than the full ICD taxonomy of over 70K codes) (Huang et al., 2022; Liu et al., 2022; Edin et al., 2023). Although these methods achieve strong performance on benchmark datasets, they are typically constrained to a predefined label set, making it difficult to scale to the full and evolving ICD taxonomy. More recent LLM-based approaches reformulate AMC as a generation or multi-step reasoning task, leveraging prompting, external tools, or coding guidelines to better emulate the workflow of human coders (Motzfeldt et al., 2025; Zheng et al., 2025; Yuan et al., 2025).

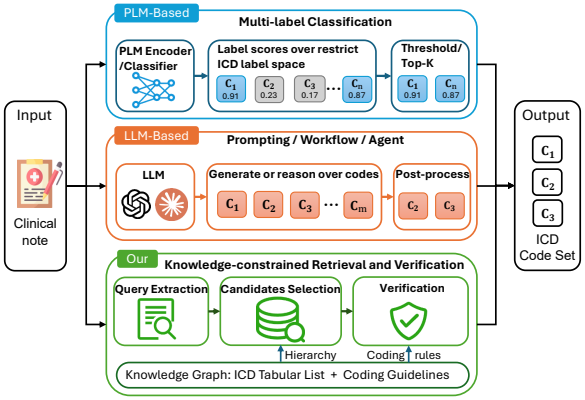


Figure 1: The difference between our framework and existing approaches. Our method better leverages LLM reasoning capabilities by integrating external knowledge.

Despite the superior performance of LLM-based approaches over traditional methods, LLMs alone remain insufficient for AMC due to three key chal-

lenges. First, most foundation LLMs are not specifically trained for AMC; even when they can process long clinical notes, they often fail to accurately map clinical concepts to the correct codes. Second, the ICD taxonomy contains over 70K labels, making the label space extremely large and causing LLMs to frequently miss rare or low-frequency ("cold") codes. Third, ICD coding is governed by complex rules and guidelines, which are not inherently encoded in LLMs, leading to outputs that may violate coding constraints or standards.

These challenges suggest that LLMs should be jointly leveraged with external knowledge to improve AMC accuracy. In light of this, we propose the **Knowledge-Guided Reasoning over Clinical Evidence with LLMs (KREL)** framework, which consists of three main components. **Query Extractor:** Given a clinical note, an LLM-based query extractor transforms code-relevant clinical descriptions into structured queries, enabling robust handling of long and unstructured narratives in clinical notes. **Candidate Selector:** Based on these queries, a guideline-driven knowledge-graph retrieval-augmented generation (KG-RAG) module retrieves and consolidates candidate ICD codes from the full ICD taxonomy, along with their definitions and coding rules. **Code Verifier:** The clinical note, together with the retrieved candidate codes, definitions, and coding rules, is then fed back into the LLM for verification, where the LLM reviews all evidence to make the final coding decisions. Notably, our design effectively integrates LLM reasoning with external ICD knowledge, enabling scalability to the full ICD-10 label space.

Figure 1 compares our framework with existing approaches and highlights, highlighting its novelty in integrating LLMs with external knowledge for AMC. We present our main contributions as follows:

- We propose the novel KREL framework for AMC. Instead of directly generating ICD codes from clinical notes, KREL formulates AMC as a retrieval, candidate refinement, and verification process grounded in structured ICD knowledge and coding guidelines.
- Our framework can scale to the full ICD coding space while leveraging ICD knowledge to guide the code reasoning process. In addition, we demonstrate its interpretability and effectiveness in improving AMC accuracy compared with baseline methods.
- We perform extensive experiments on various datasets, including MDACE, ACI-BENCH, and MIMIC-IV, demonstrating consistent improvements over competitive PLM-based and state-of-the-art LLM-based baselines.

## 2 Related Work

### 2.1 Automatic Medical Coding

Early AMC systems used rule-based and knowledge-driven NLP pipelines, which provided explicit control but required substantial maintenance across institutions and documentation styles (Farkas and Szarvas, 2008; Kang et al., 2013). The release of large EHR datasets such as MIMIC-III and MIMIC-IV helped shift AMC research towards extreme multi-label classification (Johnson et al., 2016, 2023). Neural models, including CNN- and RNN-based encoders with label-wise attention, improved code-specific evidence modelling (Mullenbach et al., 2018; Vu et al., 2020). Later work incorporated ICD hierarchy, lexical resources, and domain-adapted PLMs to better handle long notes and fine-grained code distinctions (Liu et al., 2022; Mahdi et al., 2024; Huang et al., 2022). These classification-based methods are effective in restricted or dataset-observed label spaces, but scaling to the complete ICD-10-CM taxonomy remains difficult (Huang et al., 2022; Edin et al., 2023).

LLM-based methods instead formulate AMC as code generation or multi-step reasoning. Direct prompting with general-purpose LLMs, including ChatGPT and Gemini, remains unreliable under large ICD label spaces because generated codes can be invalid, unsupported, or semantically close but coding-inaccurate (Achiam et al., 2023; Team et al., 2023; Mustafa et al., 2025). Recent workflow-based systems, such as Code Like Humans and MedDCR, decompose coding into multiple steps and incorporate ICD resources or coding procedures to improve reliability (Motzfeldt et al., 2025; Zheng et al., 2025; Yuan et al., 2025). These approaches move beyond direct prompting, but they still leave open how to combine full-space candidate retrieval, evidence grounding, and guideline-constrained verification within a single framework. This motivates our retrieval-and-verification formulation for AMC.

### 2.2 Knowledge-guided AMC

Retrieval-augmented generation (RAG) has been used to ground LLM outputs in external evidence for knowledge-intensive and clinical NLP tasks (Xiong et al., 2024; Zhao et al., 2025; Yang et al., 2025). Standard RAG is less suited to AMC because ICD-10-CM is not an unstructured document collection, but a large hierarchical label space governed by coding guidelines (CMS and NCHS, 2022; NCHS, 2022). Candidate construction therefore requires both semantic matching and structured coding dependencies

Knowledge graphs provide a natural way to encode medical concepts and relations, and have been used for entity linking, concept normalization, and clinical reasoning (Wu et al., 2025). Existing KG-based methods usually focus on concept-level inference rather than full-label-space ICD assignment. Our framework instead uses an ICD knowledge graph to support hierarchy-aware candidate retrieval and rule-aware verification. The hierarchy guides candidate search, while coding-rule relations are converted into verification context for evidence-grounded code selection.

## 3 KREL Framework

### 3.1 Problem Formulation

AMC is commonly formulated as an extreme multi-label classification task, which aims to map a clinical note to a set of ICD codes. Let $x$ denote a clinical note and $\mathcal{C}$ denote the complete ICD-10-CM code space,[1] which contains over 70K labels. The goal is to predict a code set $\hat{\mathcal{Y}} \subseteq \mathcal{C}$ that matches the gold code set $\mathcal{Y} \subseteq \mathcal{C}$.

In practice, $\mathcal{Y}$ may contain multiple interdependent labels constrained by the hierarchical structure of ICD codes and official coding guidelines. Thereby, AMC is more complex than flat label prediction without considering the interdependent relations between assigned labels. In addition, clinical expressions in free-text notes often do not align directly with standardized ICD concepts, which further complicates this problem.

To address these challenges, we reformulate AMC as a **hierarchy-aware retrieval and rule-aware verification process**. Instead of directly predicting codes from clinical notes, our framework first extracts evidence-grounded clinical queries and retrieves candidate ICD codes for each query using the ICD hierarchy. It then verifies the retrieved candidate codes against the clinical note, extracted evidence, and coding rules.

[1]ICD has multiple versions. This study mainly focuses on ICD-10-CM; adapting the framework to other versions would require rebuilding the corresponding code hierarchy and guideline resources.

### 3.2 Framework Overview

Figure 2 illustrates the KREL framework with a running example. Given a clinical note, the Query Extractor produces evidence-grounded clinical queries. The Candidate Selector retrieves and reranks ICD candidates for each query using ICD hierarchy and code descriptions, and converts coding-rule relations into rule hints and combination checks. The Code Verifier then judges the selected candidates against the full note, localized evidence spans, and rule context to produce the final ICD code set.

### 3.3 Query Extractor

For the query extractor, we first sectionize the clinical note and mark major note sections. The sectioned note is then processed by an LLM, which identifies coding-relevant diagnoses, conditions, symptoms, and clinical status descriptions, and rewrites them into concise queries for subsequent processing. The prompt used for query extraction by the LLM is provided in the Appendix E.

Formally, given a clinical note $x$, the query extractor produces a set of query-evidence pairs, represented by $\mathcal{Q}(x) = \{(q_i, E_i)\}_{i=1}^{n}$, where $q_i$ is the text describing diagnoses, conditions, symptoms, and clinical status, and $E_i$ denotes the associated evidence spans from the clinical note.

### 3.4 Knowledge-guided Candidate Selection

The selector maps each extracted query to ICD candidate lists and builds rule context for verification. It uses an ICD knowledge graph constructed from ICD code text, hierarchy, and guideline-derived relations.

#### 3.4.1 ICD Knowledge Graph Construction

We construct an ICD knowledge graph offline to encode the ICD knowledge used by downstream stages. The graph contains three types of information. First, each ICD code is associated with textual attributes, including its official description and inclusion terms, which support query–code matching. Second, ICD-10-CM provides a multi-level hierarchy, where broad categories are refined into more specific codes. Third, ICD coding guidelines define

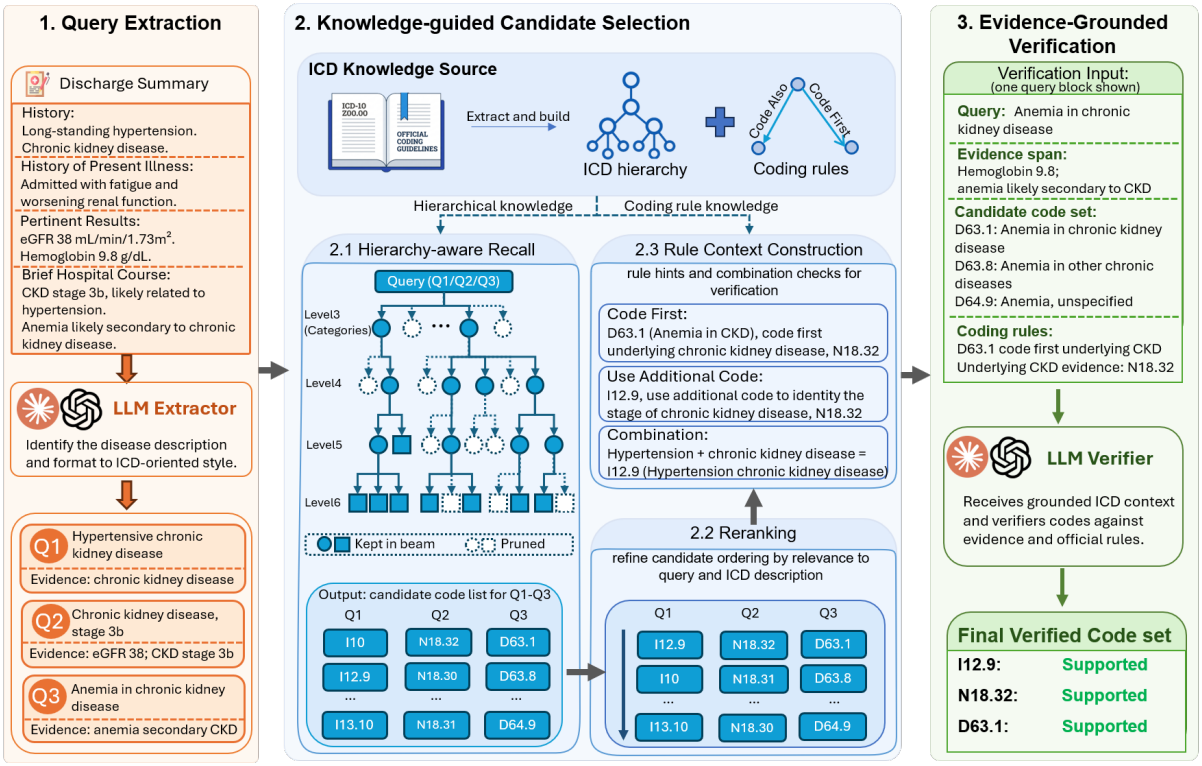


Figure 2: Overview of the KREL framework. (1) The query extractor reads the clinical note, identifies disease descriptions, and reformulates them into ICD-oriented queries (e.g., converting "CKD stage 3b" into "Chronic kidney disease stage 3b"). (2) The selector retrieves candidate ICD codes based on ICD hierarchy knowledge and coding rules. (3) The verifier makes the final coding decisions by jointly considering the candidate codes and all available information and evidence.

rule relations, including pairwise coding-rule relations such as *codeFirst*, *useAdditionalCode*, and *codeAlso*, and multi-code combination-code relations where required codes jointly support a target combination code.

Formally, we represent the graph as $\mathcal{G}_{\text{ICD}} = (\mathcal{V}_C, \mathcal{E}_H, \mathcal{E}_R, \mathcal{E}_M)$, where $\mathcal{V}_C$ denotes ICD code vertices, $\mathcal{E}_H$ denotes hierarchical relations, $\mathcal{E}_R$ denotes pairwise coding-rule relations, and $\mathcal{E}_M$ denotes multi-code combination-code relations. Each code $c_v \in \mathcal{V}_C$ is associated with textual attributes $d_v$, which are used for retrieval and reranking.

These graph components are used at different stages. The hierarchy $\mathcal{E}_H$ defines the search space for hierarchy-aware recall, while the code text $d_v$ supports semantic scoring and reranking. The rule relations $\mathcal{E}_R$ and $\mathcal{E}_M$ are not used to expand the ordinary candidate lists. They are converted into rule hints and combination checks before verification. The full KG schema is provided in Appendix B.2.

### 3.4.2 Hierarchy-aware Candidate Recall

Given a query $q_i$, the recall stage searches the ICD code hierarchy along $\mathcal{E}_H$ to obtain a core candidate set, denoted by $\mathcal{R}_i$. The relevance between $q_i$ and code $c$ is computed from their embeddings, where we adopt the same embedding LLM to generate embeddings for $q_i$. The instruction used for the embedding LLM is provided in Appendix E. Formally, let $\mathbf{z}_{q_i}$ and $\mathbf{z}_c$ denote the embeddings of $q_i$ and code $c$. Then, the relevance score between $q_i$ and $c$ is $r_{emb}(q_i, c) = \mathbf{z}_{q_i}^\top \mathbf{z}_c$.

Based on the relevance score, we retrieve ICD codes related to a query using the Hierarchy-Aware Beam Search (HBS) algorithm. To control the search cost, the retrieval process is constrained by a set of budget hyper-parameters $K_c$, $B$, $M$, $D$, and $K_f$, where $K_c$ denotes the number of initial search codes, $B$ is the beam size, $M$ is the maximum number of child codes explored for each parent code, $D$ is the maximum search depth, and $K_f$ is the maximum number of final selected codes.

Specifically, HBS starts from the top-$K_c$ ICD nodes that are most relevant to $q_i$, ranked in descending order according to $r_{emb}(q_i, c)$. The algorithm then progressively descends along the ICD hierarchy to search for more specific child codes. For each selected parent node with child nodes,

HBS retains the top-$M$ child codes with the highest scores, where the scoring function is defined later. At each level (i.e., search depth), all candidate child nodes are globally sorted, and only the top-$B$ nodes are preserved as the beam for the next search step.

For a child code $c$ expanded from its parent code $c'$, its score is computed as a combination of the parent score and the local relevance of the child node, i.e., $S(c) = \lambda S(c') + (1-\lambda) r_{emb}(q_i, c)$, to capture information propagated along the search path. In our implementation, we set $\lambda = 0.5$. Note that a leaf node may be reached via multiple paths; in such cases, we retain only the highest score among all paths. Finally, we select the top-$K_f$ leaf nodes with the highest scores to form $\mathcal{R}_i$. Details of the HBS algorithm are provided in Appendix B.3.

It is worth noting that HBS can outperform existing greedy-based retrieval algorithms (Boyle et al., 2023) in terms of recall, as it defers code pruning to leaf nodes. In contrast, greedy-based methods prune parent codes at earlier stages, which may lead to the premature removal of correct codes.

#### 3.4.3 Reranking

The candidate set produced by HBS, i.e., $\mathcal{R}_i$, still contains noisy codes, such as sibling codes or broadly related codes with similar descriptions. To filter out these noisy candidates, we rerank and subsequently truncate $\mathcal{R}_i$.

For reranking, we employ a cross-encoder LLM with pairwise relevance scoring. Specifically, for each $c \in \mathcal{R}_i$, we extract its ICD code description, including symptoms, concepts, and inclusion terms, from the KG. A pairwise reranker LLM is then used to assign a relevance score to each query–code pair $(q_i, c)$ based on the extracted code description. The instruction used for the LLM reranker to generate relevance score is presented in Appendix E. All candidate codes are subsequently sorted in descending order according to their relevance scores $r_{rerank}(q_i, c)$. Let $\tilde{\mathcal{L}}_i$ denote the reranked list of candidate codes.

Since $K_f$ is typically large, we further apply a note-level verification budget $K_V$ to reduce the cost of subsequent verification. For a note with $n$ extracted queries, we distribute this budget approximately evenly across the reranked lists $\{\tilde{\mathcal{L}}_i\}_{i=1}^{n}$ and retain the top-ranked codes from each list. If some queries contain fewer candidates than their allocated budget, the unused slots are reassigned to the highest-ranked remaining candidates across all queries. We denote the resulting budgeted candidate list for query $q_i$ as $\mathcal{L}_i$.

#### 3.4.4 Augmenting Candidate Codes with Rules

Until now, candidate codes have been selected solely by utilizing $\mathcal{E}_H$ in the KG. At this stage, we further incorporate the knowledge encoded in $\mathcal{E}R$ and $\mathcal{E}_M$ to facilitate code assignment.

We first consider the set $\mathcal{E}_R$. According to the ICD coding guidelines, there are three types of rule relations: *codeFirst*, *useAdditionalCode*, and *codeAlso*. Among them, the first two relation types are considered more critical. For a code $c \in \mathcal{L}_i$, we retrieve the top $K_a$ related codes $c'$ such that $c$ and $c'$ are connected through either *codeFirst* or *useAdditionalCode* relations. Here, both $c$ and $c'$ are selected for the same clinical note, although they may belong to different candidate lists $\mathcal{L}_i$. The retrieved codes are prioritized according to their reranking scores obtained in the reranking step. If fewer than $K_a$ related codes are retrieved, we further retrieve additional codes connected through the *codeAlso* relations by repeating the above process.

For each selected $c'$, we generate a textual description of the relation between $c$ and $c'$ as a relation hint. We aggregate all such relation hints to form a hint set, denoted as $H_c$. Let $\mathcal{H}_i$ denote the rule hint set of all $H_c$ from query $q_i$. Subsequently, $\mathcal{H}_i$ is used together with $q_i$ to facilitate the final code assignment.

Next, we consider the knowledge encoded in $\mathcal{E}M$. Each relation in $\mathcal{E}M$ specifies the relationship between a set of required codes and a target combination code. Since the required codes may be distributed across different queries, we construct the aggregated candidate set for clinical note $x$ as $\mathcal{L}_x = \bigcup_{i=1}^{n_x} \mathcal{L}_i$. If all required codes associated with a combination code in $\mathcal{E}M$ are present in $\mathcal{L}_x$, we include the corresponding combination rule, linking the required codes to the combination code, in the set $\Omega_x$.

### 3.5 Evidence-grounded Code Verification

We employ an LLM verifier to perform the final code assignment based on all the collected information. For each clinical note $x$, we invoke the LLM verifier once to predict codes using the inputs $x$, $q_i$, $E_i$, $\mathcal{L}_i$, and $\mathcal{H}_i$ for $1 \leq i \leq n$, as well as $\Omega_x$, following a two-step process. The instruction used for the LLM verifier is presented in Appendix E.

First, for each query $q_i$, the LLM verifier compares candidate codes in $\mathcal{L}_i$ against the full clinical note, the localized evidence spans $E_i$, and the corresponding code descriptions. Multiple codes may be selected for a single query, and it is also possible that no code is supported by a given query. When a candidate code is associated with any rule hint in $\mathcal{H}_i$, the verifier further evaluates the corresponding coding dependencies across all queries within $x$. For each verified code, the verifier LLM assigns one of the following ratings: SUPPORTED, POSSIBLE, or NOT SUPPORTED.

Second, after processing queries, we verify combination rules in $\Omega_x$. For each remaining combination rule, we send both the combination code, its required component codes, the clinical note and other information to the verifier, who will decide whether to support this combination code for $x$.

Finally, all codes rated as SUPPORTED or POSSIBLE are retained and included in the set $\hat{\mathcal{Y}}$ as prediction codes for the clinical note $x$.

# 4 Experimental Setup

## 4.1 Datasets

We evaluate our framework on three clinical datasets: MDACE (Cheng et al., 2023), ACI-Bench (Yim et al., 2023), and MIMIC-IV (Johnson et al., 2023). MDACE is a manually annotated medical coding dataset with ICD code assignments and code-related annotation information. ACI-Bench provides clinical notes with sentence-level links between text spans and ICD codes, supporting both performance evaluation and evidence-based analysis. MIMIC-IV is a large-scale de-identified hospital EHR database containing discharge summaries and ICD-coded diagnoses.

For MDACE and ACI-Bench, we use the official dataset splits and label inventories provided by the benchmark authors to ensure direct comparability with reported baselines.

We evaluate two label-space settings. In the *benchmark-label-space setting*, predictions are constrained to the predefined ICD label inventory of each benchmark. This setting is used for MDACE and ACI-Bench. In the *full-label-space setting*, methods select codes from the complete ICD-10-CM code space with over 70K labels. This setting is used for the fully labelled version of MDACE and a randomly sampled subset of MIMIC-IV, allowing us to evaluate coding performance under the complete ICD-10-CM label space.

## 4.2 Baselines

We compare our framework with two categories of baselines: **PLM-based methods** and **LLM-based methods**.

PLM-based methods formulate AMC as a supervised extreme multi-label classification task. In this category, **PLM-ICD** (Huang et al., 2022) and **PLM-CA** (Douglas et al., 2025) are representative baselines in previous AMC study. They perform code prediction over a predefined label space and use label-wise designs to better capture note–code correspondence, providing limited interpretability and auditability.

Instead, LLM-based methods formulate AMC as a generation or reasoning task. Existing works, such as **GPT-4o** (Hurst et al., 2024), **CoT** (Wei et al., 2022), and **CoT-SC** (Wang et al., 2022), rely on direct prompting or generic reasoning strategies to generate codes from clinical notes directly. Others, including **RRS** (Kwan, 2024), **MAC** (Li et al., 2024), **CLH** (Motzfeldt et al., 2025), and **MedDCR** (Zheng et al., 2025), model AMC as a multi-step workflow and leverage LLMs together with external tools or coding resources to better simulate the human coding process for AMC.

## 4.3 Implementation Details

We instantiate our framework with **GPT-4o** (Hurst et al., 2024) as the backbone LLMs, which are used for both query extractor, candidate selector and verifier. We construct the knowledge graph from the **ICD-10-CM Tabular List** (NCHS, 2022) and the corresponding **Official Coding Guidelines** (2022 version) (CMS and NCHS, 2022). We employ **Qwen3-Embedding-8B** (Zhang et al., 2025) as the embedding LLM and **Qwen3-Reranker-8B** (Zhang et al., 2025) as the reranking LLM in **Candidate Selector**.In the HBS algorithm, we set $K_c$=200, $B$=200, $M$=8, $D$=6 and $K_f$=30. After deduplicating candidates across queries, we keep at most $K_V = 50$ codes per note for final verification.

## 4.4 Evaluation Metrics

Following previous work (Zheng et al., 2025; Motzfeldt et al., 2025; Huang et al., 2022) in automatic medical coding we evaluate all methods using precision, recall and F1-score, and primarily report **micro-averaged** results. Although our method reformulates AMC as a retrieval-and-verification problem rather than a standard multi-label classification task, the final outputs are converted into the

| (a) Benchmark-label-space setting | | | | | | | |
|---|---|---|---|---|---|---|---|
| Method category | Model | MDACE (CM + PCS) | | | ACI-BENCH | | |
| | | Precision | Recall | F1 | Precision | Recall | F1 |
| PLM | ICD | **0.49** | 0.47 | 0.48 | 0.43 | 0.41 | 0.42 |
| | CA | 0.46 | 0.45 | 0.45 | 0.44 | 0.42 | 0.43 |
| LLM prompting | CoT | 0.30 | 0.31 | 0.30 | 0.35 | 0.50 | 0.41 |
| | CoT-SC | 0.39 | 0.43 | 0.41 | 0.36 | 0.59 | 0.44 |
| LLM workflow | RRS | 0.24 | 0.30 | 0.27 | 0.26 | 0.52 | 0.35 |
| | MAC | 0.27 | 0.31 | 0.29 | 0.23 | 0.50 | 0.31 |
| | CLH | 0.45 | 0.40 | 0.42 | 0.44 | 0.39 | 0.41 |
| | MedDCR | 0.41 | 0.55 | 0.47 | 0.43 | 0.67 | 0.52 |
| | KREL | 0.42 ± 0.006 | **0.59 ± 0.01** | **0.49 ± 0.007** | **0.61 ± 0.009** | **0.82 ± 0.01** | **0.70 ± 0.01** |
| **(b) Full-label-space setting** | | | | | | | |
| Method category | Model | MDACE (CM only) | | | MIMIC-IV-Subset | | |
| | | Precision | Recall | F1 | Precision | Recall | F1 |
| LLM prompting | Pure GPT | 0.38 | 0.25 | 0.30 | 0.47 | 0.24 | 0.31 |
| | CoT | 0.40 | 0.26 | 0.31 | **0.49** | 0.23 | 0.32 |
| | CoT-SC | 0.39 | 0.26 | 0.32 | **0.49** | 0.25 | 0.33 |
| LLM workflow | CLH | 0.40 | 0.23 | 0.29 | 0.44 | 0.18 | 0.25 |
| | KREL | **0.49 ± 0.004** | **0.53 ± 0.005** | **0.51 ± 0.003** | 0.45 ± 0.005 | **0.35 ± 0.001** | **0.39 ± 0.002** |

*Note*: Baseline results in Panel (a) are reported by Zheng et al. (2025).

Table 1: Performance comparison under benchmark-label-space and full-label-space settings

same encounter-level ICD code predictions used in previous work, ensuring direct and fair comparison.

## 5 Results and Analysis

### 5.1 Main Results

Table 1 reports results under both benchmark-label-space and full-label-space settings. Panel (a) shows that KREL remains competitive in the benchmark setting. It achieves the best F1 on both datasets, reaching 0.49 on MDACE and 0.70 on ACI-BENCH. The MDACE gain is modest and mainly recall-driven, while the result on ACI-BENCH shows higher gains in both precision and recall over PLM, prompting, and workflow baselines.

Panel (b) reports the results under full-label-space setting, which better reflects the main challenge addressed in this work. On MDACE, KREL improves F1 from the best baseline score of 0.32 to 0.51, with recall increasing from 0.26 to 0.53. On MIMIC-IV-Subset, KREL also obtains the best F1, improving from 0.33 to 0.39. The gains are smaller than in the benchmark setting, but they are consistent across datasets. These results indicate that retrieval and verification over structured ICD knowledge is more effective than direct prompting or workflow baselines when the full-label-space is considered.

| Variant | Type | Precision | Recall | F1 |
|---|---|---|---|---|
| KREL (full) | – | **0.49** | 0.53 | **0.51** |
| medSpaCy query | Replace | 0.35 | 0.19 | 0.24 |
| flat retrieval | Replace | 0.46 | 0.50 | 0.48 |
| w/o reranker | Remove | 0.46 | 0.46 | 0.46 |
| w/o rule injection | Remove | 0.47 | 0.50 | 0.49 |
| w/o verifier | Remove | 0.10 | **0.62** | 0.19 |

Table 2: Ablation study on MDACE under the full-label-space setting.

### 5.2 Ablation Study

Table 2 reports ablation results on MDACE under the full-label-space setting, where we separately remove or replace each critical component in our framework.

Since removing the query extractor entirely causes system collapse, we replace the LLM-based query extractor with the NER module from medSpaCy, which extracts medical terms from clinical notes and uses them directly as retrieval queries without evidence-grounded query reformulation. This replacement significantly degrades performance, reducing recall from 0.53 to 0.19 and F1 from 0.51 to 0.24, indicating that evidence-grounded query construction is crucial for achieving adequate candidate coverage.

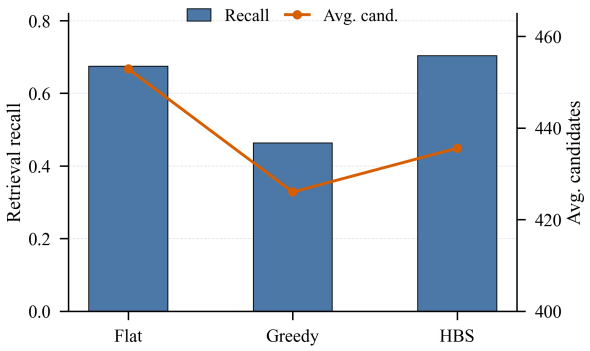


Figure 3: Comparison of different strategies for candidate code retrieval on MDACE under the full-label-space setting.

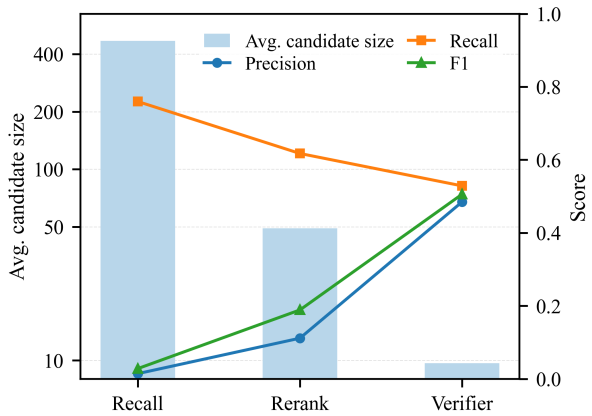


Figure 4: Evolution of the candidate set size against recall and precision performance.

We further examine the effect of removing the verifier, which leads to a substantial drop in precision from 0.49 to 0.10, despite an increase in recall to 0.62. This demonstrates that the verifier is essential for filtering unsupported retrieved candidates. Removing the reranker reduces F1 to 0.46, while removing coding rules slightly decreases F1 to 0.49. Overall, these results show that each component contributes meaningfully to the overall performance of the framework.

### 5.3 Retrieval Strategy Analysis

Figure 3 compares different retrieval strategies for candidate generation on MDACE under the full-label-space setting to demonstrate the superiority of HBS by comparing with alternatives, i.e., flat dense retrieval and greedy hierarchical search. Flat dense retrieval ranks ICD codes directly by dense query–code similarity without using the ICD hierarchy, while greedy hierarchical search traverses the hierarchy by keeping only the locally highest-scoring child at each step. HBS obtains the highest retrieval recall, improving over flat dense retrieval from 0.67 to 0.70 with a comparable candidate budget. Greedy hierarchical search performs much worse, dropping to 0.46 recall, which suggests that following only the locally best branch can prematurely discard relevant ICD subtrees. These results show that the ICD hierarchy is most effective when combined with HBS algorithm, which preserves branch diversity while controlling the search space.

### 5.4 Candidate Set Size Evolution

We further empirically analyze how the size of the candidate code set evolves across the framework pipeline. Figure 4 illustrates how the size evolves on MDACE under the full-label-space setting. The recall stage by retrieving candidate codes via HBS favours coverage with 470 codes per note on average. The average recall is 0.76, but its precision remains low at 0.01. Reranking reduces the candidate size to 49 and improves precision to 0.11 and F1 to 0.19, while retaining a recall of 0.62. The final verifier further reduces the output size to 10 codes on average and increases precision to 0.48 and F1 to 0.51, with recall decreasing to 0.53. The result indicates that candidate retrieval mainly provides coverage, while reranking and verification improve the reliability of the final code assignment.

## 6 Conclusion

In this work, we proposed **KREL**, a Knowledge-Guided Reasoning framework over Clinical Evidence with LLMs for automated medical coding. Unlike existing PLM-based methods that treat AMC as an extreme multi-label classification task and recent LLM-based approaches that rely on generation or multi-step reasoning, KREL explicitly integrates structured ICD coding guidelines into the reasoning process of LLMs. This design enables more effective interpretation of long and complex clinical narratives, better handling of the large ICD label space, and improved adherence to coding rules that are not inherently captured by LLMs. By tightly coupling external medical knowledge with LLM reasoning, KREL reduces hallucinations and enhances coding reliability. Extensive experiments on benchmark datasets demonstrate that KREL consistently outperforms strong PLM-based and state-of-the-art LLM-based baselines, showing its effectiveness and robustness for AMC.

## Limitations

**Candidate recall.** KREL verifies codes from retrieved candidates, so missed candidates cannot be recovered by later stages except for explicitly triggered combination-code checks. Maintaining high candidate recall under a fixed verifier budget remains challenging in the full ICD-10-CM label space, especially for rare or highly specific codes.

**Reliance on LLMs.** Our framework relies on LLMs for query extraction and verification. While this enables flexible evidence-grounded reasoning, it introduces cost, latency, and reproducibility considerations. Clinical deployment may require secure inference environments, locally hosted LLMs, or institution-approved LLM services.

**Data Scarcity.** Our evaluation is conducted on public EHR datasets using offline metrics. MDACE, ACI-Bench, and MIMIC-IV provide complementary benchmarks, but they do not capture the full diversity of institutional documentation styles or coding workflows. In addition, evidence annotations are not available for all datasets, limiting systematic evaluation of grounding quality. Future work should extend evaluation to more diverse clinical datasets and incorporate feedback from professional medical coders.

## Ethical Considerations

This work uses publicly available or credentialed-access clinical datasets for automated medical coding research. MDACE and ACI-Bench are public benchmark datasets, while MIMIC-IV is accessed under PhysioNet credentialing and used within the authorised data-use scope. Since AMC is a high-stakes clinical documentation and billing task, the proposed system should not be used as a replacement for professional medical coders or clinicians. Its outputs are intended to support coding review and should be subject to human validation before any clinical, administrative, or billing use. Deployment in real clinical environments would also require secure inference infrastructure, institution-approved models or services, and safeguards to prevent protected health information from being exposed to unauthorised systems.

## References


Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Joseph S Boyle, Antanas Kascenas, Pat Lok, Maria Liakata, and Alison Q O'Neil. 2023. Automated clinical coding using off-the-shelf large language models. *arXiv preprint arXiv:2310.06552*.

Hua Cheng, Rana Jafari, April Russell, Russell Klopfer, Edmond Lu, Benjamin Striner, and Matthew R Gormley. 2023. Mdace: Mimic documents annotated with code evidence. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7534–7550.

CMS and NCHS. 2022. *ICD-10-CM Official Guidelines for Coding and Reporting*. Centers for Medicare & Medicaid Services and National Center for Health Statistics, U.S. Department of Health and Human Services. FY 2022, updated April 1, 2022; October 1, 2021–September 30, 2022.

James C Douglas, Yidong Gan, Ben Hachey, and Jonathan K Kummerfeld. 2025. Less is more: Explainable and efficient icd code prediction with clinical entities. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30835–30847.

Joakim Edin, Alexander Junge, Jakob D Havtorn, Lasse Borgholt, Maria Maistro, Tuukka Ruotsalo, and Lars Maaløe. 2023. Automated medical coding on mimic-iii and mimic-iv: a critical review and replicability study. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2572–2582.

Richárd Farkas and György Szarvas. 2008. Automatic construction of rule-based icd-9-cm coding systems. *BMC bioinformatics*, 9(Suppl 3):S10.

Yue Gao, Yuepeng Chen, Minghao Wang, Jinge Wu, Yunsoo Kim, Kaiyin Zhou, Miao Li, Xien Liu, Xiangling Fu, Ji Wu, and 1 others. 2024. Optimising the paradigms of human ai collaborative clinical coding. *NPJ Digital Medicine*, 7(1):368.

Chao-Wei Huang, Shang-Chi Tsai, and Yun-Nung Chen. 2022. PLM-ICD: Automatic icd coding with pre-trained language models. In *Proceedings of the 4th Clinical Aatural Language Processing Workshop*, pages 10–20.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Shaoxiong Ji, Xiaobo Li, Wei Sun, Hang Dong, Ara Taalas, Yijia Zhang, Honghan Wu, Esa Pitkänen, and Pekka Marttinen. 2024. A unified review of deep learning for automated medical coding. *ACM Computing Surveys*, 56(12):1–41.

Alistair EW Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, and 1 others. 2023. Mimic-iv, a freely accessible electronic health record dataset. *Scientific Data*, 10(1):1.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3(1):1–9.

Ning Kang, Bharat Singh, Zubair Afzal, Erik M van Mulligen, and Jan A Kors. 2013. Using rule-based natural language processing to improve disease normalization in biomedical text. *Journal of the American Medical Informatics Association*, 20(5):876–881.

Keith Kwan. 2024. Large language models are good medical coders, if provided with tools. *arXiv preprint arXiv:2407.12849*.

Rumeng Li, Xun Wang, and Hong Yu. 2024. Exploring llm multi-agents for icd coding. *arXiv preprint arXiv:2406.15363*.

Leibo Liu, Oscar Perez-Concha, Anthony Nguyen, Vicki Bennett, and Louisa Jorm. 2022. Hierarchical label-wise attention transformer model for explainable icd coding. *Journal of Biomedical Informatics*, 133:104161.

Soha S Mahdi, Eirini Papagiannopoulou, Nikos Deligiannis, and Hichem Sahli. 2024. Co-occurrence graph-enhanced hierarchical prediction of icd codes. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 10146–10150. IEEE.

Andreas Motzfeldt, Joakim Edin, Casper L Christensen, Christian Hardmeier, Lars Maaløe, and Anna Rogers. 2025. Code like humans: A multi-agent solution for medical coding. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 22612–22627. Association for Computational Linguistics.

James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. Explainable prediction of medical codes from clinical text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: human language technologies, volume 1 (long papers)*, pages 1101–1111.

Akram Mustafa, Usman Naseem, and Mostafa Rahimi Azghadi. 2025. Evaluating hierarchical clinical document classification using reasoning-based llms. *arXiv preprint arXiv:2507.03001*.

NCHS. 2022. *ICD-10-CM Tabular List of Diseases and Injuries*. National Center for Health Statistics, Centers for Disease Control and Prevention. FY 2022, April 1, 2022 update.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Thanh Vu, Dat Quoc Nguyen, and Anthony Nguyen. 2020. A label attention model for icd coding from clinical text. *arXiv preprint arXiv:2007.06351*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Junde Wu, Jiayuan Zhu, Yunli Qi, Jingkun Chen, Min Xu, Filippo Menolascina, Yueming Jin, and Vicente Grau. 2025. Medical graph rag: Evidence-based medical large language model via graph retrieval-augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28443–28467.

Guangzhi Xiong, Qiao Jin, Zhiyong Lu, and Aidong Zhang. 2024. Benchmarking retrieval-augmented generation for medicine. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6233–6251.

Chenwei Yan, Xiangling Fu, Xien Liu, Yuanqiu Zhang, Yue Gao, Ji Wu, and Qiang Li. 2022. A survey of automated international classification of diseases coding: development, challenges, and applications. *Intelligent Medicine*, 2(03):161–173.

Rui Yang, Yilin Ning, Emilia Keppo, Mingxuan Liu, Chuan Hong, Danielle S Bitterman, Jasmine Chiat Ling Ong, Daniel Shu Wei Ting, and Nan Liu. 2025. Retrieval-augmented generation for generative artificial intelligence in health care. *Npj health systems*, 2(1):2.

Wen-wai Yim, Yujuan Fu, Asma Ben Abacha, Neal Snider, Thomas Lin, and Meliha Yetisgen. 2023. Aci-bench: a novel ambient clinical intelligence dataset for benchmarking automatic visit note generation. *Scientific Data*, 10(1):586.

Moy Yuan, Han-Chin Shing, Mitch Strong, and Chaitanya Shivade. 2025. Toward reliable clinical coding with language models: Verification and lightweight adaptation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 173–184.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, and 1 others. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Xuejiao Zhao, Siyan Liu, Su-Yin Yang, and Chunyan Miao. 2025. Medrag: Enhancing retrieval-augmented generation with knowledge graph-elicited reasoning for healthcare copilot. In *Proceedings of the ACM on Web Conference 2025*, pages 4442–4457.

Jiyang Zheng, Islam Nassar, Thanh Vu, Xu Zhong, Yang Lin, Tongliang Liu, Long Duong, and Yuan-Fang Li. 2025. Meddcr: Learning to design agentic workflows for medical coding. *arXiv preprint arXiv:2511.13361*.

# Appendix

## A Experimental Environment

All local computation, including embedding-based retrieval, candidate generation, reranking, post-processing, and metric computation, was conducted on a Linux server running `Ubuntu 22.04.5 LTS` with kernel `Linux 5.15.0-78-generic` and `glibc 2.35`. The software stack was based on `Python 3.12.3` and `PyTorch 2.8.0` compiled with `CUDA 12.8 (torch 2.8.0+cu128)`, with `cuDNN 9.10.2` enabled.

The hardware configuration consisted of one `NVIDIA GeForce RTX 5090` GPU with approximately `32GB` memory.

Embedding construction, dense retrieval, and reranking were executed with GPU acceleration on this server. LLM-based query extraction and verifier stages were executed through API-based batch inference using the corresponding model providers.

## B Implementation Details

### B.1 Dataset and Evaluation Scope

Table 3 summarizes the datasets and evaluation scope used in our experiments. MDACE and ACI-Bench are publicly available benchmark datasets derived from MIMIC clinical records, with additional task-specific annotation for medical coding evaluation (Cheng et al., 2023; Yim et al., 2023). MIMIC-IV is a credentialed-access clinical database distributed through PhysioNet (Johnson et al., 2023). We accessed and used MIMIC-IV after obtaining PhysioNet authorization, and all experiments were conducted within the permitted data-use scope.

For the benchmark-label-space setting, evaluation is restricted to the predefined label inventory of each benchmark. In particular, MDACE uses both ICD-10-CM and ICD-10-PCS labels to align with the baseline setting in Zheng et al. (2025). For the full-label-space setting, we evaluate over the complete ICD-10-CM diagnosis space, which contains 72,750 terminal diagnosis codes, using MDACE and MIMIC-IV. This setting tests whether methods can select codes from the full ICD-10-CM diagnosis space rather than from a benchmark-specific candidate label set. An anonymized implementation is available at `https://anonymous.4open.science/r/AMC-DD6E`.

### B.2 Graph Construction

We construct the ICD knowledge graph from ICD-10-CM code metadata, hierarchy information, and coding-rule annotations. The graph is defined as

$$\mathcal{G}_{\text{ICD}} = (\mathcal{V}_C, \mathcal{E}_H, \mathcal{E}_R, \mathcal{E}_M),$$

where $\mathcal{V}_C$ is the set of ICD code nodes, $\mathcal{E}_H$ is the set of hierarchy edges, $\mathcal{E}_R$ is the set of pairwise coding-rule edges, and $\mathcal{E}_M$ is the set of multi-code combination relations.

**Code nodes** For each ICD-10-CM code $c$, we create one code node $v_c \in \mathcal{V}_C$. Each code node stores the code string, official description, inclusion terms, and version information. We denote the textual attributes of code $c$ as $d_c$.

**Hierarchy edges** For each parent-child relation in the ICD taxonomy, we add a directed edge from the child code to its parent code. These edges form $\mathcal{E}_H$. For example, if code $c$ is a more specific code under parent code $p$, the graph contains an edge $c \to p$.

**Pairwise coding-rule edges** For pairwise coding-rule annotations, we add directed edges between ICD code nodes. Each edge is represented as

$$(c, \tau, c') \in \mathcal{E}_R,$$

where $c$ is the source code, $c'$ is the referenced code, and $\tau$ is the rule type. We include three rule types:

$$\tau \in \{\textit{codeFirst}, \textit{useAdditionalCode}, \textit{codeAlso}\}.$$

When a rule annotation refers to a code range, we expand the range into the corresponding ICD codes and create one edge for each target code.

**Multi-code combination relations** Some coding rules require multiple codes before a target combination code can be considered. We store each such

| Dataset | Label-space setting (size) | Code type | Test clinical notes | Test GT codes |
|---|---|---|---|---|
| MDACE | Benchmark (350) | ICD-10-CM + PCS | 122 | CM: 312; PCS: 38 |
| ACI-Bench | Benchmark (132) | ICD-10-CM | 120 | 132 |
| MDACE | Full (72750) | ICD-10-CM | 61 | 282 |
| MIMIC-IV-Subset | Full (72750) | ICD-10-CM | 500 | 1,812 |

Table 3: Dataset and evaluation scope.

rule as a combination relation

$$m = (\mathrm{Req}(m), \mathrm{Tar}(m)) \in \mathcal{E}_M,$$

where $\mathrm{Req}(m)$ is the set of required ICD codes and $\mathrm{Tar}(m)$ is the target combination code. In the graph implementation, each combination relation is stored as a rule node connected to its required code nodes and target code node.

### B.3 HBS Algorithm

**Algorithm 1:** Hierarchy-aware Beam Search for Candidate Recall

**Input:** Query $q_i$, ICD hierarchy $\mathcal{G}_H = (V_C, E_H)$, scorer $r_i(c) = r_{emb}(q_i, c)$, budgets $\Theta_R = (K_c, B, M, D, K_f)$, weight $\lambda$
**Output:** Recalled candidate list $\mathcal{R}_i$
Define $\mathrm{Top}_k(A, s)$ as the top-$k$ elements in $A$ ranked by score $s$;
Define $S(x)$ as the node score;
Initialize beam $\mathcal{B}$ with $\mathrm{Top}_{K_c}$ ICD category codes ranked by $r_i$;
Initialize scored candidates $\mathcal{A} \leftarrow \emptyset$;
**for** $d = 1$ **to** $D$ **do**
  $\mathcal{N} \leftarrow \emptyset$;
  **foreach** $(p, S(p)) \in \mathcal{B}$ **do**
    Let $\mathrm{Child}(p)$ be the child codes of $p$ in $\mathcal{G}_H$;
    **if** $\mathrm{Child}(p) = \emptyset$ **then**
      Add $(p, S(p))$ to $\mathcal{A}$;
      **continue**;
    **foreach** $u \in \mathrm{Top}_M(\mathrm{Child}(p), r_i)$ **do**
      $S(u) \leftarrow \lambda S(p) + (1 - \lambda) r_i(u)$;
      Add $(u, S(u))$ to $\mathcal{N}$;
      **if** $u$ *is a leaf code* **then**
        Add $(u, S(u))$ to $\mathcal{A}$;
  **if** $\mathcal{N} = \emptyset$ **then**
    **break**;
  $\mathcal{B} \leftarrow \mathrm{Top}_B(\mathcal{N}, S)$;
$\mathcal{R}_i \leftarrow \mathrm{Top}_{K_f}(\mathcal{A}, S)$;
**return** $\mathcal{R}_i$;

## C Experiment

### C.1 Detailed Evaluation Metric

We report micro-averaged precision, recall, and F1 over all notes. Precision and recall are then defined as

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}},$$

and

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}.$$

The F1 score is the harmonic mean of precision and recall:

$$\mathrm{F1} = 2 \cdot \frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}.$$

In AMC, precision measures how many predicted ICD codes are correct, while recall measures how many gold ICD codes are recovered. F1 provides a single measure of the trade-off between assigning accurate codes and covering all relevant diagnoses. This is important because clinical notes may contain multiple reportable conditions, and missing a relevant code or assigning an unsupported code can both reduce coding quality.

### C.2 Error Analysis

Table 4 summarizes error patterns on MDACE under the full-label-space setting. KREL predicts more codes per note on average than the LLM prompting and LLM-agent baselines, with 9.67 predicted codes per note compared with 5.98 and 5.11. This leads to more false positives, but it also substantially reduces false negatives. KREL has 255 false negatives, compared with 398 for LLM prompting and 416 for the LLM-agent baseline. The LLM-agent baseline has the fewest false positives and the lowest sibling-code false-positive rate, but this pattern is partly explained by its more conservative output size. It predicts fewer codes per note and misses more gold codes. KREL shows

| Metric | LLM prompting | LLM-agent (CLH) | KREL |
|---|---|---|---|
| Average predicted codes per note | 5.98 | 5.11 | 9.67 |
| False positives | 222 | **187** | 304 |
| False negatives | 398 | 416 | **255** |
| Combination-code recall | 0.077 | 0.077 | **0.615** |
| Sibling-code FP rate | 0.329 | **0.219** | 0.227 |

Table 4: Comparison of error patterns across method families on MDACE ICD-10-CM discharge summaries under the full-label-space setting.

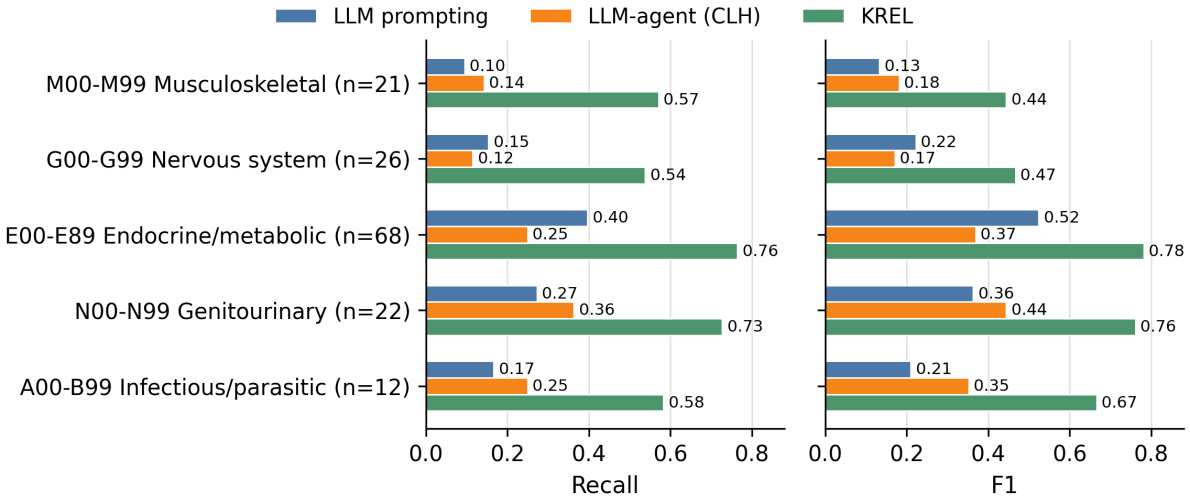


Figure 5: Performance by ICD-10-CM chapter on MDACE under the full-label-space setting. The figure compares recall and F1 for LLM prompting, the CLH LLM-agent baseline, and KREL across five ICD chapters.

the clearest advantage on combination-code recall. It reaches 0.615, while both baselines obtain 0.077. This gap suggests that the retrieval-and-verification pipeline is more effective for codes that require information from multiple conditions or query blocks. The metric sibling-code FP rate means number of false-positive codes that are siblings of a gold code. KREL also keeps the sibling-code false-positive rate close to the LLM-agent baseline, with 0.227 compared with 0.219, despite predicting a larger code set. This indicates that the additional predictions do not mainly come from uncontrolled sibling-code expansion.

Figure 5 provides chapter-level results, indicating that the main gains are concentrated in several ICD chapters. KREL improves both recall and F1 across the five reported ICD chapters. The largest gains appear in endocrine and metabolic diseases, genitourinary diseases, nervous system diseases, musculoskeletal diseases, and infectious diseases. For example, KREL reaches 0.76 recall and 0.78 F1 in endocrine and metabolic diseases, and 0.73 recall and 0.76 F1 in genitourinary diseases. The baselines remain much lower in these chapters, especially in musculoskeletal and nervous system codes. These results indicate that KREL improves coverage in several clinically diverse chapters, rather than only increasing predictions in a single code group.

## D Evidence Alignment Analysis

Table 5 evaluates whether the verifier's returned evidence aligns with human annotations on MDACE under the full-label-space setting. We compute these metrics only for true-positive code predictions, since false-positive predictions do not have corresponding gold evidence annotations. Among 286 true-positive code pairs, the verifier provides at least one evidence quote for every prediction, yielding 100% evidence coverage. In 90.6% of cases, at least one returned quote contains the human-annotated clinical mention for the same ICD code. The mean semantic cosine similarity between verifier-provided evidence and gold evidence is 0.746, suggesting that the returned quotes are usually close to the annotated supporting evidence even when the exact mention is not fully matched.

| Metric | Value |
|---|---|
| True-positive code pairs | 286 |
| Evidence coverage | 100% |
| Mention-anchor coverage | 90.6% |
| Mean semantic cosine | 0.746 |

Table 5: Evidence alignment analysis on MDACE full-label-space ICD-10-CM discharge-summary evaluation. Evidence quality is evaluated only for true-positive code predictions.

Evidence coverage measures whether the verifier returns any evidence quote for a true-positive prediction. Mention-anchor coverage measures whether a returned quote contains the human-annotated clinical mention. Mean semantic cosine is computed by embedding verifier-provided evidence quotes and gold evidence sentences with `sentence-transformers/all-MiniLM-L6-v2`, and taking the maximum cosine similarity over all predicted–gold evidence pairs for each true-positive code prediction.

## Generative AI Usage

We used ChatGPT for grammar checking, language polishing, and limited debugging support. We did not use generative AI tools to generate research ideas, experimental results, or references. All suggestions were reviewed and verified by the authors.

## E Prompt Template

## Query Extraction Prompt

You are an ICD-10-CM inpatient discharge-note diagnosis/status query extractor. Do not output ICD codes. Output only strict JSON clinical concept objects for candidate retrieval and later verification.

**Goal:**
Build evidence-grounded clinical queries that preserve the documented specificity needed for ICD-10-CM candidate recall and reranking.

**Extraction policy:**

- Use only provider-documented diagnoses, conditions, and clinically relevant statuses from the note.
- Do not infer diagnoses from labs, imaging, or medications alone.
- Include one principal reason, active conditions that affected the admission, and relevant chronic/history/status conditions that influenced care.
- Include symptoms only when no definitive diagnosis is documented for that problem or when the symptom is handled as a separate active problem.
- Preserve documented acuity, site, laterality, severity, stage, and causal linkage in the search query. Do not guess missing specificity.
- Keep status/history concepts only when explicitly documented and clinically relevant to the admission or discharge plan.

**Section awareness:** Scan discharge diagnoses, hospital course, assessment/plan, problem list, past medical history, social history, and medications before finalizing the query set. Record the section for each evidence span.

**Output schema:**

```
{
  "principal_reason": { ...one diagnosis object... },
  "active_conditions": [ ...diagnosis objects... ],
  "history_conditions": [ ...diagnosis objects... ]
}
```

Output strict JSON only with no extra keys or explanatory text:

**Diagnosis object schema:**

```
{
  "base": "core clinical concept",
  "query": "search string with explicit documented modifiers/status wording",
  "modifiers": {
    "temporality": "acute"|"chronic"|"subacute"|null,
    "site": string|null,
    "linkage": [
      {"type": "with"|"without"|"due_to"|"secondary_to", "text": string}
    ]
  },
  "evidence": [
    {
      "section": "DISCHARGE_DIAGNOSES|HOSPITAL_COURSE|ASSESSMENT_PLAN|PAST_MEDICAL_HISTORY|
          PROBLEM_LIST|MEDICATIONS|SOCIAL_HISTORY|OTHER",
      "span": "verbatim short quote from the note"
    }
  ]
}
```

Figure 6: Prompt for extracting and formatting queries from clinical note samples.

## Rule-Aware Verification Prompt

**Task Instruction**
You are retrieving ICD-10-CM concept descriptions. Given a short disease/clinical concept description (often produced by an LLM),return the most semantically matching ICD-10-CM concept descriptions.Be robust to paraphrases and minor wording differences.

Figure 7: Instruction for Qwen3-Embedding-8B

## Rule-Aware Verification Prompt

**Task Instruction**
You are reranking ICD-10-CM concept descriptions. Given a clinical concept query, return the most semantically matching ICD-10-CM concepts. Prefer exact clinical meaning, preserve documented specificity, and penalize unsupported qualifiers.

Figure 8: Instruction for Qwen3-Reranker-8B

## Rule-Aware Verification Prompt

**Task:**
Given a clinical note, evidence-grounded query blocks, and reranked ICD-10-CM candidate descriptions, verify which candidate codes apply.

**Hard constraints:**

1. Do not introduce a code outside the provided global candidate set, except for a Stage 2 combination target explicitly listed in the note-level combination checks.
2. Base decisions on the clinical note. Evidence blocks are localization hints and may be incomplete.
3. Every selected code must include one or two short verbatim note quotes.

**Verdicts:**

- **SUPPORTED:** explicitly documented for this admission or clearly stated in a clinically relevant problem/history/status context.
- **POSSIBLE:** clinically plausible from the note but less explicit than a supported diagnosis.

**Rule constraint:** Some candidate codes include normal rule hints such as "codeFirst -> X", "useAdditionalCode -> Y", or "codeAlso -> Z". If you select such a candidate as SUPPORTED or POSSIBLE, you must also select at least one referenced rule code globally across the note.

**Stage 1: query-level candidate verification**

- Verify candidates within each evidence-grounded query block.
- Use the clinical note, evidence spans, candidate descriptions, and normal coding-rule hints in the candidate block.
- Prefer the supported candidate with the best documented specificity.

**Stage 2: across-query combination verification**

- After Stage 1, review the note-level combination-rule checks.
- Combination codes may depend on conditions supported across different query blocks, so adjudicate them at the note level.
- Add a combination target only when it is explicitly listed in the combination checks and the required component code families are jointly supported by Stage 1 decisions or by the note.
- Stage 2 may change only the listed combination targets.

**You are given:**
**The full clinical note:**
`{full_clinical note_text}`

**Query, Evidence, Candidate code and Rule Hints for Stage 1**
`Query` + `Evidence` + `candidate_code_list_text`+`Rule_hints_text`

**Combination Rules for stage 2**
`{combination_rules_text}`

**Output (STRICT JSON):** Only include SUPPORTED or POSSIBLE codes.

```
{
  "verifications": [
    {
      "code": "I10",
      "verdict": "SUPPORTED|POSSIBLE",
      "evidence": ["verbatim quote", "verbatim quote"]
    }
  ]
}
```

Figure 9: Prompt for rule-aware and evidence-ground verification samples.