Enhancing LLM Medical Coding with Structured External Knowledge

Anonymous ACL submission

Abstract

Accurate medical coding requires consulting authoritative resources such as the ICD tabular list and coding guidelines. Existing LLM-based automated methods largely rely on LLMs' internal knowledge, which is prone to hallucination and cannot keep pace with guideline updates. We introduce RAG-Coding, an agentic, training-free method that augments LLMs with structured external knowledge: the tabular list is encoded as a knowledge graph capturing hierarchical and instructional code relationships, and the guidelines are distilled into concise, code-specific summaries rather than retrieved as raw text. To enable our study, we also introduce MDACE-2025, expert re-annotations of the MDACE dataset under the 2025 ICD-10-CM/PCS guidelines, adding code sequencing and justification comments. On MDACE, RAG-Coding outperforms the best LLM-based baseline by 3–13% in micro-F1 across five LLM backbones, and achieves comparable micro- and macro-F1 to the supervised state-of-the-art, with higher recall (+11%) at the cost of precision (−6%). On MDACE-2025, RAG-Coding outperforms all baselines, demonstrating effective generalisation to updated guidelines. Ablations confirm stepwise gains, highlighting the importance of integrating structured external knowledge for LLM-based medical coding.

1 Introduction

Medical coding maps clinical documentation to standardised alphanumeric codes, forming the foundation of hospital billing, medical research, and national health policy (Dong et al., 2022). Traditional manual coding is time-consuming; some hospitals experience backlogs extending over a year (Alonso et al., 2020). This burden has motivated research into automated medical coding (Edin et al., 2023).

Recent advancements in large language models (LLMs) have prompted research into their application for medical coding, leveraging their robust instruction-following abilities and extensive knowledge encoded in their parametric space. However, early results (Boyle et al., 2023; Falis et al., 2024) suggest LLMs, whether open-source or commercial, are not yet reliable coders *out-of-the-box*, demonstrating no performance advantage over the BERT-based supervised state-of-the-art (Huang et al., 2022; Edin et al., 2023).

Accurate medical coding requires understanding authoritative coding resources (Dong et al., 2022; Motzfeldt et al., 2025). Two primary examples are the ICD-10-CM tabular list[1] and the ICD-10-CM coding guidelines[2]. Professional coders routinely consult these resources (Dong et al., 2022; Motzfeldt et al., 2025). However, existing LLM-based automated methods largely overlook them, relying solely on LLMs' internal knowledge, which is prone to hallucination and cannot keep pace with the annual updates of coding resources (CMS, 2026). Existing datasets have similarly failed to keep pace: widely used datasets such as MIMIC-III, MIMIC-IV, and MDACE (Johnson et al., 2016, 2023; Cheng et al., 2023) were annotated under pre-2022 guidelines, limiting evaluation against current clinical standards. MIMIC's annotations are also noisy and lack justification for coding decisions (Cheng et al., 2023), while MDACE omits code sequencing—a mandatory workflow component that distinguishes principal from secondary diagnoses and is required for reimbursement.

To address the dataset gap, we introduce **MDACE-2025**, a new ICD-10-CM/PCS coding dataset providing expert re-annotations of the MDACE test set under the 2025 coding guidelines, with code sequencing and justification comments for complex coding decisions. Such coding justifications are important as they make annotated codes auditable, clinically traceable, and easier for

[1] A hierarchical reference organising codes from broad to specific diagnoses, with brief code usage notes (see Section 3).

[2] A 100+ page narrative document specifying how codes should be applied in clinical practice (see Section 3).

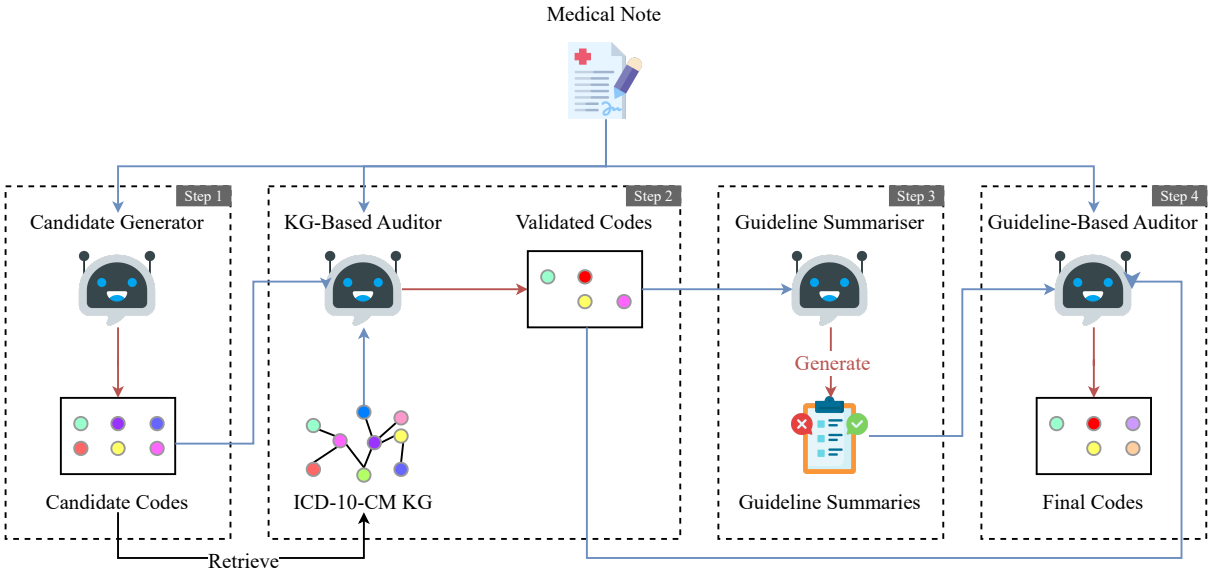

Figure 1: Overall architecture of the proposed RAG-Coding method. Blue lines denote input to the agents. Red lines denote output of the agents. Black lines denote intermediary operations that do not involve any agents.

human coders to verify and correct.

To address the methodological gap, we introduce **RAG-Coding**, an agentic, training-free automated coding method. How external knowledge is represented and integrated matters as much as whether it is available to the LLM; RAG-Coding explicitly grounds LLM coding decisions in structural knowledge: the ICD-10-CM tabular list is encoded as a knowledge graph (KG) capturing hierarchical and instructional code relationships, and the coding guidelines are distilled into concise, code-specific summaries rather than retrieved as raw text. RAG-Coding decomposes the coding task into four sequential steps: (1) generate candidate codes; (2) validate them using the KG; (3) retrieve and summarise relevant guidelines into code-specific rules; and (4) further validate using those summaries. Steps 2 and 3 define the methodological core: together, they determine how external coding knowledge is represented structurally rather than as raw text. An overview of RAG-Coding is shown in Figure 1. Of note, as the LLM-based baselines compared in this work focus exclusively on ICD-10-CM, we align our evaluation accordingly.

Our contributions are as follows:

1. We introduce MDACE-2025, the only public ICD coding dataset with 2025-guideline annotations, code sequencing, and expert justifications, opening research on guideline drift, principal-diagnosis coding, and error analysis.
2. We develop RAG-Coding, an agentic, training-free method supporting the full U.S. ICD-10-CM system (≈70K codes), grounding LLM decisions in structurally encoded knowledge: the ICD-10-CM tabular list as a KG and guidelines as code-specific summaries.
3. On MDACE, RAG-Coding achieves accuracy on par with the state-of-the-art (SOTA) supervised baseline, PLM-ICD (Huang et al., 2022) and outperforms all LLM-based baselines. On MDACE-2025, RAG-Coding outperforms all baselines, demonstrating effective generalisation to evolving guidelines.

2 Related Work

2.1 Automated Medical Coding

Automated medical coding has been studied for decades, with approaches ranging from rule-based systems (Farkas and Szarvas, 2008) to deep learning models that frame the task as multi-label classification (Mullenbach et al., 2018; Cao et al., 2020). BERT-based models pre-trained on biomedical corpora, such as PLM-ICD (Huang et al., 2022), are the supervised SOTA (Edin et al., 2023; Gan et al., 2025; Motzfeldt et al., 2025).

LLMs have been applied to ICD coding in zero- and few-shot settings (Yang et al., 2023; Boyle et al., 2023), but remain prone to hallucination and fail to adhere to mandatory coding rules. Retrieval-augmented generation (RAG) (Lewis et al., 2020) has emerged as a practical solution (Baksi et al., 2025; Motzfeldt et al., 2025). To our knowledge, CLH (Motzfeldt et al., 2025) is the only existing

method that combines RAG with explicit coding-rule enforcement, retrieving raw text excerpts from the tabular list and coding guidelines. However, CLH does not fully leverage the structure of these resources: the tabular list is treated as flat text despite its inherent hierarchical and instructional properties, and the retrieved guideline excerpts often span multiple pages and cover broad code groups, making code-specific validation difficult. RAG-Coding addresses both limitations by constructing an extensible KG from the tabular list and distilling the narrative guidelines into concise, code-specific rules for targeted validation.

2.2 Structured Knowledge Integration

Substantial prior work has integrated structured knowledge in neural networks for medical coding. Rios and Kavuluru (2018) encode ICD parent-child relationships via a graph convolutional network over label embeddings, enabling generalisation to rare and unseen codes. Related efforts incorporate code synonyms (Yuan et al., 2022) and co-occurrence graphs (Cao et al., 2020). Beyond medical coding, structured inputs have been explored for LLMs across a range of tasks, including question answering over knowledge graphs (Jiang et al., 2023), text-to-SQL generation (Xie et al., 2024), factual reasoning over heterogeneous structured data (Huang et al., 2025), and abnormal event forecasting with temporal knowledge graphs (Chi and Hsieh, 2025). RAG-Coding is methodologically situated within both lines of work, integrating structured ICD knowledge at inference time for LLMs.

2.3 Medical Coding Datasets

MIMIC-III, MIMIC-IV, and MDACE are widely used public ICD datasets (Johnson et al., 2016, 2023; Cheng et al., 2023). MIMIC's annotations are known to be noisy and lack justification for coding decisions (Dong et al., 2022; Cheng et al., 2023). MDACE addresses these quality concerns by providing professionally re-annotated ICD-10-CM/PCS codes for a subset of MIMIC-III, achieving high inter-annotator agreement (Krippendorff's $\alpha = 0.97$ across 384 inpatient encounters). Nevertheless, all three datasets were annotated under pre-2022 guidelines, limiting evaluation against current clinical standards. Moreover, MDACE omits code sequencing—a mandatory component of real-world coding workflows. These limitations motivate our introduction of MDACE-2025, described in Section 4.

3 Background: Common Coding Knowledge Sources

Below are two authoritative medical coding resources commonly used by human coders:

ICD-10-CM Tabular List. The tabular list groups the codes in a tree-structured hierarchy that defines parent-child relationships (i.e., from broad categories to specific diagnoses). It also contains instructional notes that define code usage and relationships, such as: (1) *includes*: explains the scope of a broad code category; (2) *inclusion_term*: lists synonymous or included terms for a given code; (3) *use_additional_code*: indicates codes that must also be used; (4) *excludes1*: indicates mutually-exclusive codes.

ICD-10-CM Coding Guidelines. The guidelines span over 100 pages and consist of narrative rules that human medical coders follow for accurate, consistent, and complete coding. The guidelines include two main components: general conventions and chapter-specific instructions. The general conventions establish overall coding principles, whereas the chapter-specific instructions address coding nuances within particular disease categories. A snippet of the guidelines' table of contents is displayed at the top right of Figure 2.

4 MDACE-2025

We introduce **MDACE-2025**, a medical coding dataset providing expert re-annotations of the MDACE test set under the April 2025 ICD-10-CM/PCS coding guidelines. Compared to the original annotations, MDACE-2025 also adds code sequencing and expert justification comments for complex coding decisions.

Annotation Process. Re-annotation was conducted by an experienced certified medical coder holding the AHIMA Certified Coding Specialist (CCS) and AAPC Certified Professional Coder (CPC) credentials. The annotator reviewed each encounter holistically using all available notes and the coding guidelines. Due to time and budget constraints, re-annotation was limited to the MDACE test set.

Annotation Agreement. We quantify the shift between MDACE and MDACE-2025 using Krippendorff's α, yielding $\alpha = 0.67$ for diagnosis codes, $\alpha = 0.41$ for procedure codes, and $\alpha = 0.66$ overall. Moderate agreement is expected

given the complexity of fine-grained ICD coding: prior work reports pairwise Jaccard similarity of 0.31–0.53 (Fung et al., 2019; Nesterov et al., 2025), and MDACE itself achieved an initial $\alpha = 0.53$ across 384 inpatient encounters. The lower procedure code agreement ($\alpha = 0.41$) is largely attributable to shifts in guidelines: the 2025 guidelines require procedure codes to be explicitly supported by operative notes, substantially reducing procedure code counts relative to MDACE. Full annotation statistics and code distribution shifts are provided in Appendix C.

Research Value. MDACE-2025 enables several research directions not supported by existing public datasets. First, its annotations under the 2025 guidelines allow evaluation of how well automated coding models generalise to evolving clinical standards. Second, code sequencing supports targeted research on principal diagnosis coding, which carries high billing priority and is mandatory in real-world reimbursement workflows. Third, expert justification comments allow researchers to make informed decisions when designing models and analysing errors.

5 Method

In this section, we introduce RAG-Coding, which consists of four main steps: (1) initial ICD code assignment, (2) KG-based validation, (3) guideline retrieval and summarisation, and (4) final validation using guideline summaries. Each step is executed by one agent. The methodological core lies in steps 2 and 3, which define *how* external coding knowledge is structurally encoded. The prompts used by the agents are available in Appendix A.

5.1 Problem Definition

Let x_i denote an input medical note. The objective is to map x_i to y_i, a set of valid ICD codes. This task is traditionally framed as multi-label classification, where models predict over a fixed set of codes. Following recent studies (Yang et al., 2023; Boyle et al., 2023; Baksi et al., 2025; Motzfeldt et al., 2025), we frame the task as a generation problem: given an LLM-based system $\mathcal{S}$, the output codes are produced iteratively such that $y_i = \mathcal{S}(x_i)$. This generative formulation enables $\mathcal{S}$ to produce any ICD code, whereas classification-based approaches can only output codes seen during training.

5.2 Step 1: Initial ICD Code Assignment

This step generates an initial set of relevant ICD codes, reducing the number of potential codes to a more manageable size. Prior studies (Baksi et al., 2025; Motzfeldt et al., 2025) achieve this via named entity recognition (NER) combined with semantic search—extracting diagnoses from medical notes and matching them to ICD code descriptions. Although this approach achieves great recall, it falls short in precision. Baksi et al. (2025) show that out-of-the-box LLMs tasked with direct ICD code generation can achieve similar recall but face the same precision challenge. Building on this, we use dynamic two-shot examples (see Appendix B for more details) and instruct our ***Candidate Generator*** agent to simultaneously assign a set of relevant ICD codes and their supporting evidence. This evidence generation is essential, as it strengthens the reasoning process that links documented diagnoses to their corresponding codes.

Although the generated evidence could be used to retrieve additional relevant codes, like in the NER + semantic search approach, we choose not to pursue this direction. Our preliminary finding indicates that the generated code set frequently overlaps with the retrieved code set, offering marginal recall improvements while introducing many irrelevant codes, leading to a notable precision trade-off.

5.3 Step 2: Knowledge Graph Retrieval and Validation

This step validates the candidate codes (output of step 1) with a KG that captures their relationships. We construct the KG by parsing the ICD-10-CM tabular list. We denote the complete KG as $G = (V, E)$, where:

- V is the set of nodes, consisting of all ICD-10-CM codes and instructional notes (e.g. synonym terms for codes).
- E is the set of edges, indicating the hierarchical and instructional properties between the nodes. For example, [I12.9 $\rightarrow$ is_ancestor $\rightarrow$ I12] and [I12.9 $\rightarrow$ inclusion_terms $\rightarrow$ "Hypertensive chronic kidney disease NOS, Hypertensive renal disease NOS"].

Validation begins by retrieving a subgraph $G_S = (V_S, E_S) \subseteq G$, where V_S contains the candidate codes, their ancestors, and associated instructional notes, and E_S contains the edges connected to V_S. Our ***KG-Based Auditor*** agent then validates the

candidates using the rich context in G_S, resolving issues such as incorrect codes (e.g., codes whose descriptions contradict the clinical note), conflicting codes (e.g., codes that mutually exclude each other), and missing codes (e.g., complementary codes that must be assigned together).

5.4 Step 3: Guideline Retrieval and Summarisation

This step retrieves and summarises relevant coding guidelines for the validated codes (output of step 2). Our ***Guideline Summariser*** agent retrieves related guideline sections for each code by navigating the guidelines' table of contents. For example, for code I27.20 (Pulmonary hypertension, unspecified), we retrieve the general coding guidelines on "unspecified" codes and the chapter-specific guidelines for Chapter 9 (Diseases of the Circulatory System). An illustration of this retrieval process is shown in Figure 2.

Although using the table of contents to retrieve chapter-specific guidelines may appear redundant, it ensures consideration of all relevant rules, as a code may relate to others in different chapters. For example, elevated blood pressure (R03.0) can indicate underlying hypertension (I10-I16), and guidelines in Section I.C.9 (for codes I00-I99) explicitly prohibit coding R03.0 when hypertension is definitively diagnosed. Since multiple guideline sections may be retrieved, and some chapter-specific sections span over 10 pages, filtering for relevant information is essential. To achieve this, our ***guideline summariser*** agent summarises the retrieved guidelines into bullet points containing only the rules applicable to the code being validated.

5.5 Step 4: Final Validation Using Guideline Summaries

This step further verifies the validated codes (output of step 2) using guideline summaries (output of step 3). It ensures codes are consistent with G_S and clinical guidelines. Our ***Guideline-Based Auditor*** agent cross-references the input medical note and guideline summaries, deciding to retain, remove, or replace each code with an alternative. When no guideline summary is available for a code, the default action is to retain it, as it was derived from G_S and there is no evidence from official guidelines for its removal or replacement.

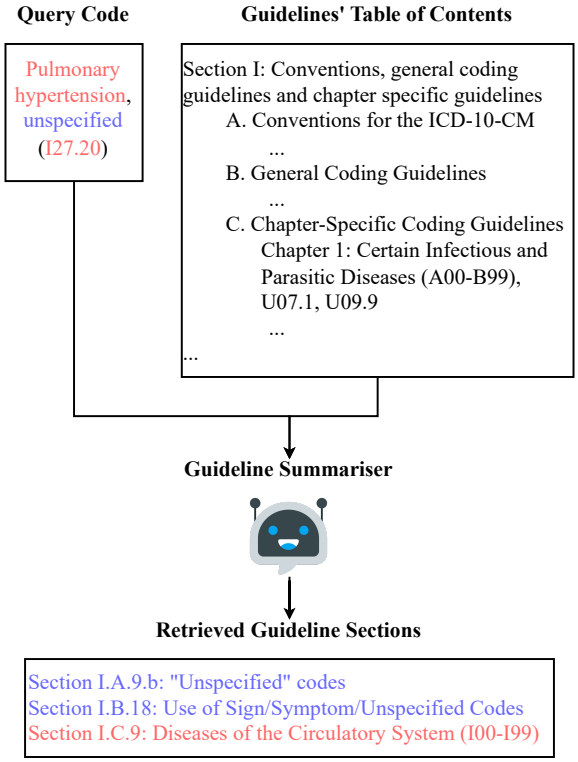

Figure 2: Illustration of our ***guideline summariser*** agent's retrieval process for an example code I27.20 (pulmonary hypertension, unspecified). The agent navigates the guidelines' table of contents to retrieve the relevant general and chapter-specific guidelines applicable to I27.20.

6 Experimental Setup

6.1 Dataset

We evaluate RAG-Coding on MDACE and MDACE-2025, which are introduced in Sections 2.3 and 4, respectively. We choose MDACE for three reasons: (1) paired with MDACE-2025, it allows us to assess generalisation to evolving clinical standards; (2) it provides high-quality annotations with high inter-annotator agreement; and (3) it offers standardised data splits, ensuring comparability with published works and reproducibility of our work. While both datasets contain ICD-10-PCS annotations, we evaluate only on ICD-10-CM codes, as the LLM-based baselines compared in this work focus exclusively on ICD-10-CM.

6.2 Metrics

Each encounter consists of one or more medical notes (e.g., discharge summary, radiology report), which we process independently during inference. During evaluation, predictions and gold-standard codes are merged at the encounter level: for note

x_i, we denote the gold-standard codes as y_i and predicted codes as $\hat{y}_i$, aggregating across all notes in the same patient encounter c_j:

$$y_j = \bigcup_{x_i \in c_j} y_i, \quad \hat{y}_j = \bigcup_{x_i \in c_j} \hat{y}_i. \tag{1}$$

This *encounter-level* evaluation reflects real-world practice (Cheng et al., 2023), in which coders review all notes from an encounter when assigning codes[3]. Following prior work (Boyle et al., 2023; Li et al., 2024; Baksi et al., 2025; Motzfeldt et al., 2025), we report micro- and macro-averaged precision, recall, and F1 on the test set. Micro-averaging aggregates counts across all code occurrences in all encounters; macro-averaging computes the metric separately for each unique code across all encounters and then averages them equally.

6.3 Implementation Details

We experiment with five different LLM backbones for our RAG-Coding method: Deepseek-V3 (Liu et al., 2025), Qwen3 (Yang et al., 2025), GPT-4o (OpenAI, 2024), GPT-4.1 (OpenAI, 2025b), and GPT-5 with low reasoning mode (OpenAI, 2025a). All LLMs are hosted locally or privately via the Azure OpenAI service to follow PhysioNet's data use agreement[4]. Exact model checkpoints and coding guideline versions are detailed in Appendix D.

6.4 Baselines

Our RAG-Coding method is compared against Tree Search (Boyle et al., 2023), MedCodER (Baksi et al., 2025), CLH (Motzfeldt et al., 2025), and PLM-ICD (Huang et al., 2022). Full descriptions of these baselines are provided in Appendix E. All LLM-based baselines (Tree Search, MedCodER, and CLH) are evaluated using the same setup as in Section 6.3 (i.e., same LLM backbones, hyperparameters, and external coding knowledge sources). CLH is the only existing baseline that uses coding guidelines; it differs from RAG-Coding primarily in how external knowledge is integrated—plain instructional notes vs. a KG, and plain guideline text vs. code-specific summaries. PLM-ICD is the SOTA supervised baseline, trained on ICD-10-CM codes from MIMIC-IV and the MDACE training set (≈16K unique codes, covering ≈20% of the full ICD-10-CM code space).

[3]We followed the same practice when annotating MDACE-2025.

[4]https://physionet.org/news/post/gpt-responsible-use

7 Results

The main result values reported in this section represent means over three runs; standard deviations (typically ≤ 0.03) are available in Appendix F.

7.1 Main Results on MDACE

Table 1 compares RAG-Coding against LLM-based baselines on the MDACE test set. Using uniform LLM backbones, RAG-Coding outperforms the best baseline by 3–13% in micro-F1 and 1–8% in macro-F1. With GPT-5 (low reasoning mode), RAG-Coding achieves a micro-F1 of 0.54 and macro-F1 of 0.32, outperforming the best baseline (CLH) by 8% in both micro- and macro-F1 over the full ICD-10-CM code space (≈70K codes). These results validate our hypothesis that integrating RAG with structured in-domain knowledge enhances LLMs' medical coding performance.

Table 2 compares RAG-Coding against PLM-ICD, the SOTA model for automated coding. RAG-Coding achieves higher micro recall (+11%), while PLM-ICD achieves higher micro precision (+6%). Overall, the two methods are on par for common codes (as shown by micro-F1 scores), but RAG-Coding slightly outperforms PLM-ICD on rare codes (as shown by macro-F1 scores), suggesting an advantage on rare codes.

We agree with Motzfeldt et al. (2025) in attributing PLM-ICD's strong performance on common codes to its supervised training. PLM-ICD is trained on data from the same intensive care unit environment as the MDACE test set. This close match between training and test distributions allows PLM-ICD to excel at identifying recurring patterns in common codes. RAG-Coding is training-free and does not benefit from such distributional similarity during inference. Moreover, PLM-ICD uses a constrained decoding space of ≈16K codes, all unique ICD-10-CM codes present in the MIMIC-IV and MDACE training sets, while RAG-Coding considers the full (≈70K) ICD-10-CM code space. For a fairer comparison, we apply a post-hoc filter restricting RAG-Coding's predictions to those 16K codes. While not a perfectly apples-to-apples comparison—RAG-Coding still reasons over the larger space initially—it narrows the inference space gap between the two methods.

7.2 Main Results on MDACE-2025

The main difference between MDACE and MDACE-2025 is the latter's expanded code an-

LLM	Method	Micro			Macro		
		Precision	Recall	F1	Precision	Recall	F1
Qwen3	Tree Search	0.08	0.17	0.11	0.05	0.05	0.04
	MedCodER	0.29	0.36	0.32	0.16	0.16	0.15
	CLH	0.27	**0.49**	0.35	0.17	**0.18**	0.17
	RAG-Coding (Ours)	**0.43**	0.39	**0.41**	**0.20**	**0.18**	**0.18**
Deepseek-V3	Tree Search	0.12	0.25	0.16	0.07	0.08	0.07
	MedCodER	0.33	**0.48**	0.39	0.18	0.19	0.17
	CLH	0.37	0.46	0.41	0.21	0.21	0.21
	RAG-Coding (Ours)	**0.46**	0.43	**0.44**	**0.25**	**0.22**	**0.22**
GPT-4o	Tree Search	0.14	0.26	0.18	0.09	0.09	0.09
	MedCodER	0.39	0.36	0.37	0.19	0.18	0.17
	CLH	0.29	0.47	0.36	0.18	0.19	0.18
	RAG-Coding (Ours)	**0.42**	**0.48**	**0.45**	**0.21**	**0.22**	**0.21**
GPT-4.1	Tree Search	0.11	0.22	0.15	0.08	0.08	0.07
	MedCodER	0.31	0.44	0.36	0.17	0.18	0.17
	CLH	0.32	0.52	0.40	0.20	0.22	0.20
	RAG-Coding (Ours)	**0.47**	**0.59**	**0.53**	**0.27**	**0.29**	**0.27**
GPT-5	Tree Search	0.17	0.33	0.22	0.10	0.11	0.09
	MedCodER	0.37	0.42	0.40	0.21	0.22	0.20
	CLH	0.40	0.55	0.46	0.24	0.26	0.24
	RAG-Coding (Ours)	**0.46**	**0.66**	**0.54**	**0.31**	**0.33**	**0.32**

Table 1: Results on the *MDACE* test set. All methods use the entire code space during inference ($\approx$ 70K codes).

LLM	Method	Micro			Macro		
		Precision	Recall	F1	Precision	Recall	F1
-	PLM-ICD	**0.54**	0.53	0.53	0.29	0.30	0.30
Qwen3	RAG-Coding (Ours)	0.47	0.39	0.43	0.21	0.20	0.19
Deepseek-V3	RAG-Coding (Ours)	0.50	0.43	0.46	0.27	0.24	0.24
GPT-4o	RAG-Coding (Ours)	0.44	0.48	0.46	0.21	0.22	0.21
GPT-4.1	RAG-Coding (Ours)	0.51	0.53	0.52	0.26	0.27	0.25
GPT-5	RAG-Coding (Ours)	0.48	**0.64**	**0.55**	**0.30**	**0.32**	**0.31**

Table 2: Results on the *MDACE* test set. PLM-ICD covers only 20% of the ICD-10-CM code space ($\approx$ 16K codes) during inference. In contrast, RAG-Coding covers the entire code space ($\approx$ 70K codes) but applies post-processing to filter out any codes outside the 16K space used by PLM-ICD to make it a direct comparison.

notations, especially for R (symptom-related) and Z (health status-related) codes, as shown in Appendix C. Table 3 compares RAG-Coding against baselines on MDACE-2025. Relative to their performance on the original MDACE test set, all methods show higher precision but proportionally lower recall, yielding comparable F1 scores across the two datasets. The precision increase suggests that the baselines and RAG-Coding correctly identified some newly added codes, which were previously penalised as false positives due to their absence in MDACE annotations. However, the expanded annotations also make high recall harder to achieve.

Table 3 demonstrates that RAG-Coding, with GPT-5 as the backbone, achieves an 8% advantage in both micro- and macro-F1 over PLM-ICD. As noted in prior work (Edin et al., 2023), PLM-ICD often struggles to differentiate between similar Z codes in imbalanced settings. This limitation is particularly visible in MDACE-2025, which provides a more comprehensive annotation of Z codes. Moreover, the evaluated PLM-ICD model was trained on older MIMIC-IV and MDACE data, limiting its adaptability to ground-truth shifts from updated guidelines. In contrast, RAG-Coding uses the same up-to-date tabular list and guidelines as those used for MDACE-2025 annotation, yielding better conformance and generalisation to its ground truth.

7.3 Ablation Study and Analysis

Table 4 shows stepwise gains confirming the contribution of each RAG-Coding component, highlighting the importance of structured external knowledge for LLM-based medical coding. To further disentangle the importance of external knowledge, we ran a closed-book variant of RAG-Coding: after

#C	Method	Micro			Macro		
		Precision	Recall	F1	Precision	Recall	F1
70K	Tree Search	0.20	0.28	0.24	0.12	0.13	0.12
	MedCodER	0.43	0.35	0.38	0.24	0.23	0.23
	CLH	0.45	0.44	0.45	0.26	0.27	0.26
	RAG-Coding (Ours)	**0.57**	**0.55**	**0.56**	**0.34**	**0.34**	**0.33**
16K	PLM-ICD	**0.62**	0.39	0.48	0.27	0.25	0.26
	RAG-Coding (Ours)	0.58	**0.55**	**0.56**	**0.35**	**0.34**	**0.34**

Table 3: Results on the *MDACE-2025* test set. "#C" refers to the number of candidate codes used during inference. All LLM-based methods are evaluated with GPT-5 (low reasoning mode); full results with other LLM backbones are available in Appendix F.

	Micro		
	Precision	Recall	F1
RAG-Coding (step 1)	0.34	**0.54**	0.41
RAG-Coding (steps 1-2)	0.42	0.53	0.47
RAG-Coding (steps 1-4)	**0.48**	**0.54**	**0.51**

Table 4: Ablation study on the *MDACE* test set (GPT-4.1 as the backbone LLM). Step 3 (retrieval and summarisation of guidelines) provides complementary information to step 4 and has no measurable impact on the metrics. Therefore, results for steps 1-3 are omitted, and the full RAG-Coding pipeline is reported as steps 1-4.

generating candidate ICD codes (step 1), the same LLM reviews and validates them using only its parametric knowledge. As detailed in Appendix G, this experiment reveals that LLMs cannot effectively correct their own coding errors without external knowledge.

We also manually examined intermediate outputs for 20 randomly selected test samples, categorising changes at each RAG-Coding step (GPT-4.1 as the LLM backbone)[5].

Candidate Generator (step 1) achieves moderate recall (0.41) but low precision (0.27). False positives concentrate in symptom and injury sub-codes; recall errors concentrate in Z-codes and circulatory codes.

KG-Based Auditor (step 2) removes 2.2 codes per sample on average, with 91% removal accuracy (39 of 43). It correctly removes codes whose descriptions contradict the medical note (e.g., A41.53 *Sepsis due to Serratia* when the note documents Klebsiella) and symptom codes rendered redundant by more specific diagnoses. Its primary failure is removing briefly mentioned but valid comorbidities, where the KG provides no conflicting signal, suggesting the removal is likely due to insufficient contextual density.

Guideline Summariser (step 3) retrieves summaries for 90% (115 of 128) of codes. Gaps are mainly in circulatory and respiratory codes, where rules are defined only at the chapter level rather than per code. When no guideline is found, the code is retained by default in step 4.

Guideline-Based Auditor (step 4) achieves 89% removal accuracy (17 of 19). Code replacements are its most valuable corrections (e.g., I10 *Hypertension* replaced by I12.9 *Hypertensive chronic kidney disease* when chronic kidney disease is co-documented). However, addition accuracy is poor (2 of 10): the agent over-interprets "prefer specificity" guidelines, adding specific codes without sufficient supporting evidence in the note.

8 Conclusion and Future Work

We propose RAG-Coding, an agentic, training-free method for automated medical coding, and introduce MDACE-2025, a new dataset with three improvements over MDACE: ICD-10-CM/PCS annotations under the 2025 ICD-10-CM guidelines, code sequencing, and justification comments. RAG-Coding grounds LLM coding decisions in structurally encoded external knowledge, yielding substantial precision and F1 gains over recent LLM-based baselines. Compared to PLM-ICD, it excels on rare codes, achieves higher recall, and generalises across the full ICD-10-CM code space. On MDACE-2025, RAG-Coding outperforms all baselines, demonstrating effective generalisation to updated coding guidelines. Ablations confirm that both the KG and guideline summaries are critical components. In future work, we plan to expand the annotation scope of MDACE-2025 and to extend RAG-Coding to coding systems beyond ICD-10-CM and to other languages.

[5]Due to space constraints, the full analysis is available in Appendix H; we plan to promote it to the main text in the camera-ready version.

Limitations

A main limitation of this paper is its exclusive focus on the ICD-10-CM coding ontology for evaluation. This focus may limit the generalisation of our findings to other ontologies, such as ICD-10-PCS, CPT, or ICD-10-AM, which exhibit different ontology structures and complexities that could affect RAG-Coding's performance. Although both MDACE and MDACE-2025 contain ICD-10-PCS annotations, we evaluate only on ICD-10-CM codes, as the LLM-based baselines compared in this work focus exclusively on ICD-10-CM. While adapting RAG-Coding and some of the baselines to ICD-10-PCS is possible, it requires substantial effort and presents an exciting opportunity for future research.

Further, like most existing studies in this area, our work focuses only on English-language medical notes. Given that medical coding is a global task involving diverse linguistic contexts, this English-centric approach may limit the applicability of RAG-Coding to non-English settings, where variations in terminology, syntax, and cultural nuances could introduce challenges.

References

Vera Alonso, João Vasco Santos, Marta Pinto, Joana Ferreira, Isabel Lema, Fernando Lopes, and Alberto Freitas. 2020. Problems and barriers during the process of clinical coding: a focus group study of coders' perceptions. *Journal of medical systems*, 44:1–8.

Krishanu Das Baksi, Elijah Soba, John J Higgins, Ravi Saini, Jaden Wood, Jane Cook, Jack I Scott, Nirmala Pudota, Tim Weninger, Edward Bowen, and Sanmitra Bhattacharya. 2025. MedCodER: A generative AI assistant for medical coding. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: Industry Track)*, pages 449–459, Albuquerque, New Mexico. Association for Computational Linguistics.

Joseph Boyle, Antanas Kascenas, Pat Lok, Maria Liakata, and Alison O'Neil. 2023. Automated clinical coding using off-the-shelf large language models. In *Deep Generative Models for Health Workshop NeurIPS 2023*.

Pengfei Cao, Yubo Chen, Kang Liu, Jun Zhao, Shengping Liu, and Weifeng Chong. 2020. HyperCore: Hyperbolic and co-graph representation for automatic ICD coding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3105–3114, Online. Association for Computational Linguistics.

Hua Cheng, Rana Jafari, April Russell, Russell Klopfer, Edmond Lu, Benjamin Striner, and Matthew 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, Toronto, Canada. Association for Computational Linguistics.

Jou-An Chi and Shu-Kai Hsieh. 2025. Structured vs. unstructured inputs in LLMs: Evaluating the semantic and pragmatic predictive power in abnormal event forecasting. In *Proceedings of the 37th Conference on Computational Linguistics and Speech Processing (ROCLING 2025)*, pages 237–248, National Taiwan University, Taipei City, Taiwan. Association for Computational Linguistics.

CMS. 2026. Icd-10 coordination and maintenance committee meetings. https://www.cms.gov/medicare/coding-billing/icd-10-codes/icd-10-coordination-maintenance-committee-meetings. [Online; accessed 24-May-2026].

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Hang Dong, Matúš Falis, William Whiteley, Beatrice Alex, Joshua Matterson, Shaoxiong Ji, Jiaoyan Chen, and Honghan Wu. 2022. Automated clinical coding: what, why, and where we are? *NPJ digital medicine*, 5(1):159.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2025. The faiss library. *IEEE Transactions on Big Data*.

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.

Matúš Falis, Aryo Pradipta Gema, Hang Dong, Luke Daines, Siddharth Basetti, Michael Holder, Rose S Penfold, Alexandra Birch, and Beatrice Alex. 2024. Can gpt-3.5 generate and code discharge summaries? *Journal of the American Medical Informatics Association*, 31(10):2284–2293.

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

Kin Wah Fung, Julia Xu, S Trent Rosenbloom, and James R Campbell. 2019. Using snomed ct-encoded problems to improve icd-10-cm coding—a randomized controlled experiment. *International journal of medical informatics*, 126:19–25.

Yidong Gan, Maciej Rybinski, Ben Hachey, and Jonathan K. Kummerfeld. 2025. Aligning AI research with the needs of clinical coding workflows: Eight recommendations based on US data analysis and critical review. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 909–922, Vienna, Austria. Association for Computational Linguistics.

Chao-Wei Huang, Shang-Chi Tsai, and Yun-Nung Chen. 2022. PLM-ICD: Automatic ICD coding with pretrained language models. In *Proceedings of the 4th Clinical Natural Language Processing Workshop*, pages 10–20, Seattle, WA. Association for Computational Linguistics.

Sirui Huang, Yanggan Gu, Zhonghao Li, Xuming Hu, Li Qing, and Guandong Xu. 2025. StructFact: Reasoning factual knowledge from structured data with large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 7521–7552, Vienna, Austria. Association for Computational Linguistics.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023. StructGPT: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251, Singapore. Association for Computational Linguistics.

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.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

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

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, and 1 others. 2025. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Andreas Geert 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, Suzhou, China. 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, New Orleans, Louisiana. Association for Computational Linguistics.

Alexandr Nesterov, Andrey Sakhovskiy, Ivan Sviridov, Airat Valiev, Vladimir Makharev, Petr Anokhin, Galina Zubkova, and Elena Tutubalina. 2025. RuCCoD: Towards automated ICD coding in Russian. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 2558–2585, Suzhou, China. Association for Computational Linguistics.

OpenAI. 2024. Hello gpt-4o. `https://openai.com/index/hello-gpt-4o`. [Online; accessed 1-Dec-2025].

OpenAI. 2025a. Gpt-5 is here. `https://openai.com/gpt-5`. [Online; accessed 12-Oct-2025].

OpenAI. 2025b. Introducing gpt-4.1 in the api. `https://openai.com/index/gpt-4-1/`. [Online; accessed 1-Dec-2025].

OpenAI. 2025c. text-embedding-3-large. `https://platform.openai.com/docs/models/text-embedding-3-large`. [Online; accessed 1-Dec-2025].

Anthony Rios and Ramakanth Kavuluru. 2018. Few-shot and zero-shot multi-label learning for structured label spaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3132–3142, Brussels, Belgium. Association for Computational Linguistics.

Yuanzhen Xie, Xinzhou Jin, Tao Xie, Mingxiong Lin, Liang Chen, Chenyun Yu, Lei Cheng, Chengxiang Zhuo, Bo Hu, and Zang Li. 2024. Decomposition for enhancing attention: Improving LLM-based text-to-SQL through workflow paradigm. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10796–10816, Bangkok, Thailand. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Zhichao Yang, Sunjae Kwon, Zonghai Yao, and Hong Yu. 2023. Multi-label few-shot icd coding as autoregressive generation with prompt. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 5366–5374.

Zheng Yuan, Chuanqi Tan, and Songfang Huang. 2022. Code synonyms do matter: Multiple synonyms matching network for automatic ICD coding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 808–814, Dublin, Ireland. Association for Computational Linguistics.

A Prompts for RAG-Coding Agents

A.1 Candidate Generator

```
You are an expert clinical coder.

Instructions:
Your task is to output all relevant ICD-10-CM codes that are relevant to the medical note.
Each code must be supported by one or more verbatim evidence taken directly from the medical note (no paraphrasing, interpretation, or rewriting).
Do not infer or code any condition not explicitly documented. For example, if the note mentions 'smokes 1 pack per day for many years.', use F17.20 (nicotine dependence, unspecified), not F17.210 (nicotine dependence, cigarettes) as the type of tobacco product is not clearly documented.

JSON Output Format (you must strictly follow this format and output nothing else):
{
    "results": [
        {
            "code": "<alphanumeric ICD-10-CM code>",
            "description": "<official description of the code>",
            "evidence": [
                "<text snippet that directly supports the code>",
                (continue for all relevant evidence)
            ],
        },
        (continue for all relevant ICD-10-CM codes)
    ]
}
```

A.2 KG-Based Auditor

```
You are an expert medical coder.

Instruction:
Select the most appropriate ICD-10-CM codes from the provided list based on the patient's note.
Your reasoning must be grounded in the provided knowledge graph.

Input Format:
<Note> ... </Note> - The patient's medical note.
<Codes> ... </Codes> - A comma-separated list of potential ICD-10-CM codes.
<KnowledgeGraph> ... </KnowledgeGraph> - A structured list of relationships providing additional context.

Knowledge Graph Format:
The knowledge graph consists of triplets in the format [subject, predicate, object], where:
Subject: An ICD-10-CM code or a range (e.g. I10, I10-I16).
Predicate: The type of relationship, such as:
    "description" - textual explanation of the code.
    "inclusion_term" - synonymous or included terms.
    "use_additional_code" - indicates codes that must also be used.
    "code_first" - indicates the code that should be listed first.
    "ancestor" - links to a parent or higher-level code.
    "excludes1" - codes that must not be used together.
    "includes" - conditions or terms encompassed by the code.
Object: The value or linked entity (e.g. text, Boolean, or another ICD code).

Output Format (you must strictly follow this JSON format and output nothing else):
{
    "results": [
        {
            "code": "<alphanumeric ICD-10-CM code>",
            "justification": "<brief justification for assigning this code>"
        },
        (continue for all selected/assigned ICD-10-CM codes)
    ]
}
```

A.3 Guideline-Based Auditor

```
You are an expert medical coder.

Your task is to audit a provided medical note against a list of assigned ICD-10-CM codes using official coding guidelines. You *must not* use any abbreviations in your response.

**Auditing Process**
For each code, use this format:
Code: [The code to audit]
Guidelines: [Must be fetched via the get_relevant_summaries_for_code() tool]
Thought: [If guidelines found, reason about whether the code should be retained, removed, or replaced (by an alternative code). If guidelines not found, **retain the code and move to the next code**]
Decision: [Whether to retain/remove/replace the code]
... (this Code/Guidelines/Thought/Decision should repeat N times, where N is the number of codes)

**Output Format**
Output only the refined codes in this exact format (nothing else):
Final Answer: Code1, Code2, Code3
```

B Dynamic Few-Shot for Candidate Generator

We use FAISS (Douze et al., 2025) as the vector store. All notes in the MDACE training set

Few-Shot Size	Micro		
	Precision	Recall	F1
$k = 0$	0.30	**0.53**	0.38
$k = 1$	0.36	0.51	0.42
$k = 2$	**0.40**	0.51	**0.44**
$k = 3$	0.38	0.50	0.43

Table 5: Results of the ***Candidate Generator*** agent using different numbers of dynamic few-shot examples on the *MDACE* validation set.

are indexed using text-embedding-3-large (OpenAI, 2025c) as the embedding model. For each inference instance (medical note), we compute its embedding using the same embedding model and retrieve the top k most similar indexed samples. These retrieved samples serve as in-context examples for the ***candidate generator*** (step 1). As shown in Table 5, we experimented with $k \in \{0, 1, 2, 3\}$ and found that $k = 2$ yields the best F1 score on the MDACE validation set.

C Comparison Between MDACE and MDACE-2025

The test split shared by MDACE and MDACE-2025 contains 61 encounters and 116 notes. We quantify code annotation shifts using Krippendorff's α, yielding α values of 0.67 for diagnosis codes, 0.41 for procedure codes, and 0.66 overall. Discrepancies arise primarily from guideline updates, additions of supplementary Z codes (e.g. for medical history and allergies), and stricter procedure code restrictions to those supported by operative notes. Figure 3 shows the distribution of ICD-10-CM codes by chapter (i.e. first character) in the test sets of MDACE and MDACE-2025; Table 6 presents an overview of the statistical differences between them.

D LLM Checkpoints and Coding Guideline Versions

The LLM checkpoints used in this study are: deepseek-V3.2, qwen3-235b-a22b-instruct-2507, gpt-4o-2024-05-13, gpt-4.1-2025-04-14, and gpt-5-2025-08-07. We empirically set the model temperature to 0.2.

For experiments on the MDACE dataset, we use the April 1st 2022, coding guidelines and tabular list; this version is chosen over the original 2021 materials because the latter contains an incomplete embedded table of contents. For the MDACE-2025 dataset, we use the April 1st, 2025 version of the guidelines and tabular list.

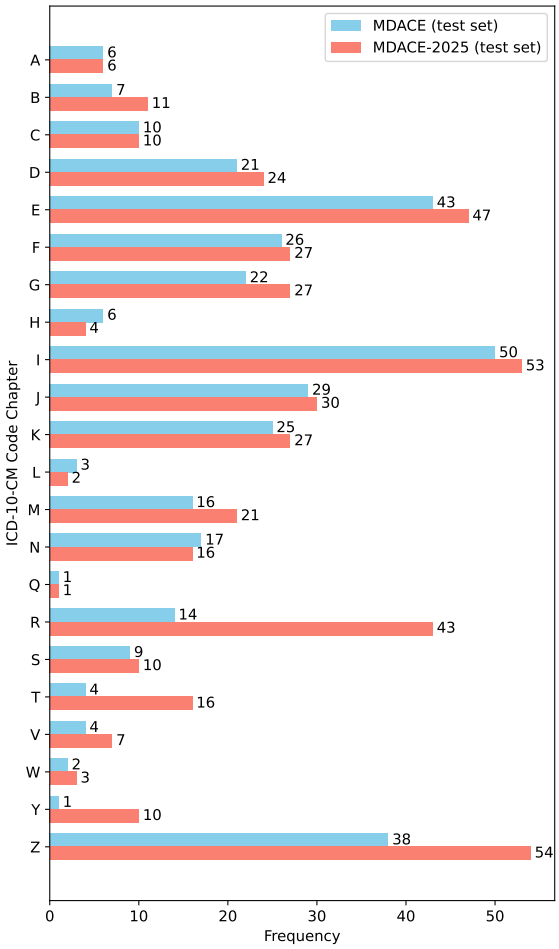

Figure 3: Distribution of ICD-10-CM codes by chapter.

E Description of Baselines

RAG-Coding is compared against these baselines:

- **Tree Search** (Boyle et al., 2023) introduces a LLM-based tree search algorithm, navigating the ICD ontology recursively to predict codes level by level.
- **MedCodER** (Baksi et al., 2025) uses an LLM to extract diagnoses and generate ICD codes. It retrieves additional codes via semantic search on the extracted diagnoses and indexed code descriptions, then re-ranks the combined set of generated and retrieved codes.
- **CLH** (Motzfeldt et al., 2025) is an agentic method simulating human coding workflows. It identifies codable text spans, retrieves candidate codes via semantic search against the

Dataset	# of unique ICD-10-CM diagnosis codes	# of unique ICD-10-PCS procedure codes	# of average codes per encounter	% of encounters with justifications
MDACE	320	43	11.11	-
MDACE-2025	451	16	14.84	11.48

Table 6: Dataset statistics on the test split shared by MDACE and MDACE-2025. Key shifts include expanded diagnosis annotations, a reduction in procedure annotations attributable to guideline changes, and the addition of expert justification comments for complex coding decisions.

alphabetical index[6], and validates them using coding guidelines and the tabular list.

- **PLM-ICD** (Huang et al., 2022) is not LLM-based and is the SOTA method for automated coding (Edin et al., 2023; Gan et al., 2025; Motzfeldt et al., 2025). PLM-ICD uses a BERT-based encoder (Devlin et al., 2019) pre-trained on biomedical texts and is fine-tuned with label-wise attention.

Tree Search has an additional parameter defining the maximum number of iterations; we set it to 50, the optimal value reported in the original paper. Similarly, MedCodER has an additional parameter defining the number of retrieved codes per diagnosis; we set it to 1, the optimal value reported in the original paper. PLM-ICD is trained on ICD-10-CM codes from the entire MIMIC-IV dataset and the MDACE training set, where the number of unique codes is approx. 16K. This means PLM-ICD covers only about 20% of the ICD-10-CM code space during inference, while LLM-based methods use the entire space (≈70K).

F Full Results

The full version of the main experiment results on the MDACE and MDACE-2025 test sets are presented in Table 7, Table 8, Table 9, and Table 10.

G Code Auditing with LLMs' Internal Knowledge

The ablation results in Table 11 show that LLMs cannot effectively correct coding errors using only their internal knowledge. With GPT-4.1, we implemented a prompt-based self-correction mechanism, representing a naïve version of SELF-REFINE (Madaan et al., 2023) without external feedback. When given code predictions from the ***Candidate Generator*** (step 1), the LLM failed to detect false positives and false negatives reliably.

H Detailed Analysis of Individual Components

We also manually examined intermediate outputs for 20 randomly selected test samples, categorising changes at each RAG-Coding step (GPT-4.1 as the LLM backbone).

H.1 Candidate Generator (step 1)

Candidate Generator produces an average of 8.6 candidate codes per sample against 8.0 gold diagnosis codes, achieving a precision of 0.27 and recall of 0.41 on this subset of 20 samples. False positives are heavily skewed by code category. Symptom and sign codes (Chapter R) have a 96% false positive rate (23 false positives, 1 true positive): these symptoms and signs are documented but should not be assigned because a specific diagnosis is already present. Injury codes (Chapters S–T) show an 89% false positive rate (17 false positives, 2 true positives), with errors mainly occurring at the fourth-digit fine-grained level. Recall errors are most prevalent in Z-codes (14 missed), which encode administrative context such as personal and family history, and in circulatory codes (11 missed), suggesting ***Candidate Generator*** underweights contextual comorbidities.

H.2 KG-based Auditor (step 2)

KG-based Auditor acts as a conservative pruner: it removes an average of 2.2 codes per sample (from 8.6 to 6.4) and adds zero new codes across all 20 samples. Of the 43 total removals, 39 (91%) are correct, and 4 (9%) are incorrect removals of true target codes.

The first four categories in Table 12 reflect four KG-driven correction mechanisms:

- **Symptom subsumed by diagnosis:** symptom and sign codes (e.g., R55 *Syncope*,

[6]The alphabetical index lists terms alphabetically to locate ICD-10-CM codes. Multiple terms describing the same condition may map to one code.

LLM	Method	Micro			Macro		
		Precision	Recall	F1	Precision	Recall	F1
Qwen3	Tree Search	0.08 ± 0.01	0.17 ± 0.00	0.11 ± 0.01	0.05 ± 0.01	0.05 ± 0.00	0.04 ± 0.01
	MedCodER	0.29 ± 0.02	0.36 ± 0.03	0.32 ± 0.02	0.16 ± 0.01	0.16 ± 0.01	0.15 ± 0.01
	CLH	0.27 ± 0.03	**0.49 ± 0.03**	0.35 ± 0.03	0.17 ± 0.02	**0.18 ± 0.01**	0.17 ± 0.01
	RAG-Coding	**0.43 ± 0.02**	0.39 ± 0.01	**0.41 ± 0.02**	**0.20 ± 0.01**	**0.18 ± 0.01**	**0.18 ± 0.01**
Deepseek-V3	Tree Search	0.12 ± 0.01	0.25 ± 0.02	0.16 ± 0.02	0.07 ± 0.01	0.08 ± 0.01	0.07 ± 0.01
	MedCodER	0.33 ± 0.02	**0.48 ± 0.02**	0.39 ± 0.02	0.18 ± 0.01	0.19 ± 0.00	0.17 ± 0.01
	CLH	0.37 ± 0.04	0.46 ± 0.03	0.41 ± 0.03	0.21 ± 0.02	0.21 ± 0.01	0.21 ± 0.02
	RAG-Coding	**0.46 ± 0.03**	0.43 ± 0.02	**0.44 ± 0.03**	**0.25 ± 0.02**	**0.22 ± 0.02**	**0.22 ± 0.01**
GPT-4o	Tree Search	0.14 ± 0.01	0.26 ± 0.03	0.18 ± 0.02	0.09 ± 0.01	0.09 ± 0.01	0.09 ± 0.01
	MedCodER	0.39 ± 0.01	0.36 ± 0.01	0.37 ± 0.01	0.19 ± 0.01	0.18 ± 0.02	0.17 ± 0.01
	CLH	0.29 ± 0.02	0.47 ± 0.00	0.36 ± 0.02	0.18 ± 0.01	0.19 ± 0.01	0.18 ± 0.01
	RAG-Coding	**0.42 ± 0.01**	**0.48 ± 0.01**	**0.45 ± 0.01**	**0.21 ± 0.01**	**0.22 ± 0.01**	**0.21 ± 0.01**
GPT-4.1	Tree Search	0.11 ± 0.00	0.22 ± 0.01	0.15 ± 0.01	0.08 ± 0.01	0.08 ± 0.01	0.07 ± 0.01
	MedCodER	0.31 ± 0.01	0.44 ± 0.01	0.36 ± 0.01	0.17 ± 0.01	0.18 ± 0.01	0.17 ± 0.01
	CLH	0.32 ± 0.01	0.52 ± 0.01	0.40 ± 0.00	0.20 ± 0.00	0.22 ± 0.01	0.20 ± 0.01
	RAG-Coding	**0.47 ± 0.02**	**0.59 ± 0.05**	**0.53 ± 0.03**	**0.27 ± 0.03**	**0.29 ± 0.03**	**0.27 ± 0.03**
GPT-5	Tree Search	0.17 ± 0.01	0.33 ± 0.00	0.22 ± 0.00	0.10 ± 0.00	0.11 ± 0.00	0.09 ± 0.01
	MedCodER	0.37 ± 0.01	0.42 ± 0.01	0.40 ± 0.01	0.21 ± 0.01	0.22 ± 0.01	0.20 ± 0.01
	CLH	0.40 ± 0.01	0.55 ± 0.02	0.46 ± 0.02	0.24 ± 0.01	0.26 ± 0.01	0.24 ± 0.01
	RAG-Coding	**0.46 ± 0.01**	**0.66 ± 0.01**	**0.54 ± 0.01**	**0.31 ± 0.01**	**0.33 ± 0.01**	**0.32 ± 0.00**

Table 7: Extended version of Table 1 reporting mean ± standard deviation over three runs on the *MDACE* test set.

LLM	Method	Micro			Macro		
		Precision	Recall	F1	Precision	Recall	F1
-	PLM-ICD	**0.54 ± 0.02**	0.53 ± 0.02	0.53 ± 0.02	0.29 ± 0.02	0.30 ± 0.01	0.30 ± 0.02
Qwen3	RAG-Coding	0.47 ± 0.02	0.39 ± 0.02	0.43 ± 0.02	0.21 ± 0.01	0.20 ± 0.01	0.19 ± 0.02
Deepseek-V3	RAG-Coding	0.50 ± 0.03	0.43 ± 0.03	0.46 ± 0.03	0.27 ± 0.02	0.24 ± 0.01	0.24 ± 0.01
GPT-4o	RAG-Coding	0.44 ± 0.01	0.48 ± 0.01	0.46 ± 0.01	0.21 ± 0.01	0.22 ± 0.01	0.21 ± 0.01
GPT-4.1	RAG-Coding	0.51 ± 0.04	0.53 ± 0.01	0.52 ± 0.02	0.26 ± 0.01	0.27 ± 0.01	0.25 ± 0.01
GPT-5	RAG-Coding	0.48 ± 0.03	**0.64 ± 0.03**	**0.55 ± 0.01**	**0.30 ± 0.02**	**0.32 ± 0.02**	**0.31 ± 0.02**

Table 8: Extended version of Table 2 reporting mean ± standard deviation over three runs on the *MDACE* test set.

R57.9 *Shock, unspecified*) are correctly removed when hierarchical KG edges reveal that a co-assigned diagnosis already subsumes them.

- **Wrong sub-code (laterality/specificity):** inclusion-term and ancestor edges reveal laterality or sub-type errors (e.g., a lower-extremity arterial thrombosis code selected when the upper-extremity is documented).
- **Code description–note mismatch:** the KG code description directly contradicts the medical note, unambiguously flagging a wrong code (e.g., A41.53 *Sepsis due to Serratia* when the note documents Klebsiella).
- **Undocumented/inferred diagnosis:** codes inferred from lab values or imaging alone (e.g., D64.9 *Anaemia* from haemoglobin values; J18.9 *Pneumonia* from a CT infiltrate) are correctly removed because the code descriptions and their KG synonyms are not documented in the medical note.

The main systematic failure is *comorbidity over-pruning*: the four incorrect removals involve valid but briefly documented comorbidities: F17.200 *Nicotine dependence*, E11.9 *Type 2 diabetes*, F32.A *Depression*, and M19.90 *Osteoarthritis*. The KG provides no signal against these codes, yet ***KG-based Auditor*** removes them. We hypothesise this is due to insufficient contextual density: each comorbidity appears only once or twice in the note (typically in the past-medical-history section), which becomes harder to keep track of when the medical note is lengthy.

H.3 Guideline Summariser (step 3)

Guideline Summariser retrieves applicable guidelines for 90% of step 2 output codes (115 of 128).

LLM	Method	Micro			Macro		
		Precision	Recall	F1	Precision	Recall	F1
Qwen3	Tree Search	0.10 ± 0.01	0.15 ± 0.02	0.12 ± 0.01	0.06 ± 0.00	0.06 ± 0.00	0.05 ± 0.00
	MedCodER	0.32 ± 0.02	0.27 ± 0.02	0.29 ± 0.01	0.17 ± 0.01	0.15 ± 0.01	0.15 ± 0.01
	CLH	0.30 ± 0.02	**0.38 ± 0.02**	0.33 ± 0.02	0.18 ± 0.02	**0.18 ± 0.01**	**0.17 ± 0.01**
	RAG-Coding	**0.46 ± 0.03**	0.29 ± 0.02	**0.35 ± 0.02**	**0.19 ± 0.02**	0.17 ± 0.01	**0.17 ± 0.02**
Deepseek-V3	Tree Search	0.14 ± 0.01	0.22 ± 0.00	0.17 ± 0.00	0.09 ± 0.00	0.09 ± 0.00	0.08 ± 0.00
	MedCodER	0.34 ± 0.02	**0.35 ± 0.01**	0.34 ± 0.01	0.18 ± 0.01	0.18 ± 0.00	0.17 ± 0.01
	CLH	0.41 ± 0.02	**0.35 ± 0.03**	0.38 ± 0.02	0.20 ± 0.02	**0.21 ± 0.01**	0.20 ± 0.01
	RAG-Coding	**0.48 ± 0.03**	**0.35 ± 0.02**	**0.40 ± 0.02**	**0.23 ± 0.02**	0.20 ± 0.01	**0.21 ± 0.01**
GPT-4o	Tree Search	0.17 ± 0.01	0.22 ± 0.02	0.19 ± 0.02	0.11 ± 0.01	0.10 ± 0.01	0.10 ± 0.01
	MedCodER	0.42 ± 0.01	0.26 ± 0.01	0.32 ± 0.00	0.18 ± 0.01	0.16 ± 0.01	0.16 ± 0.01
	CLH	0.32 ± 0.03	**0.37 ± 0.01**	0.35 ± 0.02	0.18 ± 0.01	**0.19 ± 0.01**	0.18 ± 0.02
	RAG-Coding	**0.44 ± 0.02**	0.34 ± 0.01	**0.39 ± 0.02**	**0.19 ± 0.01**	**0.19 ± 0.01**	**0.19 ± 0.01**
GPT-4.1	Tree Search	0.14 ± 0.01	0.19 ± 0.01	0.16 ± 0.01	0.09 ± 0.01	0.09 ± 0.01	0.08 ± 0.01
	MedCodER	0.35 ± 0.01	0.33 ± 0.01	0.34 ± 0.01	0.18 ± 0.00	0.18 ± 0.00	0.17 ± 0.00
	CLH	0.37 ± 0.02	0.39 ± 0.01	0.38 ± 0.01	0.21 ± 0.01	0.22 ± 0.01	0.21 ± 0.02
	RAG-Coding	**0.53 ± 0.03**	**0.40 ± 0.01**	**0.46 ± 0.01**	**0.25 ± 0.01**	**0.23 ± 0.01**	**0.23 ± 0.01**
GPT-5	Tree Search	0.20 ± 0.00	0.28 ± 0.00	0.24 ± 0.01	0.12 ± 0.01	0.13 ± 0.01	0.12 ± 0.00
	MedCodER	0.43 ± 0.02	0.35 ± 0.01	0.38 ± 0.01	0.24 ± 0.01	0.23 ± 0.01	0.23 ± 0.01
	CLH	0.45 ± 0.02	0.44 ± 0.02	0.45 ± 0.02	0.26 ± 0.02	0.27 ± 0.03	0.26 ± 0.03
	RAG-Coding	**0.57 ± 0.02**	**0.55 ± 0.03**	**0.56 ± 0.02**	**0.34 ± 0.02**	**0.34 ± 0.02**	**0.33 ± 0.02**

Table 9: Extended version of Table 3, covering more LLM backbones and reporting mean ± standard deviation over three runs on the *MDACE-2025* test set.

LLM	Method	Micro			Macro		
		Precision	Recall	F1	Precision	Recall	F1
-	PLM-ICD	**0.62 ± 0.02**	0.39 ± 0.01	0.48 ± 0.02	0.27 ± 0.01	0.25 ± 0.01	0.26 ± 0.01
Qwen3	RAG-Coding	0.50 ± 0.02	0.29 ± 0.02	0.36 ± 0.02	0.20 ± 0.01	0.17 ± 0.01	0.18 ± 0.01
Deepseek-V3	RAG-Coding	0.52 ± 0.02	0.35 ± 0.02	0.42 ± 0.01	0.24 ± 0.01	0.20 ± 0.01	0.22 ± 0.01
GPT-4o	RAG-Coding	0.45 ± 0.02	0.34 ± 0.01	0.39 ± 0.01	0.19 ± 0.01	0.19 ± 0.01	0.18 ± 0.01
GPT-4.1	RAG-Coding	0.56 ± 0.03	0.40 ± 0.01	0.46 ± 0.01	0.25 ± 0.01	0.24 ± 0.00	0.23 ± 0.01
GPT-5	RAG-Coding	0.58 ± 0.01	**0.55 ± 0.03**	**0.56 ± 0.02**	**0.35 ± 0.02**	**0.34 ± 0.02**	**0.34 ± 0.02**

Table 10: Extended version of Table 3, covering more LLM backbones and reporting mean ± standard deviation over three runs on the *MDACE-2025* test set.

	Micro		
	Precision	Recall	F1
RAG-Coding (step 1)	0.34	**0.54**	**0.41**
with SC	**0.35**	0.53	**0.41**
with SC ×2	**0.35**	0.53	**0.41**

Table 11: Ablation study on the *MDACE* test set. We use GPT-4.1 as the backbone LLM. 'SC' denotes self-correction, where the LLM validates the candidate codes (output of step 1) using only its internal knowledge. 'SC × 2' applies self-correction twice.

Reason for Code Removal	Accuracy
Symptom subsumed by diagnosis	13/13 (100%)
Wrong sub-code (laterality/specificity)	13/13 (100%)
Code description–note mismatch	8/8 (100%)
Undocumented/inferred diagnosis	5/5 (100%)
Comorbidity over-pruning	0/4 (0%)
Total	39/43 (91%)

Table 12: Step 2 code removals categorised by reason. Accuracy is defined as the proportion of correct removals (i.e., the removed code was not a true target code).

The 10% gap is concentrated in circulatory and respiratory codes, where relevant rules are defined in chapter-level narrative rather than code-specific narrative. When no guideline is found, the code is retained by default in step 4.

H.4 Guideline-Based Auditor (step 4)

Guideline-Based Auditor makes 31 changes across the 20 samples (19 removals, 10 additions, and 2 replacements).

Correct changes. Removals are the most reliable action (89%; 17 of 19 correct): guidelines consistently lead to removal of codes unsupported by explicit documentation or made redundant by a co-assigned code. The two correct replacements represent the ***Guideline-Based Auditor***'s highest-value contribution. In one sample, the guideline for I10 prohibits its use when chronic kidney disease is co-documented; the auditor replaces I10 *Hypertension* with I12.9 *Hypertensive chronic kidney disease*, the target code. In another sample, I48.91 (*Atrial fibrillation, unspecified*) is correctly replaced by I48.0 (*Paroxysmal atrial fibrillation*). There are no incorrect replacements. The two correct additions follow a required companion-code pattern: when severe sepsis is documented, coding guidelines mandate both an underlying sepsis code and a severity code; accordingly, the auditor correctly adds A41.9 *Sepsis, unspecified* and R65.21 *Severe sepsis with septic shock*.

Incorrect changes. Addition accuracy is poor (20%; 2 of 10 correct), revealing two failure modes. The more prevalent is *specificity over-correction* (6 cases): the auditor misinterprets "prefer specificity when documentation supports it", assigning a more specific code even when the documentation is insufficient. The second failure mode is *unsupported code insertion* (2 cases): codes added by clinical inference without a direct guideline support (e.g., Q60.0 *Renal agenesis* added when only a horseshoe kidney variant was documented). Notably, all incorrect changes in step 4 occur despite the relevant code having a guideline available, confirming that the failure is one of guideline misapplication rather than guideline absence.

I Cost Analysis

RAG-Coding is designed with affordability in mind. We report token consumption and actual costs for RAG-Coding (using GPT-4.1 as the backbone model) in Table 13. On average, RAG-Coding costs $0.25 USD to fully code an encounter in the MDACE validation set, where each encounter contains an average of 2.03 notes. Step 3, which involves retrieving and summarising coding guidelines, is the most expensive component, incurring 68% of the overall cost. However, these summaries can be cached and reused, so the cost of step 3 approaches zero over time, reducing the long-term inference cost to approximately $0.08 per encounter. Consequently, RAG-Coding remains practical and cost-effective even on large-scale datasets.

	Tokens Consumed	API Cost ($ USD)
RAG-Coding (step 1)	13,046.30	0.03
RAG-Coding (step 2)	6,779.02	0.02
RAG-Coding (step 3)	85,636.73	0.17
RAG-Coding (step 4)	7,360.03	0.03
Total	112,822.08	0.25

Table 13: Cost breakdown for RAG-Coding (GPT-4.1 backbone). This experiment was conducted on the *MDACE* validation set. Each row shows token (prompt and completion) usage and actual API cost for individual RAG-Coding components. The final row is the total across all components. Note: Step 3 output is cached, so its cost approaches zero in repeated/long-term use.

J Potential Risks

While our proposed RAG-Coding method enhances medical coding performance by grounding predictions in retrieved clinical references, it can produce incorrect code suggestions that could affect financial and research outcomes if used without review. The introduced MDACE-2025 dataset is a re-annotated version of public datasets and therefore poses no new privacy risks.