

REFINE: LLM Refinement over Budgeted Text-Attributed Graphs for Personalized Medical Concept Representation

Mohsen Nayebi Kerdabadi, Arya Hadizadeh Moghaddam, Dongjie Wang, Zijun Yao*

University of Kansas, USA

{mohsen.nayebi, a.hadizadehm, wangdongjie, zyao}@ku.edu

Abstract

Learning rich medical concept representations is essential for EHR prediction. Text-attributed knowledge graphs (TKGs) provide a natural foundation by organizing heterogeneous medical relations together with textual semantics. However, most existing encoders process concepts uniformly across patients, despite the fact that a code’s meaning and predictive value depend on patient-specific clinical context and trajectory. Learning patient-personalized concept representations from TKGs introduces two key challenges: (1) deciding how much KG context to incorporate for each observed code, and (2) aligning semantic information with the patient-specific relational structure. We propose REFINE, a KG-aware budgeted LLM graph refinement framework for patient-personalized medical concept encoding. Starting from a global TKG, REFINE constructs patient-specific temporal graphs. A sequential reinforcement learning policy selects a personalized KG expansion budget for each observed code. The resulting patient graph is processed by a heterogeneous GNN to capture relation-aware structural dependencies, while a frozen LLM uses graph-aware soft prompts to semantically refine concept representations. Experiments on MIMIC-III and MIMIC-IV show that REFINE consistently improves diverse EHR backbones, outperforms strong baselines, and demonstrates robust gains across component ablation, KG selection, and data insufficiency.

1 Introduction

Electronic health records (EHRs) provide a longitudinal view of patient health through high-dimensional sequences of diagnosis, medication, and procedure codes, enabling a wide range of clinical prediction tasks (Nayebi Kerdabadi et al., 2023; Jiang et al., 2023). A central challenge in EHR mining is to learn medical concept representations that capture both rich clinical semantics

and heterogeneous relational dependencies among codes, such as treatment patterns and comorbidities. Text-attributed knowledge graphs (TKGs) provide a natural way to organize these semantic and relational signals by linking medical concepts through typed relations and textual attributes. However, most existing concept encoders learn codes in a static manner across patients, overlooking that the same concept can have different clinical meaning and predictive value depending on patient history, visit context, and treatment trajectory. The same diagnosis may reflect a stable chronic condition in one patient but signal acute deterioration in another; likewise, a medication may be routine in one trajectory but highly informative when paired with specific comorbidities or procedures. Thus, beyond learning general concept semantics from TKGs, medical concept representations should be personalized to each patient’s local clinical context. Personalizing concept representations, however, is non-trivial and raises two challenges.

The first challenge concerns the relational aspect of the TKG: how much KG context should be attached to each observed code in a patient’s history. Existing KG-based EHR models (Jiang et al., 2023; Kerdabadi et al., 2026) often apply fixed expansion rules, retrieving the same hop depth or neighbor budget for every code across patients. This uniform strategy ignores that the value of relational context varies across patients and concepts. A code may be self-explanatory in one trajectory but require richer multi-hop evidence in another to reveal comorbidities, treatment dependencies, or medication interactions. Conversely, unnecessary neighbors can introduce noise, increase computation, and dilute salient clinical signals. Therefore, personalized concept representation requires a graph-budgeting mechanism that adaptively determines how much relational context to retrieve for each observed code within a patient’s history, rather than committing to a single global expansion budget.

*Corresponding author.

The second challenge concerns the semantic text aspect of the TKG: how to use LLMs as patient-personalized semantic refiners rather than generic text encoders. Although LLMs can encode concept semantics, isolated encoding is insufficient because a code’s meaning depends on co-occurring codes, visit position, temporal history, and selected KG context. We therefore define refinement as adapting a code’s general semantics to the patient’s specific clinical context, enabling patient-specific representations of the same code. To refine semantic knowledge in interaction with patient-specific EHR and TKG structure, the key question is how to expose such structure to the LLM. Directly serializing the full patient history and all KG neighbors into text is inefficient, redundant, and does not explicitly preserve the relational scaffold. Thus, patient-personalized semantic refinement requires a compact mechanism that lets the LLM jointly reason over textual semantics, longitudinal EHR context, and selected relational evidence.

In this work, we propose a medical concept encoder for EHR prediction named **REFINE**: KG-Aware LLM Refiner for Budgeted Patient-Personalized Concept Encoding. Starting from a global TKG over EHR codes, REFINE constructs a personalized graph for each patient’s EHR history. For every observed code, a sequential reinforcement learning policy selects a code-specific KG expansion budget, allowing the same medical code to receive rich multi-hop context for one patient but minimal context for another. The policy makes these decisions sequentially, accounting for interactions among observed codes and the cumulative relational context already added to the patient graph. The resulting graph is encoded by two parallel branches: a heterogeneous GNN captures relation-aware structural dependencies, while a frozen LLM refiner uses graph-aware mixed prompts to refine code semantics through interactions among hard text tokens, code soft tokens, and KG-context soft tokens. The hard tokens describe the task and visit-code structure, while the soft tokens compactly represent observed codes and their selected KG context. Finally, the two representations are fused into patient-personalized medical concept embeddings for downstream EHR prediction. Our contributions are:

- We introduce REFINE, a patient-personalized medical concept encoder that learns context-dependent representations for EHR codes by adapting general TKG knowledge to each pa-

tient’s clinical context.

- We develop a patient-specific graph construction and budgeting framework that extracts patient graphs from a global TKG and uses sequential reinforcement learning to select code- and patient-specific relational context.
- We propose a graph-aware LLM refinement mechanism that encodes observed codes and selected KG neighborhoods as soft prompt tokens, enabling semantic refinement through interactions with the patient-specific visit-code scaffold.
- We integrate the resulting personalized concept representations into standard EHR prediction backbones and evaluate their effectiveness on sequential diagnosis prediction.

2 Methodology

2.1 Notation and Problem Definition

EHRs. Let $\mathcal{P}$ denote the patient set and let $\mathcal{C} = \mathcal{C}^{\text{dx}} \cup \mathcal{C}^{\text{rx}} \cup \mathcal{C}^{\text{px}}$ denote the global vocabulary of diagnosis, medication, and procedure codes. The EHR of patient $p \in \mathcal{P}$ is a temporally ordered visit sequence $X_p = (V_{p,1}, \dots, V_{p,T_p})$, where each visit $V_{p,t} \subseteq \mathcal{C}$ is a set of observed codes.

Predictive Task. Given the observed history $X_p^{\leq t^*} = (V_{p,1}, \dots, V_{p,t^*})$, our goal is to learn a patient-personalized medical concept encoder. For each observed code $c \in V_{p,t}$, $1 \leq t \leq t^*$, the encoder Φ_Θ produces

$$\mathbf{z}_{p,t,c} = \Phi_\Theta(c, X_p^{\leq t^*}) \in \mathbb{R}^d, \quad (1)$$

The resulting representations are used by a downstream EHR prediction model. For next-visit diagnosis prediction, we predict:

$$\hat{\mathbf{y}}_p = f_\Omega\left(\{\text{Pool}(\{z_{p,t,c} : c \in V_{p,t}\})\}_{t=1}^{t^*}\right) \quad (2)$$

where $\hat{\mathbf{y}}_p$ is next-visit label and $\hat{\mathbf{y}}_p \in [0, 1]^{|\mathcal{C}^{\text{dx}}|}$.

2.2 Method Summary

We propose REFINE, a patient-personalized medical concept encoder with five steps, shown in Figure 1. (1) We construct a global text-attributed KG whose nodes are medical codes, with LLM-inferred semantic relations. (2) For each patient, we extract a patient-specific graph containing patient, visit, observed-code, and selected KG-neighbor nodes, connected by structural EHR edges and retrieved KG edges. (3) REFINE learns a graph-budgeting

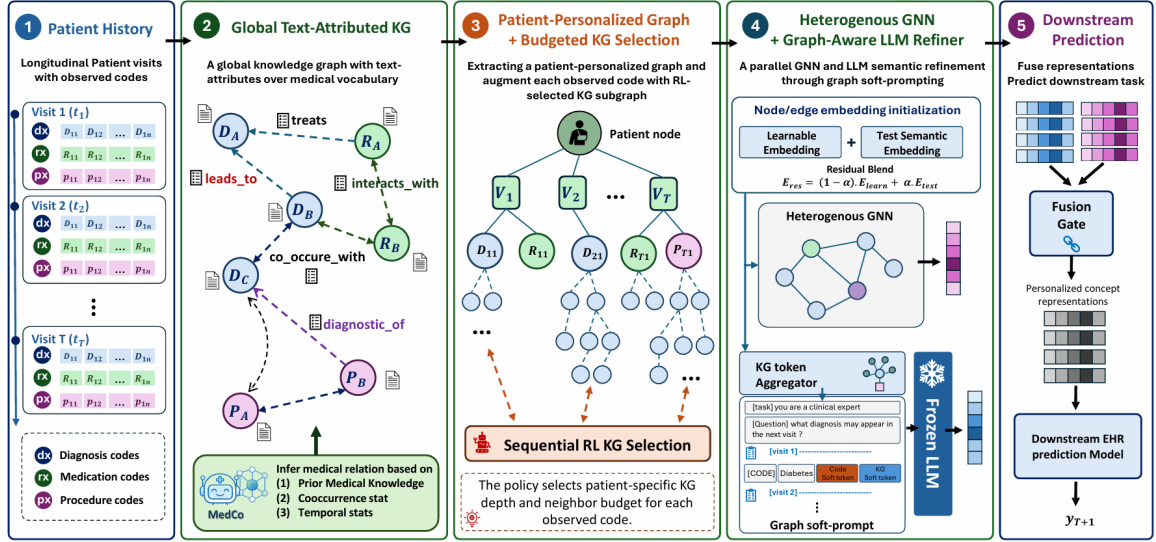


Figure 1: Overview of REFINE. Starting from longitudinal EHR visits and a global text-attributed medical KG, REFINE constructs a patient-personalized graph with RL-selected KG context for each observed code. A heterogeneous GNN and frozen LLM refiner process the selected graph in parallel, and their fused representations are used for downstream EHR prediction.

policy that selects how much KG context to retrieve for each observed code, allowing expansion depth and neighbor budget to vary for codes across patients. (4) The budgeted patient graph is processed by parallel GNN and LLM branches: the GNN captures structural dependencies through message passing, while the LLM refines concept representations based on patient semantic context through mixed prompting. (5) REFINE fuses the GNN and LLM outputs into patient-personalized concept embeddings for downstream EHR prediction.

2.3 Global TKG Construction

We first construct a global TKG over the full medical code vocabulary following Kerdabadi et al. (2026). Candidate code pairs are identified from longitudinal EHRs using statistically reliable intra-visit co-occurrences and next-visit transitions. For each pair, we compute support, conditional probabilities, smoothed PMI, and statistical significance, and filter non-significant pairs. The remaining candidate pairs are passed to a type-constrained LLM relation induction module. Given two codes, EHR-derived statistics, and a type-specific relation set, the LLM assigns a directed semantic relation and a short clinical rationale, yielding clinically meaningful relations such as *treats*, *leads_to*, and *diagnostic_of*. Nodes and edges are further associated with textual attributes, including concept descriptions and relation rationales. Appendix A.1 provides details of the global TKG construction.

We denote the resulting TKG as:

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \phi, \psi), \quad (3)$$

where $\mathcal{V} = \mathcal{C}$ is the set of medical concept nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of directed edges. The mapping $\phi : \mathcal{V} \rightarrow \mathcal{T}$ assigns each node to a clinical type, and $\psi : \mathcal{E} \rightarrow \mathcal{R}$ assigns each edge to a semantic relation type. For an edge $e = (u, v) \in \mathcal{E}$, we write $\psi(e) = r$ and equivalently denote the typed relation as (u, r, v) . Each node v is associated with a textual concept description d_v , and each edge e is associated with a textual relation rationale η_e . Thus, $\mathcal{G}$ is a heterogeneous TKG used as the semantic substrate for extracting patient-specific graphs. Unlike global concept encoders that assign a single embedding to each code, REFINE produces instance-specific code representations conditioned on patient history and the local KG neighborhood.

2.4 Patient-Personalized Graph Construction

Given patient p 's history $X_p^{\leq t^*} = \{V_{p,t}\}_{t=1}^{t^*}$, we construct a patient-personalized graph

$$\mathcal{G}_p^{\leq t^*} = (\mathcal{V}_p, \mathcal{E}_p). \quad (4)$$

Let $\mathcal{S}_p = \bigcup_{t=1}^{t^*} V_{p,t}$ denote the set of distinct medical codes observed in the patient history. For each distinct observed code $c \in \mathcal{S}_p$, REFINE selects a patient-dependent KG subgraph from $\mathcal{G}$:

$$\mathcal{G}_{p,c}^{\text{kg}} = \left(\mathcal{V}_{p,c}^{\text{kg}}, \mathcal{E}_{p,c}^{\text{kg}} \right), \quad \mathcal{G}_{p,c}^{\text{kg}} \subseteq \mathcal{G}. \quad (5)$$

Thus, the same medical code may receive different relational context across different patients.

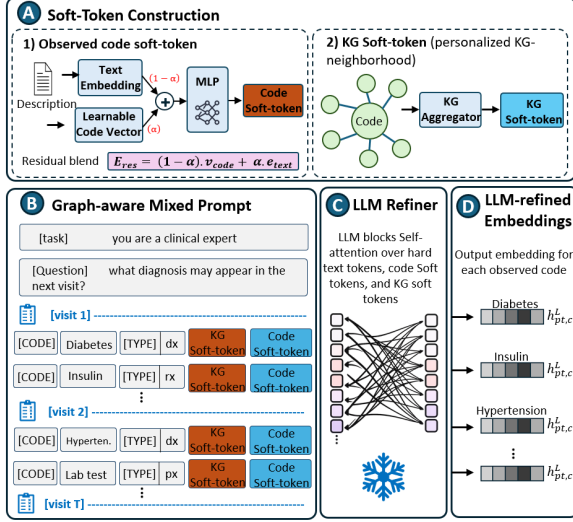


Figure 2: Graph-aware mixed prompting. Patient visits are serialized into a structured prompt with hard text tokens, code soft tokens, and KG-context soft tokens, enabling a frozen LLM to generate personalized semantic representations for observed medical codes.

We use u_p for the patient node, $u_{p,t}^V$ for the visit node at time t , and $u_{p,c}$ for the local node associated with code c . The selected KG code set is $\mathcal{C}_p^{\text{kg}} = \bigcup_{c \in S_p} \mathcal{V}_{p,c}^{\text{kg}}$. The local code vocabulary is then $\mathcal{C}_p^{\text{loc}} = \mathcal{S}_p \cup \mathcal{C}_p^{\text{kg}}$. Accordingly, the patient-graph node set is

$$\mathcal{V}_p = \{u_p\} \cup \{u_{p,t}^V\}_{t=1}^{t^*} \cup \{u_{p,c} : c \in \mathcal{C}_p^{\text{loc}}\}. \quad (6)$$

The edge set contains structural EHR edges and selected KG edges:

$$\mathcal{E}_p = \mathcal{E}_p^{\text{ehr}} \cup \mathcal{E}_p^{\text{kg}}. \quad (7)$$

The structural edges connect the patient node to all visit nodes and each visit node to its observed codes: $\mathcal{E}_p^{\text{ehr}} = \mathcal{E}_p^{p \leftrightarrow v} \cup \mathcal{E}_p^{v \leftrightarrow c}$. Here, $\mathcal{E}_p^{p \leftrightarrow v}$ contains edges between u_p and $u_{p,t}^V$, and $\mathcal{E}_p^{v \leftrightarrow c}$ contains edges between $u_{p,t}^V$ and $u_{p,c}$ for all $c \in \mathcal{V}_{p,t}$. The selected KG edges are obtained by mapping selected global KG edges to their corresponding local code nodes:

$$\mathcal{E}_p^{\text{kg}} = \left\{ (u_{p,a}, u_{p,b}) : (a, b) \in \bigcup_{c \in \mathcal{S}_p} \mathcal{E}_{p,c}^{\text{kg}} \right\}. \quad (8)$$

Each selected KG edge inherits its semantic relation type and textual rationale from the global TKG $\mathcal{G}$. The neighborhood selection policy is described in Sec. 2.5, determining how much relational context to retrieve for each observed code.

2.5 Personalized Concept Sub-KG Extraction

Using the same KG expansion budget for all codes is suboptimal, since the utility of relational context depends on the patient history and code ambiguity. We therefore learn a patient-specific policy that decides how much KG context to retrieve for each distinct observed code root in the patient history. For each root code $c \in S_p$, REFINE selects a patient-specific KG subgraph from the global TKG.

Expansion actions. For each distinct observed code root $c \in S_p$, we define a discrete expansion action $a = (K, \mathbf{m}) \in \mathcal{A}$, where $K \in \{0, \dots, K_{\max}\}$ is the KG expansion depth and $\mathbf{m} = (m_1, \dots, m_K)$ specifies the per-hop fanout. The no-expansion action is $a_0 = (0, \emptyset)$. For example, $(2, (2, 1))$ expands up to two hops from the observed code: it first retains the top-2 one-hop neighbors and then the top-1 second-hop neighbor for each selected first-hop node. Neighbors are not sampled randomly; candidate edges are ranked using evidence scores derived from the statistical evidence computed during global KG construction. Thus, the RL policy controls the patient-code-specific expansion budget, while neighbor selection within each budget remains evidence-grounded.

Sequential decision process. The policy makes sequential decisions over the distinct observed code roots S_p . Initially, each root code is assigned the no-expansion action a_0 . At step ℓ , the policy selects a macro-action $b_\ell = (c, a)$, which updates the expansion action for one distinct observed code root. This sequential formulation allows each decision to condition on the KG context already selected for other codes. Let $\mathcal{E}_p^{\text{kg},(\ell)}$ and $\mathcal{C}_p^{\text{kg},(\ell)}$ denote the selected global KG edges and selected KG codes after step ℓ . The incremental cost of assigning action a to root code c is

$$\Delta C_\ell(c, a) = \omega_E \left| \mathcal{E}_{p,c}^{\text{kg}}(a) \setminus \mathcal{E}_p^{\text{kg},(\ell)} \right| + \omega_C \left| \mathcal{V}_{p,c}^{\text{kg}}(a) \setminus \left(\mathcal{C}_p^{\text{kg},(\ell)} \cup S_p \right) \right| \quad (9)$$

The process terminates when the policy selects a stop action, reaches the maximum number of steps, or exhausts the graph budget.

Policy and value networks. At step ℓ , the policy observes a state vector $\mathbf{s}_\ell$ that summarizes the patient history, selected KG context, and remaining budget. For each feasible macro-action $b = (c, a)$, we construct an action feature $\xi_\ell(b)$ containing the

state, code representation, action embedding, and incremental cost. The actor scores feasible macro-actions as

$$\pi_\eta(b \mid \mathbf{s}_\ell) = \text{softmax}_b(f_\eta(\boldsymbol{\xi}_\ell(b))), \quad (10)$$

and the critic estimates the state value

$$V_\omega(\mathbf{s}_\ell) = f_\omega(\mathbf{s}_\ell). \quad (11)$$

During training, macro-actions are sampled from π_η for exploration; during validation and inference, we use greedy selection.

Reward. During Stage I, after the policy constructs the selected patient graph $\mathcal{G}_p^{\text{sel}}$, we evaluate it using the frozen GNN-only proxy predictor. Let $\mathcal{L}_p(\mathcal{G})$ denote the corresponding prediction loss for patient p under graph $\mathcal{G}$, and let $\mathcal{G}_p^0$ be the baseline graph obtained by assigning a_0 to all observed codes. The terminal reward is

$$R_p = \mathcal{L}_p(\mathcal{G}_p^0) - \mathcal{L}_p(\mathcal{G}_p^{\text{sel}}) - \lambda_{\text{cost}} C_p. \quad (12)$$

Here, the graph cost is

$$C_p = \omega_E |\mathcal{E}_p^{\text{kg,sel}}| + \omega_C |\mathcal{C}_p^{\text{kg,sel}} \setminus \mathcal{S}_p|. \quad (13)$$

Thus, the policy is rewarded for improving prediction relative to the no-expansion graph while being penalized for adding unnecessary KG context.

RL objective. Let $\tau_p = \{b_\ell\}_{\ell=1}^{L_p}$ denote the selected macro-action trajectory for patient p . We optimize an actor-critic objective with entropy regularization. The advantage at step ℓ is

$$A_\ell = R_p - V_\omega(\mathbf{s}_\ell). \quad (14)$$

The RL loss is

$$\begin{aligned} \mathcal{L}_{\text{RL}} = \mathbb{E}_p \sum_{\ell=1}^{L_p} \Big[& -\lambda_{\text{act}} \log \pi_\eta(b_\ell \mid \mathbf{s}_\ell) A_\ell \\ & + \lambda_{\text{crit}} (V_\omega(\mathbf{s}_\ell) - R_p)^2 - \lambda_{\text{ent}} H(\pi_\eta(\cdot \mid \mathbf{s}_\ell)) \Big]. \end{aligned} \quad (15)$$

2.6 Graph-Aware LLM Refinement for Personalized Concept Representation

After constructing the patient-personalized graph $\mathcal{G}_p^{\leq t^*}$, REFINE processes it with two parallel branches: a heterogeneous GNN branch and an LLM refinement branch. Both are grounded in the same fixed text-derived semantics from node descriptions d_c and edge rationales η_e , but each branch maintains its own trainable parameters.

Node and edge embedding initialization. For each branch $q \in \{G, L\}$, where G and L denote the GNN and LLM branches, respectively, we initialize code and KG-edge representations with a residual blend of learnable embeddings and fixed text-derived semantics. For a code node $u_{p,c}$ with description encoding $\tilde{\mathbf{d}}_c$, we define

$$\mathbf{h}_{u_{p,c}}^{q,0} = (1 - \lambda_c^q) \mathbf{a}_c^q + \lambda_c^q \mathbf{W}_d^q \tilde{\mathbf{d}}_c, \quad (16)$$

where $\mathbf{a}_c^q$ is a learnable code embedding. Patient and visit nodes use learnable type and position embeddings. For a selected KG edge $e = (u_{p,a}, u_{p,b})$ corresponding to the global edge $(a, b) \in \mathcal{E}$, with relation $r_e = \psi((a, b))$ and rationale encoding $\tilde{\eta}_{(a,b)}$, we define

$$\mathbf{e}_e^q = (1 - \lambda_r^q) \mathbf{a}_{r_e}^q + \lambda_r^q \mathbf{W}_\eta^q \tilde{\eta}_{(a,b)}. \quad (17)$$

This initialization injects clinical semantics while preserving trainable branch-specific flexibility.

Heterogeneous GNN branch. The GNN branch performs message passing over the selected patient graph $\mathcal{G}_p^{\leq t^*}$. Let $\chi(u)$ denote the node type of u and r_e denote the edge type of $e \in \mathcal{E}_p$. For each layer ℓ , an edge-specific message is computed as

$$\mathbf{m}_{u \rightarrow v}^{(\ell)} = f_{r_e}^{\text{msg}} \left(\mathbf{h}_u^{G,\ell}, \mathbf{e}_e^G \right), \quad e = (u, v). \quad (18)$$

Messages are aggregated by relation type and used to update the destination node with a type-specific update function:

$$\mathbf{m}_{v,r}^{(\ell)} = \text{AGG}_r \{ \mathbf{m}_{u \rightarrow v}^{(\ell)} : e \in \mathcal{E}_p, r_e = r \}, \quad (19)$$

$$\mathbf{h}_v^{G,\ell+1} = f_{\chi(v)}^{\text{upd}} \left(\mathbf{h}_v^{G,\ell}, \{ \mathbf{m}_{v,r}^{(\ell)} \}_{r \in \mathcal{R}_p(v)} \right). \quad (20)$$

where $\mathcal{R}_p(v) = \{r_e : e = (u, v) \in \mathcal{E}_p\}$ denotes the set of incoming edge types at node v . After L layers, the GNN produces a structural representation $\mathbf{h}_{p,c}^G \in \mathbb{R}^d$ for each distinct observed code, refined by the patient-specific selected graph.

LLM refiner branch. For the LLM branch, each observed code is represented by a code soft token and a KG-context soft token. The code soft token is obtained by projecting the LLM-branch residual code representation into the LLM hidden space: $\mathbf{s}_{p,c} = \mathbf{W}_{\text{soft}}^c \mathbf{h}_{u_{p,c}}^{L,0} \in \mathbb{R}^{d_{\text{LLM}}}$. To summarize the selected KG context $\mathcal{G}_{p,c}^{\text{kg}}$, we aggregate the LLM-branch representations of its selected neighbor nodes and relations:

$$\begin{aligned} \mathbf{k}_{p,c} = \text{KGPool} \Big(& \{ \mathbf{h}_{u_{p,v}}^{L,0} : v \in \mathcal{V}_{p,c}^{\text{kg}} \}, \\ & \{ \mathbf{e}_e^L : e \in \mathcal{E}_{p,c}^{\text{kg}} \} \Big). \end{aligned} \quad (21)$$

KGPool($\cdot$) applies a lightweight GNN-style aggregator over the selected code-specific KG sub-graph, combining selected neighbor-node embeddings and relation/rationale-aware edge embeddings, producing a compact KG-context token for code c .

Mixed Graph Prompt. We keep the LLM frozen and construct a mixed prompt without modifying its tokenizer or embedding table. Hard tokens encode the task, visit structure, code names, and code types, while soft tokens inject code-level and KG-context semantics. For patient p , the prompt follows a visit-code hierarchy:

$$\mathcal{J}(p, t^*) = [x_{\text{task}}, x_{\text{quest}}, \{\Gamma(V_{p,t})\}_{t=1}^{t^*}], \quad (22)$$

where x_{task} and x_{quest} denote the hard-token task instruction and prediction query, respectively, and each visit block contains its observed codes,

$$\Gamma(V_{p,t}) = [\Gamma(p, t, c) : c \in V_{p,t}], \quad (23)$$

and each code block is represented as

$$\Gamma(p, t, c) = [\text{name}(c), \phi(c), \langle \mathbf{s}_{p,c} \rangle, \langle \mathbf{k}_{p,c} \rangle]. \quad (24)$$

LLM refinement branch. Let $\mathbf{X}_p \in \mathbb{R}^{M_p \times d_{\text{LM}}}$ be the final mixed embedding sequence obtained by interleaving LLM token embeddings for hard text with code and KG soft tokens. The frozen LLM encodes this sequence as

$$\mathbf{H}_p^L = \text{LLM}(\mathbf{X}_p). \quad (25)$$

For each observed code $c \in V_{p,t}$, we extract the hidden state at its code-soft-token position and project it back to the model hidden space:

$$\mathbf{h}_{p,t,c}^L = \mathbf{W}_L \mathbf{H}_{p,i(p,t,c)}^L \in \mathbb{R}^d, \quad (26)$$

where $i(p, t, c)$ denotes the prompt position of the code soft token. This representation captures semantic interactions among hard textual context, code soft tokens, and selected KG-context tokens. When a code appears in multiple visits, its input code and KG-context soft tokens are shared across occurrences, while the resulting LLM hidden states remain visit-specific through their distinct prompt positions.

Fusion of GNN and LLM Representations. The GNN and LLM branches capture complementary patient-personalized information: $\mathbf{h}_{p,c}^G$ represents graph-structural context, while $\mathbf{h}_{p,t,c}^L$ represents LLM-refined semantic context. We combine

them using gated fusion:

$$\mathbf{g}_{p,t,c} = \sigma(\mathbf{W}_F[\mathbf{h}_{p,c}^G; \mathbf{h}_{p,t,c}^L]), \quad (27)$$

$$\mathbf{z}_{p,t,c} = \mathbf{g}_{p,t,c} \odot \mathbf{h}_{p,t,c}^L + (1 - \mathbf{g}_{p,t,c}) \odot \mathbf{h}_{p,c}^G \quad (28)$$

$\mathbf{z}_{p,t,c}$ is then used for downstream EHR prediction.

2.7 Integration with Downstream Models

REFINE serves as a plug-in concept encoder for standard EHR models. Given the observed history $X_p^{\leq t^*}$, it produces a patient-personalized representation for each observed code $c \in V_{p,t}$:

$$\mathbf{z}_{p,t,c} = \text{REFINE}_{\Theta}(c, X_p^{\leq t^*}, \mathcal{G}) \quad (29)$$

These code representations are aggregated within each visit and passed to an EHR backbone:

$$\mathbf{h}_p = f_{\Omega}\left(\{\text{Pool}(\{\mathbf{z}_{p,t,c} : c \in V_{p,t}\})\}_{t=1}^{t^*}\right) \quad (30)$$

The final next-visit diagnosis prediction is

$$\hat{\mathbf{y}}_p = \sigma(\mathbf{W}_o \mathbf{h}_p + \mathbf{b}_o), \quad \hat{\mathbf{y}}_p \in [0, 1]^{|C^{\text{dx}}|}. \quad (31)$$

In Stage II, with the selected patient graphs fixed, we train the full model using multi-label binary cross-entropy between $\hat{\mathbf{y}}_p$ and the next-visit diagnosis label $\mathbf{y}_p$.

3 Experiments

Experimental Setting. We evaluate on two public EHR benchmarks, MIMIC-III (Johnson et al., 2016) and MIMIC-IV (Johnson et al., 2023). Table 2 reports cohort statistics. The task is next-visit diagnosis prediction over imbalanced label spaces (515 codes in MIMIC-III; 562 in MIMIC-IV). Implementation details, including LLM/GNN configurations and the two-stage training pipeline, are provided in Appendix A.3. The source code is available at <https://github.com/mohsen-nyb/REFINE.git>

Evaluation Metrics. We report results averaged over five folds for **AUPRC** (both general and also stratified by label frequency to assess performance across code rarity levels), **Acc@k** (top- k accuracy normalized by $\min(k, |\mathbf{y}_{t+1}|)$), and **F1**.

4 Results

We evaluate REFINE through five research questions: **RQ1:** Does REFINE improve downstream EHR prediction as a plug-in concept encoder? **RQ2:** How does REFINE compare with existing

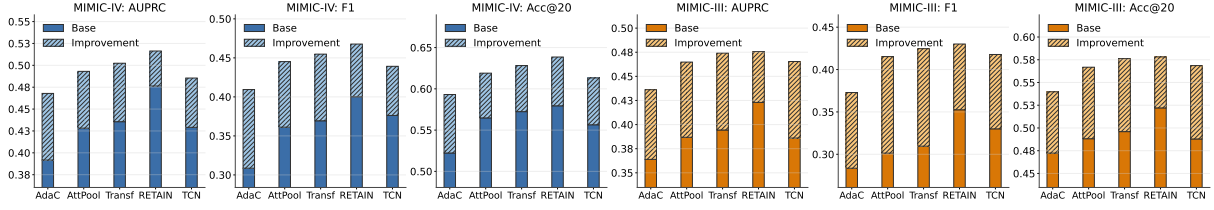


Figure 3: Performance gains from integrating REFINE as a plug-in concept encoder across prediction backbones.

Table 1: Baseline performance comparison. We report overall AUPRC, F1, and Acc@k, and stratified AUPRC by label frequency quartiles. The first row corresponds to the selected base model, Transformer, and subsequent rows denote plug-in variants, for example REFINE = REFINE + Base. All values are reported as percentages.

Model		General Performance					Label-Frequency AUPRC			
		AUPRC	F1	Acc@15	Acc@20	Acc@30	0–25%	25–50%	50–75%	75–100%
MIMIC-III	Base (Vaswani, 2017)	39.44	30.96	46.47	49.60	57.73	38.64	47.30	70.63	74.30
	GRAM (Choi et al., 2017)	40.10	32.32	47.75	51.02	58.71	40.06	49.84	71.78	75.16
	MMORE (Song et al., 2019)	40.95	33.08	48.42	51.73	59.34	40.72	50.92	72.55	75.86
	KAME (Ma et al., 2018)	40.54	32.83	48.17	51.20	58.99	39.70	49.49	71.70	75.02
	G-BERT (Shang et al., 2019)	40.83	33.15	48.47	51.49	59.22	39.93	49.78	71.94	75.18
	HAP (Zhang et al., 2020)	40.70	32.95	48.30	51.34	59.12	39.82	49.63	71.82	75.10
	ADORE (Cheong et al., 2023)	40.92	33.20	48.55	51.63	59.28	40.05	49.86	71.99	75.21
	KAMPNet (An et al., 2023)	41.37	33.70	49.10	52.20	59.82	40.50	50.59	72.50	75.78
	GraphCare (Jiang et al., 2023)	43.35	35.46	52.76	56.00	62.75	44.80	58.16	70.97	65.72
	Rel-LLM (Wu et al., 2025)	43.58	36.05	52.90	56.08	62.92	44.95	58.35	73.85	78.40
	LINKO (Nayebi Kerdabadi et al., 2025)	44.10	37.05	53.05	56.20	63.00	45.20	59.05	74.25	79.20
	MedCo (Kerdabadi et al., 2026)	45.35	39.90	53.22	56.32	63.18	45.73	64.00	73.80	82.80
	REFINE	47.41	42.46	54.38	57.62	64.60	47.92	66.55	76.28	86.95
MIMIC-IV	Base (Vaswani, 2017)	43.56	36.94	54.02	57.24	63.64	44.93	53.50	51.85	73.10
	GRAM (Choi et al., 2017)	44.72	37.78	54.76	58.02	64.56	45.76	54.39	52.60	73.80
	MMORE (Song et al., 2019)	45.28	38.35	55.27	58.54	64.96	46.25	54.88	53.47	74.30
	KAME (Ma et al., 2018)	44.39	37.43	54.45	57.65	64.09	45.22	54.05	52.18	73.52
	G-BERT (Shang et al., 2019)	44.80	37.86	54.86	58.06	64.50	45.63	54.52	52.76	73.94
	HAP (Zhang et al., 2020)	44.75	37.70	54.75	58.00	64.43	45.55	54.37	52.69	73.84
	ADORE (Cheong et al., 2023)	44.94	38.00	55.00	58.19	64.65	45.78	54.64	52.90	74.02
	KAMPNet (An et al., 2023)	45.48	38.65	55.61	58.86	65.32	46.31	55.34	54.00	74.60
	GraphCare (Jiang et al., 2023)	46.90	40.12	55.89	59.14	65.71	48.31	60.31	73.70	66.33
	Rel-LLM (Wu et al., 2025)	47.20	40.55	56.45	59.85	66.35	48.70	56.80	56.40	76.60
	LINKO (Nayebi Kerdabadi et al., 2025)	47.55	41.10	57.05	60.55	67.05	49.10	57.10	56.95	77.20
	MedCo (Kerdabadi et al., 2026)	48.28	43.10	58.30	61.55	67.60	49.95	57.35	57.70	78.20
	REFINE	50.24	45.48	59.48	62.82	69.00	52.30	57.98	59.65	81.55

Table 2: Data statistics for MIMIC-III and MIMIC-IV.

Metric	MIMIC-III	MIMIC-IV
# Patients	7,515	18,829
# Visits (samples)	12,430	25,028
# Labels/sample	12.31	10.56
# Unique conditions (ICD)	515	562
# Conditions/sample	22.94	59.50
# Drugs/sample	54.95	118.16
# Unique drugs	468	510
# Procedures/sample	5.92	5.41
# Unique procedures	280	322

baselines? **RQ3:** Which components contribute the most to performance? **RQ4:** Does learned KG budgeting outperform fixed KG selection strategies? **RQ5:** Does REFINE improve prediction under data scarcity? RQ1–RQ4 are addressed in the main results, and RQ5 in Appendix A.6.

4.1 RQ1: Plug-in Encoder Evaluation

We evaluate whether REFINE improves standard EHR backbones when used as a plug-in concept encoder. We integrate REFINE into five representative models: **AttPool**, a simple attention pooling predictor; **AdaCare** (Ma et al., 2020); **Trans-**

Table 3: Component-wise ablation of REFINE on MIMIC-IV and MIMIC-III.

	Variant	AUPRC	F1	Acc@20
MIMIC-IV	Full REFINE	50.24	45.48	62.82
	w/o RL budgeting	49.18	44.18	61.92
	w/o full LLM refiner	49.32	44.05	61.84
	w/o soft prompt tokens	49.38	44.12	61.90
	w/o KG soft tokens	49.72	44.78	62.32
	w/o full personalized GNN	47.35	40.82	60.25
	Global code representation (MedCo)	48.28	43.10	61.55
MIMIC-III	Full REFINE	47.41	42.46	57.62
	w/o RL budgeting	46.38	41.02	56.70
	w/o full LLM refiner	46.51	40.88	56.62
	w/o soft prompt tokens	46.57	40.96	56.68
	w/o KG soft tokens	46.92	41.68	57.10
	w/o full personalized GNN	43.85	36.62	56.14
	Global code representation (MedCo)	45.35	39.90	56.32

former (Vaswani, 2017); **RETAIN** (Choi et al., 2016b); and **TCN** (Bai et al., 2018). For each backbone, we compare the base model with its REFINE-enhanced variant. As shown in Figure 3, REFINE consistently improves performance across backbones, demonstrating its effectiveness as a plug-in personalized concept encoder.

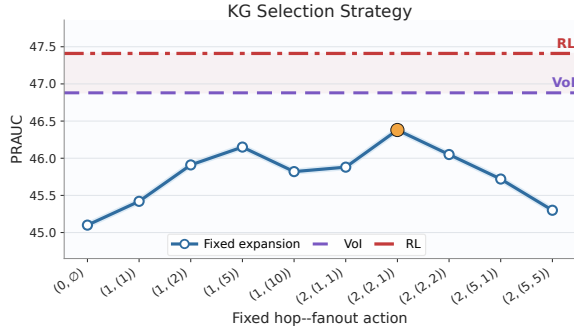


Figure 4: AUPRC comparison of KG selection strategies.

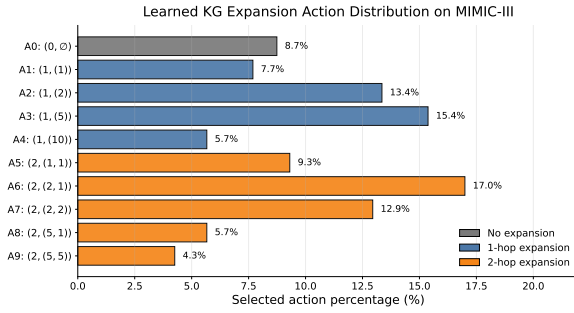


Figure 5: Distribution of learned KG expansion actions.

4.2 RQ2: Baseline Comparison

We compare REFINE with representative medical concept encoders, including the **base** Transformer (Vaswani, 2017), ontology-based methods (**GRAM** (Choi et al., 2017), **MMORE** (Song et al., 2019), **KAME** (Ma et al., 2018), **G-BERT** (Shang et al., 2019), **HAP** (Zhang et al., 2020)), and KG-based methods: **ADORE** (Cheong et al., 2023) uses SNOMED relations; **Graph-Care** (Jiang et al., 2023) combines UMLS- and LLM-derived edges; **KAMPNet** (An et al., 2023) and **LINKO** (Nayebi Kerdabadi et al., 2025) construct cross-type connections; **Rel-LLM** follows relational graph prompting (Wu et al., 2025); and **MedCo** (Kerdabadi et al., 2026) learns global KG-LLM concept embeddings from a medical TKG. All methods use the same downstream backbone. As shown in Table 1, REFINE achieves the best overall performance, highlighting the benefit of patient-personalized, budgeted TKG refinement. For rare-condition analysis, we stratify diagnosis labels into frequency quartiles; REFINE achieves the best AUPRC in six of eight quartile groups.

4.3 RQ3: Ablation Study

Table 3 shows that each component contributes to the final performance of REFINE. Removing the personalized GNN yields the largest drop, confirming that the frozen LLM alone is insufficient and

that patient-graph structure is central to REFINE. Removing the LLM refiner or using only hard name tokens also degrades performance, showing that semantic refinement benefits from trainable soft prompts rather than text names alone. The drop without KG soft tokens indicates that selected relational context adds value beyond observed-code soft tokens. Finally, removing RL budgeting consistently hurts performance, demonstrating the importance of adaptively selecting patient- and code-specific KG context instead of using an unbudgeted or fixed expansion. Even without the LLM refiner, REFINE outperforms MedCo, showing that personalized budgeted TKG construction alone improves over global KG-based code representations.

4.4 RQ4: Personalized KG Selection

Figure 4 compares different KG selection strategies. Fixed strategies apply the same hop-fanout expansion to all observed codes, and their performance varies substantially across actions, indicating that a single global KG budget is suboptimal. The value-of-information (VoI) strategy¹ improves over fixed expansion by using a learned utility critic to select visit-level KG expansion actions. The critic is trained with sparse label-derived utility targets, while validation and test-time selection is label-free. In contrast, the RL policy performs sequential code-level budgeting, allowing different observed code roots to receive different KG expansions while conditioning on previously selected KG context. Its superior performance demonstrates the benefit of adaptive, fine-grained KG selection.

Analysis of learned RL action distribution. To better understand the learned selection behavior, Figure 5 shows the distribution of RL policy expansion actions on MIMIC-III. The policy does not collapse to a single hop-fanout configuration, but instead selects a diverse range of no-expansion, one-hop, and two-hop actions, with moderate one-hop and two-hop budgets among the more frequently selected configurations. This diversity suggests that the selector balances predictive utility against graph-expansion cost across different patient-code instances.

Qualitative personalization analysis. To make this behavior more concrete, we inspect representative RL policy decisions from individual MIMIC-III prediction samples. Here, code frequency de-

¹Details of the VoI strategy are provided in Appendix A.5.

Table 4: Representative examples of RL-policy budget personalization on MIMIC-III. Code frequency is the number of prediction samples containing the code at least once.

Code / Patient	Code Freq.	Selected Budget	Patient Context	Interpretation
conditions:14.5.23	7	2-hop (5, 1)	2 visits, 9 observed codes	A rare diagnosis receives deeper multi-hop relational context in a short patient history.
procedures:7.12.2	10	2-hop (5, 1)	1 visit, 41 observed codes	A rare procedure also receives deeper expansion, showing that adaptive budgeting is not limited to diagnosis codes.
drugs:B05XA	10,809	0-hop	5 visits, 211 observed codes	A highly frequent drug receives no additional KG context, showing selective expansion.
conditions:16.10.1.4	683	1-hop (5)	5 visits, 182 observed codes	A medium-frequency diagnosis receives only one-hop context expansion.

notes the number of prediction samples containing a code at least once. As shown in Table 4, the policy does not uniformly expand all observed code roots. For example, a rare diagnosis appearing in only 7 samples receives a 2-hop budget in a patient history with 2 visits and 9 observed codes, while a rare procedure appearing in 10 samples also receives a 2-hop budget in another patient. In contrast, a frequent medication appearing in 10,809 samples receives no KG expansion in a patient with 5 visits and 211 observed codes, while a medium-frequency diagnosis receives only local 1-hop context. These examples show that the RL policy assigns heterogeneous budgets across codes and patient contexts rather than uniformly expanding patient graphs.

5 Related Work

The increasing availability of EHRs has advanced clinical prediction methods, ranging from early sequential models (Choi et al., 2016a,b) to attention-based and Transformer architectures (Li et al., 2020; Nayeji Kerdabadi et al., 2023; Hu et al., 2025; Hadizadeh Moghaddam et al., 2024), as well as graph-based approaches (Xu et al., 2022; Yang et al., 2023). These methods primarily improve patient-level prediction architectures, whereas our focus is on learning patient-personalized representations of the medical concepts.

A major line of work improves medical code representations by incorporating pre-existing medical ontologies or KG structure. GRAM (Choi et al., 2017), MMORE (Song et al., 2019), KAME (Ma et al., 2018), and HAP (Zhang et al., 2020) exploit hierarchical ontology structure to refine concept embeddings through parent-child dependencies and attention mechanisms. Beyond hierarchies, G-BERT (Shang et al., 2019), ADORE (Cheong et al., 2023), KAMPNet (An et al., 2023), and LINKO (Nayeji Kerdabadi et al., 2025) incorporate richer relational graphs, multi-source knowledge, or contrastive objectives. GCL (Lu et al.,

2021), RAM-EHR (Xu et al., 2024), KARE (Jiang et al., 2024), and GraphCare (Jiang et al., 2023) further integrate retrieved knowledge, auxiliary text, or patient-specific graph context. However, GraphCare relies on fixed neighborhood expansion, whereas REFINE learns sequential patient- and code-specific KG budgets, allowing different observed codes within the same patient to receive different amounts of relational context.

Recent work also combines LLMs with relational or medical graph representations. Rel-LLM (Wu et al., 2025) injects explicit relational structure through graph-aware soft prompting, while REFINE compresses the selected code-specific KG neighborhood into compact soft context tokens within a visit-aware prompt, avoiding explicit serialization of every retrieved relation. MedCo (Kerdabadi et al., 2026) learns a single global KG-LLM representation per medical concept. In contrast, REFINE learns patient-code representations conditioned on longitudinal history and adaptively selected KG context, using this relational evidence to guide LLM semantic refinement.

6 Conclusion

We presented REFINE, a patient-personalized plug-in medical concept encoder that learns EHR code representations from budgeted text-attributed patient graphs. REFINE uses sequential reinforcement learning to select code- and patient-specific KG budgets, builds compact personalized graphs, and combines a heterogeneous GNN with a frozen LLM through graph-aware soft prompts. Experiments on MIMIC-III and MIMIC-IV show consistent gains across diverse backbones and strong baselines. Ablation and KG-selection analyses confirm the complementary benefits of personalized graph construction, adaptive budgeting, and LLM refinement, while showing that the learned policy adapts budgets across codes and patient contexts. REFINE also remains effective under limited supervision and across frozen LLM backbones.

7 Limitations and Future Work

A limitation of our KG selection module is its pre-defined discrete action space with fixed hop and neighbor budgets. Although neighbors within each selected budget are chosen by evidence-based edge rankings rather than random sampling, the policy does not directly select the exact nodes or adapt the budget continuously at each hop. Future work will explore more flexible RL strategies that jointly select, rank, and budget KG nodes while remaining efficient for large-scale EHR prediction.

8 Ethical Considerations

We comply with the ACL Ethics Policy throughout this study. All experiments use publicly available, de-identified EHR datasets, which provide strong privacy protections. We do not send patient-level records or identifiable clinical histories to external LLM services. LLM-based KG construction uses only concept-level information and aggregated EHR-derived statistics, and the LLM used in RE-FINE is run locally and kept frozen. AI-assisted tools were used solely for writing and grammar checking, and not for data analysis, experimental design, or generating scientific claims. The proposed model is intended for research on EHR representation learning and should not be used for clinical decision-making without prospective validation, fairness evaluation, and expert oversight.

9 Acknowledgments

This research was supported by the National Science Foundation under Award No. 2544634.

References

- Yang An, Haocheng Tang, Bo Jin, Yi Xu, and Xiaopeng Wei. 2023. Kampnet: multi-source medical knowledge augmented medication prediction network with multi-level graph contrastive learning. *BMC Medical Informatics and Decision Making*, 23(1):243.
- Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 10.
- Chin Wang Cheong, Kejing Yin, William K Cheung, Benjamin CM Fung, and Jonathan Poon. 2023. Adaptive integration of categorical and multi-relational ontologies with ehr data for medical concept embedding. *ACM Transactions on Intelligent Systems and Technology*, 14(6):1–20.
- Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F Stewart, and Jimeng Sun. 2016a. Doctor ai: Predicting clinical events via recurrent neural networks. In *Machine learning for healthcare conference*, pages 301–318. PMLR.
- Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F Stewart, and Jimeng Sun. 2017. Gram: graph-based attention model for healthcare representation learning. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 787–795.
- Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. 2016b. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. *Advances in neural information processing systems*, 29.
- Aaron Grattafiori and et al. 2024. [The llama 3 herd of models](#). *arXiv preprint arXiv:2407.21783*.
- Arya Hadizadeh Moghaddam, Mohsen Nayebi Kerdabadi, Mei Liu, and Zijun Yao. 2024. Contrastive learning on medical intents for sequential prescription recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 748–757.
- Jinxiang Hu, Mohsen Nayebi Kerdabadi, Xiaohang Mei, Joseph Cappelleri, Richard Barohn, and Zijun Yao. 2025. Recurrent neural networks and attention scores for personalized prediction and interpretation of patient-reported outcomes. *Journal of Biopharmaceutical Statistics*, pages 1–11.
- Binyuan Hui, Jian Yang, Zeyu Cui, Jiayi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.
- Pengcheng Jiang, Cao Xiao, Adam Cross, and Jimeng Sun. 2023. Graphcare: Enhancing healthcare predictions with personalized knowledge graphs. *arXiv preprint arXiv:2305.12788*.
- Pengcheng Jiang, Cao Xiao, Minhao Jiang, Parminder Bhatia, Taha Kass-Hout, Jimeng Sun, and Jiawei Han. 2024. Reasoning-enhanced healthcare predictions with knowledge graph community retrieval. *arXiv preprint arXiv:2410.04585*.
- 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.

- Mohsen Nayebi Kerdabadi, Arya Hadizadeh Moghadam, Chen Chen, Dongjie Wang, and Zijun Yao. 2026. Text-attributed knowledge graph enrichment with large language models for medical concept representation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16544–16560.
- Yikuan Li, Shishir Rao, José Roberto Ayala Solares, Abdelaali Hassaine, Rema Ramakrishnan, Dexter Canoy, Yajie Zhu, Kazem Rahimi, and Gholamreza Salimi-Khorshidi. 2020. Behrt: transformer for electronic health records. *Scientific reports*, 10(1):7155.
- Chang Lu, Chandan K Reddy, Prithwish Chakraborty, Samantha Kleinberg, and Yue Ning. 2021. Collaborative graph learning with auxiliary text for temporal event prediction in healthcare. *arXiv preprint arXiv:2105.07542*.
- Fenglong Ma, Quanzeng You, Houping Xiao, Radha Chitta, Jing Zhou, and Jing Gao. 2018. Kame: Knowledge-based attention model for diagnosis prediction in healthcare. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 743–752.
- Liantao Ma, Junyi Gao, Yasha Wang, Chaohe Zhang, Jiangtao Wang, Wenjie Ruan, Wen Tang, Xin Gao, and Xinyu Ma. 2020. Adacare: Explainable clinical health status representation learning via scale-adaptive feature extraction and recalibration. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 825–832.
- Meta. 2024. Llama-3.2-1b model card. <https://huggingface.co/meta-llama/Llama-3.2-1B>. Hugging Face model card, accessed 2026-04-13.
- Mohsen Nayebi Kerdabadi, Arya Hadizadeh Moghadam, Bin Liu, Mei Liu, and Zijun Yao. 2023. Contrastive learning of temporal distinctiveness for survival analysis in electronic health records. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 1897–1906.
- Mohsen Nayebi Kerdabadi, Arya Hadizadeh Moghadam, Dongjie Wang, and Zijun Yao. 2025. Multi-ontology integration with dual-axis propagation for medical concept representation. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 2201–2211.
- Qwen Team. 2024. Qwen2.5-0.5b model card. <https://huggingface.co/Qwen/Qwen2.5-0.5B>. Hugging Face model card, accessed 2026-04-13.
- Junyuan Shang, Tengfei Ma, Cao Xiao, and Jimeng Sun. 2019. Pre-training of graph augmented transformers for medication recommendation. *arXiv preprint arXiv:1906.00346*.
- Lihong Song, Chin Wang Cheong, Kejing Yin, William K Cheung, Benjamin CM Fung, and Jonathan Poon. 2019. Medical concept embedding with multiple ontological representations. In *IJCAI*, volume 19, pages 4613–4619.
- A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.
- Fang Wu, Vijay Prakash Dwivedi, and Jure Leskovec. 2025. Large language models are good relational learners. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7835–7854.
- Ran Xu, Wenqi Shi, Yue Yu, Yuchen Zhuang, Bowen Jin, May D Wang, Joyce C Ho, and Carl Yang. 2024. Ram-ehr: Retrieval augmentation meets clinical predictions on electronic health records. *arXiv preprint arXiv:2403.00815*.
- Ran Xu, Yue Yu, Chao Zhang, Mohammed K Ali, Joyce C Ho, and Carl Yang. 2022. Counterfactual and factual reasoning over hypergraphs for interpretable clinical predictions on ehr. In *Machine Learning for Health*, pages 259–278. PMLR.
- Nianzu Yang, Kaipeng Zeng, Qitian Wu, and Junchi Yan. 2023. Molerec: Combinatorial drug recommendation with substructure-aware molecular representation learning. In *Proceedings of the ACM Web Conference 2023*, pages 4075–4085.
- Muhan Zhang, Christopher R King, Michael Avidan, and Yixin Chen. 2020. Hierarchical attention propagation for healthcare representation learning. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 249–256.

A Appendix

A.1 Global Text-Attributed KG Construction

We reuse the evidence-grounded TKG construction protocol from MedCo (Kerdabadi et al., 2026), where the induced relation schema and representative KG edges were clinically validated. Therefore, this work does not introduce a new KG induction pipeline; instead, it studies how to personalize the use of this validated global TKG for patient-specific EHR prediction.

We construct a global text-attributed knowledge graph (KG) over the medical code vocabulary. Let $\mathcal{C} = \mathcal{C}^{\text{dx}} \cup \mathcal{C}^{\text{rx}} \cup \mathcal{C}^{\text{px}}$ denote the set of diagnosis, medication, and procedure codes. The resulting global graph is denoted as

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \phi, \psi), \quad (32)$$

where $\mathcal{V} = \mathcal{C}$ is the set of medical concept nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of directed edges. The mapping $\phi : \mathcal{V} \rightarrow \mathcal{T}_c$ assigns each node to a clinical type, where $\mathcal{T}_c = \{\text{dx}, \text{rx}, \text{px}\}$, and $\psi : \mathcal{E} \rightarrow \mathcal{R}$ assigns each edge to a semantic relation type. For an edge $e = (u, v) \in \mathcal{E}$, we write $\psi(e) = r$ and equivalently denote the typed relation as (u, r, v) . Each node v is associated with a textual concept description d_v , and each edge e is associated with a textual relation rationale η_e .

Candidate pair extraction. Each patient record is represented as a sequence of visits $\{V_{p,1}, \dots, V_{p,T_p}\}$, where each visit contains a de-duplicated set of medical codes. We extract candidate code pairs using two complementary EHR-derived evidence channels. The first channel captures intra-visit co-occurrence, where two codes appear in the same visit. The second channel captures next-visit temporal transitions, where a source code appears in $V_{p,t}$ and a target code appears in $V_{p,t+1}$. These two channels allow the global KG to encode both simultaneous clinical associations and temporally ordered dependencies.

Pairwise statistical evidence. For each candidate directed pair (c_i, c_j) and evidence channel $m \in \{\text{co}, \text{temp}\}$, we compute support counts, a smoothed conditional probability, a PMI-style association score, and a statistical significance score. The smoothed conditional probability is

$$P_m(c_j | c_i) = \frac{x_m(c_i, c_j) + \alpha}{x_m(c_i) + \alpha|\mathcal{C}|}, \quad (33)$$

where $x_m(c_i, c_j)$ is the pair count under channel m , $x_m(c_i)$ is the corresponding source-code count, and α is a Laplace smoothing constant. We also compute a PMI-style score:

$$\text{PMI}_m(c_i, c_j) = \log \frac{p_m(c_i, c_j)}{p_m^{\text{src}}(c_i)p_m^{\text{tgt}}(c_j)}. \quad (34)$$

Here, $p_m(\cdot)$ denotes empirical probabilities estimated from the corresponding counts. For intra-visit co-occurrence, $p_m^{\text{src}} = p_m^{\text{tgt}}$, while for temporal transitions the source and target marginals are computed from adjacent visits. We further compute a χ^2 test of dependence from a 2×2 contingency table. To avoid data leakage, all EHR-derived co-occurrence and temporal-transition statistics used to construct the global TKG are computed only from the training split within each fold.

Evidence filtering. For each pair, we consolidate the co-occurrence and temporal statistics into a compact evidence summary:

$$\mathbf{a}_{ij} = [\mathbf{a}_{ij}^{\text{co}}; \mathbf{a}_{ij}^{\text{temp}}], \quad (35)$$

where each channel-specific vector contains support, conditional probability, PMI, and the χ^2 significance score. Candidate pairs with insufficient support, weak association, or non-significant dependence are removed before LLM-based relation induction. This filtering step prevents assigning semantic relations to arbitrary code pairs and ensures that the candidate set is grounded in empirical EHR evidence.

Type-constrained LLM relation induction. Each remaining candidate pair is passed to a type-constrained LLM relation induction module. For a pair (c_i, c_j) , the allowed relation set is selected according to the source and target code types:

$$\mathcal{R}_{ij} = \mathcal{R}_{\phi(c_i), \phi(c_j)}. \quad (36)$$

For example, diagnosis–medication pairs may use relations such as `treats`, `prevents`, `contraindicated_for`, and `causes_adverse_event`, while diagnosis–diagnosis pairs may use relations such as `causes`, `risk_factor_for`, `leads_to`, and `complicates`. Ambiguous cases are handled by conservative abstention labels such as `no_significant_relation` and `cannot_decide`.

The LLM prompt includes the two code identifiers, code names, clinical types, parent categories,

marginal frequencies, the EHR-derived evidence summary $\mathbf{a}_{ij}$, a short glossary of the statistical metrics, and the allowed type-specific relation set $\mathcal{R}_{ij}$. The LLM is instructed to return a structured output containing a directed relation label, an oriented triple, a confidence score, and a short clinical rationale. We use the confidence score and format checks for quality control, while the final graph stores the selected relation type through $\psi(e)$ and the textual rationale as the edge text η_e .

Node and edge textual attributes. After relation induction, we enrich the graph with textual attributes. For each node $v \in \mathcal{V}$, we attach a concise clinical description d_v generated using a type-specific prompt. The description summarizes the medical meaning of the code, such as disease presentation for diagnoses, therapeutic role for medications, or clinical purpose for procedures.

For each retained edge $e = (u, v) \in \mathcal{E}$, we attach the LLM-generated relation rationale η_e as the edge text. Thus, the text-attributed graph contains both node-level concept semantics and edge-level relational semantics:

$$v \mapsto d_v, \quad e \mapsto \eta_e. \quad (37)$$

The statistical evidence is used during graph construction and relation induction, but in the current formulation the edge attribute used by the downstream model is the textual rationale η_e .

Directionality. The induced edges are directed and preserve their clinical orientation. For example, if the LLM assigns a relation indicating that a medication treats a diagnosis, the resulting edge direction follows the induced semantic triple. For message passing, we add inverse computational edges during local graph materialization to support bidirectional information flow in the GNN. They do not change the semantic direction of the original global KG relation.

Use in patient-specific temporal graph construction. The global graph $\mathcal{G}$ serves as the source graph from which patient-specific temporal graphs are extracted. Given a patient p and a prediction cutoff time t^* , only codes observed up to t^* are used as root nodes. The selected neighborhoods of these roots provide relational KG context for the patient graph. The node descriptions $\{d_v\}$ and edge rationales $\{\eta_e\}$ provide the textual semantics used to initialize or generate the graph-contextualized code and KG soft prompts in REFINE.

A.2 LLM-Based Node Description Construction

In addition to relation-typed edges, we enrich each medical concept node with a textual description following the node-level enrichment strategy of Kerdabadi et al. (2026). For each node $v \in \mathcal{V}$, corresponding to a medical code c , we generate a concise clinical description d_v using a type-specific LLM prompt:

$$d_v = \text{LLM}(\mathcal{P}_{\text{desc}}(c, \phi(v))), \quad (38)$$

where $\phi(v) \in \mathcal{T}_c$ denotes the code type, and $\mathcal{P}_{\text{desc}}$ is a prompt template specialized for diagnosis, medication, and procedure codes.

The prompt provides the code identifier, code name, clinical type, and available parent category information, and instructs the LLM to generate a single clinically focused paragraph. The type-specific guidance asks for different information depending on the code category: diagnosis descriptions summarize the condition definition, typical presentation, risk factors, and clinical management; procedure descriptions summarize clinical purpose, indications, contraindications, and peri-procedural considerations; and medication descriptions summarize pharmacologic class, mechanism, indications, adverse effects, and common interaction patterns. To avoid unsupported or overly specific content, the prompt asks the LLM to avoid patient-level facts, URLs, dosage thresholds, hospital policies, and uncertain claims.

The LLM output is parsed as structured JSON and the generated description is attached to the corresponding node as its textual attribute:

$$v \mapsto d_v. \quad (39)$$

These node descriptions provide fixed text-derived clinical semantics used by REFINE to initialize code representations and construct graph-aware soft prompts. Together with the edge rationales η_e generated during the relation induction, they make $\mathcal{G}$ a relation-typed and text-attributed medical KG.

A.3 Implementation Details

We map ICD diagnosis/procedure codes to CCS categories and medications from National Drug Codes (NDC) to the Anatomical Therapeutic Chemical (ATC) classification. We use Llama-3.2-1B (Grattafiori and et al., 2024; Meta, 2024) as the frozen LLM backbone. All reported results are averaged over five-fold experiments using the same preprocessing and evaluation protocol.

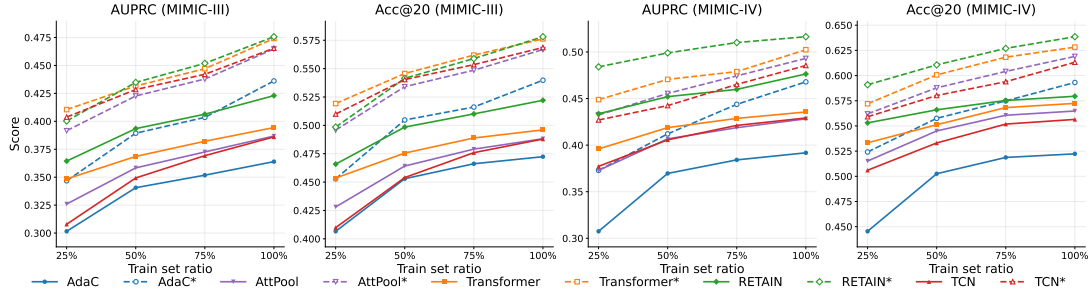


Figure 6: Data insufficiency analysis on MIMIC-III and MIMIC-IV under different training-set ratios. We compare each backbone model with its REFINE-enhanced variant across AUPRC and Acc@20. Solid lines denote the original backbone models, while dashed lines with hollow markers denote the corresponding models augmented with REFINE. Across datasets, metrics, and backbone architectures, REFINE consistently improves performance, demonstrating its robustness under limited supervision.

Hyperparameters. The heterogeneous GNN uses GAT-style message passing with hidden dimension 128, 4 layers, 2 attention heads, dropout 0.4, pooled-code patient representation, and bidirectional KG message passing. We run the frozen LLM in bfloat16 precision and do not update its tokenizer, embedding table, or backbone parameters. The trainable components include the heterogeneous GNN, residual code and relation embeddings, text-semantic projection layers, KG-context pooling module, soft-token projections, GNN-LLM fusion layer, downstream EHR backbone, and prediction head. We use graph-aware mixed prompts with hard text tokens for the task and visit-code scaffold, and two soft tokens per observed code: one code soft token and one KG-context soft token.

We train with Adam using a learning rate of 5×10^{-4} , weight decay 0, and batch size 256. In the first-stage graph-budgeting phase, we warm up the proxy GNN for 120 epochs and then train the sequential RL selector for up to 50 epochs with early stopping after at least 5 epochs, patience 5, and minimum improvement 10^{-5} on the mean RL reward. The RL selector uses additive transitions with minimum and maximum decision steps of 2 and 5, respectively. The KG cost weights are $\omega_C = 0.05$ for selected nodes and $\omega_E = 0.5$ for selected edges, with cost penalty $\lambda_{\text{cost}} = 10^{-4}$. The actor, critic, and entropy weights are 0.1, 0.1, and 0.02, respectively. During RL training, actions are sampled for exploration, while validation and inference use greedy selection. The task model is frozen during RL selector training, and selected patient graphs are exported for second-stage full REFINE training.

Two-stage training pipeline. Algorithm 1 summarizes the two-stage training pipeline of REFINE. We train REFINE in two stages to separate discrete graph-budget learning from the more expensive LLM-based semantic refinement. In the first stage, we train a personalized graph-budgeting policy using a GNN-only proxy predictor, avoiding repeated LLM rollouts during discrete graph selection. For each patient p , let $\mathcal{S}_p = \bigcup_{t \leq t^*} V_{p,t}$ denote the set of distinct observed code roots up to the prediction cutoff. The policy selects from a discrete action grid $a = (K, \mathbf{m})$, where K is the KG expansion depth and $\mathbf{m} = (m_1, \dots, m_K)$ specifies the per-hop fanout. The grid includes the no-expansion action $a_0 = (0, \emptyset)$ and bounded expansions up to $K_{\max} = 2$.

In Algorithm 1, A_p denotes the current action map that assigns each root code $c \in \mathcal{S}_p$ to an expansion action $a \in \mathcal{A}$. At RL step ℓ , $\mathcal{U}_\ell$ denotes the feasible macro-action set, where each macro-action $b = (c, a)$ updates the expansion action for one distinct observed code root. The operator $\text{EXPAND}(\mathcal{S}_p, A_p, \mathcal{G})$ materializes the selected patient graph $\mathcal{G}_p^{\text{sel}}$ from the global TKG $\mathcal{G}$ according to the action map A_p . We use $C_{\text{kg}}(A_p)$ to denote the induced KG cost, B_{kg} for the graph budget, $L_{\max}$ for the maximum number of RL decisions, and $\mathcal{M}$ for the exported selected-graph cache. The process terminates when the policy selects the stop action, reaches $L_{\max}$, or exhausts the graph budget.

The reward is computed from the downstream GNN-only prediction loss improvement relative to the no-expansion graph, with an additional penalty for selected KG nodes and edges. During training, actions are sampled from the actor for exploration; during validation and inference, we use greedy selection. The best selected patient-specific graphs are exported and cached in $\mathcal{M}$.

In the second stage, we fix the selected patient graphs and train the full REFINE architecture for next-visit diagnosis prediction. For each patient p , GNNENC($\cdot$) produces structural representations H_p^G , LLMREFINE($\cdot$) produces frozen LLM-refined semantic representations H_p^L , and FUSE($\cdot$) combines them into personalized code representations $Z_p = \{z_{p,t,c}\}$. The LLM remains frozen, and only the trainable REFINE components are updated, including the heterogeneous GNN, residual code and relation embeddings, text-semantic projection layers, KG-context pooling module, soft-token projections, GNN-LLM fusion layer, downstream EHR backbone, and prediction head. This design separates graph-budget learning from LLM-based semantic refinement while preserving patient-personalized code representations from the selected graph context.

A.4 KG-Context Pooling and Prompt Design

For each patient-code pair (p, c) , REFINE constructs a selected code-specific KG subgraph $\mathcal{G}_{p,c}^{\text{kg}}$. The goal of KGPool($\cdot$) is to summarize this selected relational context as a single KG-context soft token $\mathbf{k}_{p,c}$ that can be inserted into the frozen LLM prompt.

We considered several implementations, including mean pooling, gated attention, and relation-aware aggregation over selected neighbor nodes and KG edges. The best-performing implementation uses a lightweight GNN-style aggregator. For each selected KG edge $(a, b) \in \mathcal{E}_{p,c}^{\text{kg}}$, we first combine the LLM-branch representation of the neighbor node with the corresponding relation/rationale-aware edge representation. These edge-aware messages are then aggregated over the selected code-specific subgraph to produce a compact context vector, which is projected into the LLM hidden space as $\mathbf{k}_{p,c}$. This design preserves local relational information while avoiding the need to insert every selected KG triple as a separate prompt object.

We also explored a more explicit relational prompt format inspired by recent relational LLM prompting work, where relational structures are serialized in a JSON-like format and graph-derived embeddings are inserted as soft prompt tokens. Rel-LLM, for example, argues that flat text serialization can obscure relational structure, introduce redundancy, and exceed LLM context limits, and instead uses graph encoders and structured prompts to preserve relational information. Following this intu-

ition, we adapted a triplet-style prompt in which selected KG evidence was represented as soft-tokenized triples of the form (node, edge, node) inside our JSON visit-code hierarchy. However, this design substantially increased prompt length and the number of soft tokens, especially for patients with many observed codes and multi-hop KG expansions. Empirically, this led to higher memory and computational costs and weaker performance than the compact KGPool design. Therefore, REFINE uses KGPool to summarize each selected code-specific KG neighborhood into one KG-context token, balancing relational expressiveness and prompt efficiency.

A.5 Value-of-Information KG Selection

As an alternative to fixed KG expansion and sequential RL budgeting, we consider a learned value-of-information (VoI) selection strategy. The goal is to predict the KG expansion action expected to provide the largest predictive benefit relative to a no-expansion baseline. We use the same discrete hop-fanout action grid as the RL policy: $(0, \emptyset)$, $(1, (1))$, $(1, (2))$, $(1, (5))$, $(1, (10))$, $(2, (1, 1))$, $(2, (2, 1))$, $(2, (2, 2))$, $(2, (5, 1))$, and $(2, (5, 5))$.

VoI uses a learned critic to score candidate expansion actions at the visit level. During training, sparse utility targets are constructed from training labels. Let $\mathcal{G}_p^0$ denote the baseline patient graph in which all visits use the no-expansion action. For a candidate action $a \in \mathcal{A}$ at visit t , let $\mathcal{G}_p^{(t,a)}$ denote the graph in which visit t uses action a while all other visits remain at no expansion. Given a prediction loss $\mathcal{L}_p(\cdot)$, the target utility is defined as

$$\Delta_{p,t}(a) = \mathcal{L}_p(\mathcal{G}_p^0) - \mathcal{L}_p(\mathcal{G}_p^{(t,a)}). \quad (40)$$

These sparse utility targets supervise the VoI critic, which predicts a utility score $\hat{\Delta}_{p,t}(a)$ for each candidate action. At validation and test time, ground-truth labels are not used for KG selection. Instead, the selected VoI action is

$$\hat{a}_{p,t} = \arg \max_{a \in \mathcal{A}} \hat{\Delta}_{p,t}(a). \quad (41)$$

Unlike fixed expansion, VoI adapts KG expansion across visit contexts and performs label-free selection at inference. However, its visit-level decisions are made independently rather than sequentially conditioning on KG context already selected for other code roots. In contrast, the RL policy makes sequential code-level decisions conditioned

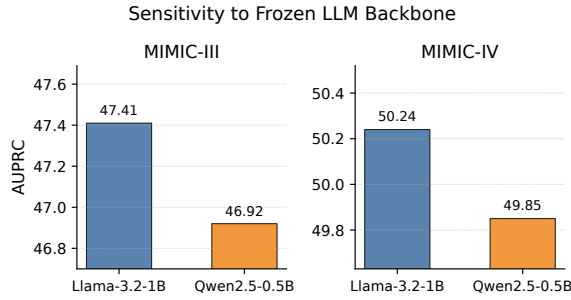


Figure 7: Sensitivity of REFINE to the frozen LLM backbone on MIMIC-III and MIMIC-IV. REFINE remains stable when replacing Llama-3.2-1B with the smaller Qwen2.5-0.5B.

on the patient state, remaining budget, and previously selected KG context, enabling finer-grained personalized graph construction.

A.6 RQ5: Data Insufficiency Analysis

To evaluate the robustness of REFINE under limited supervision, we conduct a data insufficiency analysis by varying the fraction of the available training set. This setting simulates practical clinical scenarios where labeled EHR data are limited. As shown in Figure 6, REFINE consistently improves downstream prediction performance across different training-set ratios on both MIMIC-III and MIMIC-IV. The improvements are observed across multiple backbone models, including AdaCare, AttPool, Transformer, RETAIN, and TCN, indicating that REFINE functions as a general plug-in refinement module rather than being specific to one architecture. The gains are especially meaningful in low-data regimes, where base EHR models have limited supervision to learn reliable representations for sparse and rare medical codes. By incorporating patient-specific graph context and semantic refinement, REFINE provides additional inductive bias that helps improve both AUPRC and Acc@20. Importantly, the enhanced models remain stronger than their base counterparts even at the full-data setting, suggesting that REFINE provides complementary information beyond what is learned from labeled sequences alone. Overall, these results show that REFINE is robust to data insufficiency and can improve EHR prediction performance across datasets, architectures, and supervision levels.

A.7 Sensitivity to LLM Backbone

We evaluate the sensitivity of REFINE to the frozen LLM backbone by comparing

Llama-3.2-1B (Grattafiori et al., 2024; Meta, 2024) with the smaller Qwen2.5-0.5B (Hui et al., 2024; Qwen Team, 2024). As shown in Figure 7, replacing Llama-3.2-1B with Qwen2.5-0.5B leads to a moderate decrease in AUPRC on both MIMIC-III and MIMIC-IV, but the overall performance remains competitive. This suggests that REFINE can benefit from its graph-contextual refinement mechanism across different small frozen LLM backbones. However, this analysis is limited to relatively small models, and future work should evaluate larger LLM backbones to better understand how backbone capacity affects semantic refinement and downstream EHR prediction.

Algorithm 1 Two-stage training of REFINE

Require: Training set $\mathcal{D}$, global TKG $\mathcal{G}$, action set $\mathcal{A}$, null action a_0 , graph budget B_{kg} , minimum/maximum decision steps $L_{\min}, L_{\max}$

Ensure: Trained parameters Θ

Stage I: RL-based graph budgeting

```
1: Initialize and warm up GNN-only proxy predictor; freeze its parameters
2: Initialize actor-critic policy  $(\pi_\eta, V_\omega)$ 
3: for epoch = 1, ...,  $E_1$  do
4:   for  $p \in \mathcal{D}$  do
5:      $\mathcal{S}_p \leftarrow \bigcup_{t \leq t^*} V_{p,t} \triangleright$  distinct observed code roots
6:      $A_p \leftarrow \{a_c \leftarrow a_0 : c \in \mathcal{S}_p\}$ 
7:      $\mathcal{G}_p^{\text{sel}} \leftarrow \text{EXPAND}(\mathcal{S}_p, A_p, \mathcal{G})$ 
8:     for  $\ell = 1, \dots, L_{\max}$  do
9:       if  $C_{\text{kg}}(A_p) \geq B_{\text{kg}}$  then
10:        break
11:       end if
12:        $\mathcal{U}_\ell \leftarrow \text{FEASIBLE}(\mathcal{S}_p, A_p, \mathcal{A}, B_{\text{kg}}) \triangleright$ 
         additive, budget-feasible updates
13:       if  $\ell \geq L_{\min}$  then
14:          $\mathcal{U}_\ell \leftarrow \mathcal{U}_\ell \cup \{\text{STOP}\}$ 
15:       end if
16:       if  $\mathcal{U}_\ell = \emptyset$  then
17:        break
18:       end if
19:       Select  $b_\ell \sim \pi_\eta(\cdot \mid s_\ell)$  over  $\mathcal{U}_\ell$ 
20:       if  $b_\ell = \text{STOP}$  then
21:        break
22:       end if
23:       Let  $b_\ell = (c_\ell, a_\ell)$ ; update  $A_p[c_\ell] \leftarrow a_\ell$ 
24:        $\mathcal{G}_p^{\text{sel}} \leftarrow \text{EXPAND}(\mathcal{S}_p, A_p, \mathcal{G})$ 
25:     end for
26:      $R_p \leftarrow L_p(\mathcal{G}_p^0) - L_p(\mathcal{G}_p^{\text{sel}}) - \lambda_{\text{cost}} C_{\text{kg}}(A_p)$ 
27:     Update  $(\pi_\eta, V_\omega)$  using the actor-critic objective
         with reward  $R_p$ 
28:   end for
29: end for
30: Export graph cache  $\mathcal{M}$  by running the learned policy greedily for each  $p \in \mathcal{D}$ 
Stage II: Full REFINE training
31: Initialize full REFINE with frozen LLM parameters
32: for epoch = 1, ...,  $E_2$  do
33:   for minibatch  $\mathcal{P} \subset \mathcal{D}$  do
34:     for  $p \in \mathcal{P}$  do
35:       Retrieve  $\mathcal{G}_p^{\text{sel}}$  from cache  $\mathcal{M}$ 
36:        $H_p^G \leftarrow \text{GNNENC}(\mathcal{G}_p^{\text{sel}})$ 
37:        $H_p^L \leftarrow \text{LLMREFINE}(\mathcal{G}_p^{\text{sel}})$ 
38:        $Z_p \leftarrow \text{FUSE}(H_p^G, H_p^L)$ 
39:        $\hat{y}_p \leftarrow f_\Omega(Z_p)$ 
40:     end for
41:     Update  $\Theta$  using  $\sum_{p \in \mathcal{P}} \mathcal{L}_{\text{BCE}}(\hat{y}_p, y_p)$ 
42:   end for
43: end for
44: return  $\Theta$ 
```
