AutoMap: Automatic Medical Code Mapping for Clinical Prediction Model Deployment

Zhenbang Wu[1], Cao Xiao[2], Lucas M. Glass[3], David M. Liebovitz[4], and Jimeng Sun[1]✉

[1] University of Illinois at Urbana-Champaign, {zw12,jimeng}@illinois.edu
[2] Amplitude, danica.xiao@amplitude.com
[3] IQVIA, lucas.glass@iqvia.com
[4] Northwestern University, david.liebovitz@nm.org

Abstract. Given a deep learning model trained on data from a source hospital, how to deploy the model to a target hospital automatically? How to accommodate heterogeneous medical coding systems across different hospitals? Standard approaches rely on existing medical code mapping tools, which have several practical limitations.

To tackle this problem, we propose AutoMap to automatically map the medical codes across different EHR systems in a coarse-to-fine manner: **(1) Ontology-level Alignment:** We leverage the ontology structure to learn a coarse alignment between the source and target medical coding systems; **(2) Code-level Refinement:** We refine the alignment at a fine-grained code level for the downstream tasks using a teacher-student framework.

We evaluate AutoMap using several deep learning models with two real-world EHR datasets: eICU and MIMIC-III. Results show that AutoMap achieves relative improvements up to 3.9% (AUC-ROC) and 8.7% (AUC-PR) for mortality prediction, and up to 4.7% (AUC-ROC) and 3.7% (F1) for length-of-stay estimation. Further, we show that AutoMap can provide accurate mapping across coding systems. Lastly, we demonstrate that AutoMap can adapt to two challenging scenarios: (1) mapping between completely different coding systems and (2) between completely different hospitals.

Keywords: Medical code mapping · Clinical prediction model deployment · Electronic health records

1 Introduction

Deep learning models have been widely used in clinical predictive modeling with electronic health record (EHR) data [33]. These models often leverage medical codes as an important data source summarizing patients' health status [5, 7, 14]. However, in real-world clinical practice, a variety of different coding systems are used across hospital EHR systems [3]. As a result, models trained on data from a source hospital are often hard to adapt to a target hospital where other coding systems are used. A method that can **accommodate different medical**

coding systems across hospitals for easy model deployment is highly desirable. Standard approaches rely on existing medical code mapping tools (e.g., Unified Medical Language System (UMLS) [4]), which have significant practical limitations due to the following challenges:

- **Rare coding systems.** Existing commercial and free code mapping tools are only available for a few widely used coding systems (e.g., ICD-9, ICD-10 and SNOMED CT) [32]. Hospitals using rare or private coding systems cannot benefit from these tools.
- **Limited labeled data.** While large hospitals may fine-tune the pre-trained models to adapt to their coding systems, small hospitals with limited labeled data often fail to do so.
- **No access to source data.** Even worse, the source data usually cannot be shared with the target hospital due to privacy and legal concern.

In this paper, we propose `AutoMap` for automatic medical code mapping across different hospitals EHR systems. `AutoMap` constructs appropriate target embeddings unsupervisedly based on the target EHR data and maps the target embeddings to the source embeddings, so that the deep learning model trained on the source data can be seamlessly deployed to the target data without any manual code mapping. More specifically, `AutoMap` learns the mapping across different coding systems in a coarse-to-fine manner:

- **Embedding.** The medical code embeddings will be constructed from the target EHR data unsupervisedly.
- **Ontology-level Alignment.** We leverage the ontology structure to map medical coding groups via iterative self-supervised learning. In this step, we obtain a coarse mapping from groups of target embeddings to the groups of source embeddings.
- **Code-level Refinement.** We refine the coarse mapping at a fine-grained code level via a teacher-student framework. It utilizes a discriminator (teacher A) to align two coding systems at the code level, and the backbone model (teacher B) to optimize the mapping based on the final prediction.

We evaluate `AutoMap` using multiple backbone deep learning models on two real-world EHR datasets: eICU [25] and MIMIC-III [13]. Results show that with a limited set of labeled data, `AutoMap` achieves relative improvements up to 3.9% on AUC-ROC score and 8.7% on AUC-PR score for mortality prediction; and up to 4.7% on AUC-ROC score and 3.7% on F1 score for length-of-stay estimation. Further, we evaluate the mapping accuracy of `AutoMap` and show that `AutoMap` improves the best baseline method by 8.2% in similarity score and 11.3% on hit@10 score. Lastly, we demonstrate that `AutoMap` can still achieve acceptable results under two challenging scenarios: (1) mapping between completely different coding systems: the model is trained on diagnosis codes and deployed on medication codes; (2) mapping between completely different hospitals: the model is trained and deployed in hospitals from different regions.

It is important to note that we do not argue to completely replace existing code mapping tools. Instead, the main contribution of `AutoMap` is to **evaluate**

the potential of a novel approach to automatically learn the code mapping from clinical data, which provides a new direction to support model deployment across different medical coding systems, and complements existing code mapping tools.

2 Related Work

Medical Code Mapping Tools. There exists a variety of commercial and free tools for mapping across different EHR ecosystems. UMLS [4] provides the mapping among ICD-9, ICD-10 and SNOMED CT. Observational Medical Outcomes Partnership (OMOP) [12] and Fast Healthcare Interoperability Resources (FHIR) [20] define the standards for representing clinical data in a consistent format. Relying on these tools, some recent works try to support model deployment across hospitals by transforming the EHR data into a standard format [26, 31]. However, creating such tools requires a lot of domain knowledge and human labor [32]. These mapping tools are only available for widely-used coding systems and can be easily outdated due to code updates. To address this, `AutoMap` proposes to learn the code mapping from clinical data, which complements existing code mapping tools.

Cross-lingual Word Mapping. Our medical code mapping problem has some similarity to the cross-lingual research. Cross-lingual word mapping methods work by mapping the word embeddings in two languages to a shared space using translation pairs [1], shared tokens [29], adversarial learning [9], or the nearest neighbors of similarity distributions [2]. Inspired by [2], `AutoMap` also leverages the similarity distributions [22] to align medical codes. However, there are significant differences between EHR and natural languages: (1) medical codes often reside in a concept hierarchy; (2) medical codes are often noisier. To address this, instead of directly mapping medical codes, `AutoMap` adopts a coarse-to-fine method by first performing ontology-level alignment and then code-level refinement.

3 Preliminaries

We first define a few key concepts, and then present our setting in Def. 6. Detailed notations can be found in the appendix.

Definition 1 (EHR Dataset). *In EHR data, a patient has a sequence of visits:* $V_p = [v_p^{(1)}, v_p^{(2)}, \ldots, v_p^{(n_p)}]$*, where* n_p *is the number of visits of patient* p*. For model training, each patient has a label* $\mathbf{y}_p$ *(e.g., mortality or length-of-stay). We will drop the subscript* $_p$ *whenever it is unambiguous. Each visit of a patient is represented by its corresponding medical codes, specified by* $v^{(i)} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{m^{(i)}}\}$*, where* $m^{(i)}$ *is the total number of codes of the* i*-th visit. Each medical code* $\mathbf{c} \in \{0,1\}^{|\mathcal{C}|}$ *is a one-hot vector (i.e.,* $\|\mathbf{c}\|_1 = 1$*), where* $\mathcal{C}$ *denotes the set of all medical codes in the dataset.*

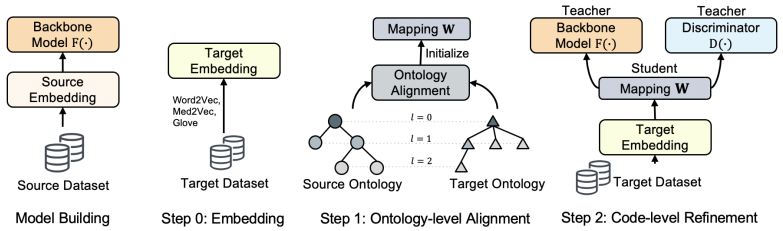

Fig. 1. `AutoMap` supports model deployment by automatically mapping the medical code embeddings across different coding systems in a coarse-to-fine manner: (0) Embedding that initializes the target code embedding matrix; (1) Ontology-level Alignment that leverages the ontology structure to learn the coarse ontology mapping; (2) Code-level Refinement that refines the mapping at the fine-grained code level for the downstream task with a teacher-student framework.

Our setting involves two datasets: a source dataset $*S$ for pre-training the backbone model but unavailable during deployment, and a mostly unlabeled target dataset $*T$ for deploying the model. The two datasets can have completely different medical codes. We also utilize separate medical ontology structures for source and target medical codes.

Definition 2 (Medical Ontology). *A medical ontology $\mathcal{O}$ specifies the hierarchy of medical codes in the form of a parent-child relationship. Formally, an ontology $\mathcal{O}$ is a directed acyclic graph whose nodes are $\mathcal{C} \cup \overline{\mathcal{C}}$. Here, $\mathcal{C}$ is the set of medical codes (often leaf nodes in the ontology), and $\overline{\mathcal{C}}$ is the set of other intermediate codes (i.e., non-leaf nodes) representing more general concepts.*

For simplicity, we define a function $\text{ancestor}(\mathbf{c}, l) : \{0,1\}^{|\mathcal{C}|} \times \mathbb{Z} \rightarrow \{0,1\}^{|\overline{\mathcal{C}}|}$, which maps a given medical code $\mathbf{c} \in \{0,1\}^{|\mathcal{C}|}$ to its l-th level ancestor code (i.e., category). For example, in Fig. 1, the root node is the 0-th level ancestor code of all leaf codes.

Definition 3 (Medical Code Embedding). *To fully utilize the code semantic information, it is a common practice to convert the medical code from one-hot vector $\mathbf{c} \in \{0,1\}^{|\mathcal{C}|}$ to a dense embedding vector $\mathbf{e} \in \mathbb{R}^d$ [5, 14], where d is the embedding dimensionality. This can be done via an embedding matrix $\mathbf{E} \in \mathbb{R}^{|\mathcal{C}| \times d}$, where each row corresponds to the embedding for a medical code. The embedding can be computed as $\mathbf{e} = \mathbf{E}^\top \mathbf{c}$.*

We denote the embedding matrices for source and target datasets as $\mathbf{E}_S$ and $\mathbf{E}_T$, respectively. The source embedding $\mathbf{E}_S$ is provided with the trained backbone model as the input. And the target embedding $\mathbf{E}_T$ will be learned using the target dataset. In this work, to deploy the backbone model, we will learn to map the target medical codes to the source.

Definition 4 (Code Embedding Mapping). *We define the mapping from the embedding space of one medical coding system to another as $\phi(\mathbf{E}) : \mathbb{R}^d \rightarrow \mathbb{R}^d$.*

We will learn the embedding mapping $\phi(\cdot)$ that maps the target embedding to the source via $\phi(\mathbf{E}_T)$.

Definition 5 (Backbone Deep Learning Model). *The backbone deep learning model $F(\cdot)$ takes EHR sequences and the corresponding medical code embeddings as the input and then outputs the prediction: $\hat{\mathbf{y}} = F([v^{(i)}]_{i=1}^n, \phi(\mathbf{E}))$, where $\hat{\mathbf{y}}$ is the corresponding predictions for label $\mathbf{y}$. The backbone model $F(\cdot)$ is pretrained on source dataset $*S$ and deployed on target dataset $*T$ with a different coding system. Note that the embedding mapping $\phi(\cdot)$ degenerates to the identity function if the backbone model $F(\cdot)$ is trained and deployed on the same coding system.*

Definition 6 (Predictive Model Deployment). *Given a backbone model $F(\cdot)$ and source code embedding matrix $\mathbf{E}_S$, a mostly unlabeled target dataset $*T$ in a different coding system, and the medical ontologies $\mathcal{O}_S, \mathcal{O}_T$ for both coding systems, the goal is to optimize the mapping $\phi(\cdot)$ on the target dataset $*T$, as given by Eq.* (1)*,*

$$\arg\min_{\phi(\cdot)} \mathcal{L}(F(\cdot), \mathbf{E}_S, *T, \mathcal{O}_S, \mathcal{O}_T, \phi(\cdot)), \tag{1}$$

where $\mathcal{L}(\cdot)$ denotes the designated loss function. The prediction on the target dataset can be obtained via $F(V_T, \phi(\mathbf{E}_T))$, where V_T is a sequence of visits from the target dataset $*T$, and $\phi(\mathbf{E}_T)$ is the transformed target embeddings.

In our setting, we can only access the source code embedding $\mathbf{E}_S$ and ontology $\mathcal{O}_S$ instead of the source data $*S$. This is more realistic in deployment setting since the source data often cannot be shared due to legal and privacy concern. In contrast, the source embedding matrix $\mathbf{E}_S$ can be more easily provided along with the backbone model $F(\cdot)$, and the code ontologies are usually publicly accessible. We also assume that the target dataset $*T$ is mostly unlabeled, since the target site may often be some small hospital.

4 AutoMap Method

We propose AutoMap for automatic code mapping across different hospitals EHR systems. The mapping will be done in a coarse-to-fine manner, enabled by the adaptation process shown in Fig. 1. Embedding (step 0) first initializes the target code embedding matrix $\mathbf{E}_T$. Ontology-level alignment (step 1) then derives the initial coarse mapping $\phi(\cdot)$ via iterative self-supervised learning. Code-level refinement (step 2) further fine-tunes the mapping $\phi(\cdot)$ at the code level using a teacher-student framework.

4.1 Step 0: Embedding

As mentioned in Def. 3, we first convert the target medical codes from one-hot vector $\mathbf{c}_T \in \{0,1\}^{|\mathcal{C}|}$ to a corresponding dense embedding vector $\mathbf{e}_T \in \mathbb{R}^d$.

We use GloVe [24] to learn the target code embedding matrix $\mathbf{E}_T$ via a global co-occurrence matrix of medical codes. Other unsupervised learning algorithms such as Med2Vec [7] and Word2Vec [21] can also be used. We employ GloVe because of its computational efficiency. After this, we parameterize $\phi(\cdot)$ by a mapping matrix $\mathbf{W} \in \mathbb{R}^{d \times d}$. The mapping matrix $\mathbf{W}$ can be used to transform the target code embedding via $\mathbf{E}_T\mathbf{W}$.

4.2 Step 1: Ontology-level Alignment

In this step, we will first learn a coarse mapping $\mathbf{W}$ at the ontology level. This first step is essential because direct code level mapping is difficult and unnecessary: (1) It is difficult due to the large number of medical codes; (2) It is also unnecessary since many codes have similar clinical meanings. Therefore, we follow a common practice to first group similar codes using code ontology [7, 5, 28] and learn the mapping on groups instead of leaf-level codes. For example, ICD-9 code 438.11 "late effects of cerebrovascular disease, aphasia" corresponds to five ICD-10 codes (I69.020, I69.120, I69.220, I69.320, I69.920). While it is hard to directly align the ICD-9 code to each of these five ICD-10 codes, we can first coarsely map the ICD-9 code to I00-I99 "diseases of the circulatory system", and then gradually refine the mapping to I60-I99 "cerebrovascular diseases", I69 "cerebrovascular diseases", and eventually the five-leaf codes. By leveraging the medical ontology, we can use more general medical concepts as "anchor points" to better align two coding systems.

Next, we introduce the building blocks of the iterative self-supervised learning (i.e., ontology grouping, unsupervised seed induction, Procrustes refinement), and then present the ontology-level alignment algorithm.

Ontology Grouping At a given hierarchy level l, we group the codes according to their l-th level ontology categories. Specifically, the i-th group $\mathcal{G}_i^{(l)}$ consists of all the leaf medical codes whose l-th level category is $\mathbf{c}_i$, as in Eq. (2),

$$\mathcal{G}_i^{(l)} = \{\mathbf{c}_j \mid \text{ancestor}(\mathbf{c}_j, l) = \mathbf{c}_i, \mathbf{c}_j \in \mathcal{C}\}, \tag{2}$$

where $\mathbf{c}_i \in \overline{\mathcal{C}}$ is the corresponding l-th level category code. We will drop the superscript $^{(l)}$ whenever it is unambiguous. To obtain the group embedding $\mathbf{g}_i$, we first calculate the mean group embedding $\overline{\mathbf{g}}_i$ by averaging all the code embeddings in that group, as in Eq. (3a); then, we represent the group embedding as the closest code embedding, as in Eq. (3b),

$$\overline{\mathbf{g}}_i = \text{mean}\{\mathbf{e}_j \mid \mathbf{c}_j \in \mathcal{G}_i\}, \tag{3a}$$

$$\mathbf{g}_i = \underset{\mathbf{e}_j}{\text{argmin}}\{\mathbf{e}_j\overline{\mathbf{g}}_i^\top \mid \mathbf{c}_j \in \mathcal{C}\}, \tag{3b}$$

where $\mathbf{e}_j$ is the embedding vector for the code $\mathbf{c}_j$, and $\mathbf{e}_j\overline{\mathbf{g}}_i^\top \in \mathbb{R}$ calculates the distance between the code $\mathbf{c}_j$ and the mean group embedding $\overline{\mathbf{g}}_i$. Intuitively, $\mathbf{g}_i$ can be viewed as the "median" group embedding. We select the top-k groups

based on the group size, since we want to first learn a coarse mapping while including too many groups may introduce too much granular information. As a result, we have $\mathbf{G}_T, \mathbf{G}_S \in \mathbb{R}^{k \times d}$ for target and source groups, where each row corresponds to an embedding vector for a particular group. We present with the same k to reduce clutter, though it can be different for source and target groups.

We note that when the ontology is not available, AutoMap can still apply by using a clustering algorithm (e.g., k-Means) to group the medical codes. Specifically, we provide additional experiments on this setting in the appendix.

Unsupervised Seed Induction Given the l-th level source and target coding groups $\mathbf{G}_S$ and $\mathbf{G}_T$, we can initialize a coarse alignment in a fully unsupervised way. More specifically, we first calculate the similarity matrices, as in Eq. (4),

$$\mathbf{M}_T = \mathbf{G}_T \mathbf{G}_T^\top;\ \mathbf{M}_S = \mathbf{G}_S \mathbf{G}_S^\top, \tag{4}$$

where $\mathbf{M}_T, \mathbf{M}_S \in \mathbb{R}^{k \times k}$. Each row in the similarity matrices $\mathbf{M}_T, \mathbf{M}_S$ represents the similarities of the corresponding group to all the other groups. Under the ideal case where the embedding spaces between different coding systems are isometric[5], one can permute the rows and columns of $\mathbf{M}_T$ to obtain $\mathbf{M}_S$. We introduce the following heuristics to find the optimal permutation (i.e., a mapping dictionary) of this NP-hard problem. We perform row-wise sort on $\mathbf{M}_T$ and $\mathbf{M}_S$ (i.e., elements in each row are sorted based only on the order in that particular row), as in Eq. (5a). Under the isometric assumption, codes with the same meaning will have exactly the same row vector in $\tilde{\mathbf{M}}_T$ and $\tilde{\mathbf{M}}_S$, suggesting that we can find the mapping dictionary $\mathbf{D} \in \mathbb{R}^{k \times k}$ via nearest neighbor search over row vectors in $\tilde{\mathbf{M}}_T$, as shown in Eq. (5b),

$$\tilde{\mathbf{M}}_T = \text{sorted}(\mathbf{M}_T); \tilde{\mathbf{M}}_S = \text{sorted}(\mathbf{M}_S), \tag{5a}$$

$$\mathbf{D}[i,j] = \begin{cases} 1, & \text{if } j = \text{argmax}((\tilde{\mathbf{M}}_T \cdot \tilde{\mathbf{M}}_S^\top)[i,:]) \\ 0, & \text{otherwise,} \end{cases} \tag{5b}$$

where $\cdot$ denotes matrix multiplication.

Procrustes Optimization At a given hierarchy level l, we optimize the inducted mapping dictionary $\mathbf{D}$ by iterating the following two steps.

1. The mapping $\mathbf{W} \in \mathbb{R}^{d \times d}$ is obtained by maximizing the similarities for the current dictionary $\mathbf{D}$, as given by Eq. (6a). This optimization problem is known as the Procrustes problem [27] and has a closed form solution, as in Eq. (6b),

$$\underset{\mathbf{W}}{\text{argmin}} \|\mathbf{D} \odot (\underbrace{\mathbf{G}_T \mathbf{W}}_{\text{transformed target embedding}} \mathbf{G}_S^\top)\|_1, \tag{6a}$$

$$\mathbf{W} = \mathbf{U}\mathbf{V}^\top, \text{ where } \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \text{SVD}(\mathbf{G}_T^\top \mathbf{D} \mathbf{G}_S), \tag{6b}$$

[5] In practice, the isometry requirement will not hold exactly, but it can be assumed to hold approximately, or the problem of mapping two code embedding spaces without supervision would be impossible.

where $\odot$ denotes Hadamard product, and SVD denotes Singular Value Decomposition.

2. A new dictionary $\mathbf{D}$ is induced, as in Eq. (7),

$$\mathbf{D}[i,j] = \begin{cases} 1, & \text{if } j = \text{argmax}((\mathbf{G}_T\mathbf{W}\mathbf{G}_S^\top)[i,:]) \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Iterative Self-supervised Learning We now introduce the self-supervised learning strategy, which maps the two coding systems at multiple resolutions iteratively. Starting from a coarse hierarchy level l, we obtain the l-th level medical coding groups $\mathbf{G}_S$ and $\mathbf{G}_T$ with Eq. (2, 3). Then we induct the l-th level seed mapping dictionary $\mathbf{D}^{(l)}$ with Eq. (4, 5). Next, we merge the current and previous level mapping dictionaries, as $\mathbf{D}^{(l)} = \mathbf{D}^{(l)} + \mathbf{D}^{(l-1)}$. Lastly, we optimize the merged mapping dictionary $\mathbf{D}^{(l)}$ using Eq. (6, 7). We gradually increase l (going down in the ontology) during iterative self-supervised learning until we reach the leaf level to learn the mapping at multiple resolutions. We note that source and target codes can use different grouping level l. We present with the same l to reduce clutter.

In this way, we learn a coarse mapping matrix $\mathbf{W}$ between two medical coding systems at the ontology level. This step is inspired by [2]. However, instead of directly mapping medical codes, `AutoMap` leverages the ontology structure and iteratively maps medical coding groups in a coarse-to-fine manner, allowing `AutoMap` to better align coding systems with different granularities.

4.3 Step 2: Code-level Refinement

While we have performed step 1 (ontology-level alignment) to initialize the mapping, the mapping is still too coarse and need further refining. Moreover, there is no guarantee of the performance for the downstream tasks (i.e., mortality prediction and length-of-stay estimation). Thus, it is preferred to further fine-tune the mapping at the code level for downstream tasks.

To do this, we propose a teacher-student framework, where the discriminator $D(\cdot)$ (teacher A) refines the mapping matrix $\mathbf{W}$ (student) via adversarial learning; and the backbone model $F(\cdot)$ (teacher B) optimizes the mapping matrix $\mathbf{W}$ (student) based on the final prediction task. Below we describe the framework in detail.

Teacher A: Discriminator We leverage the adversarial learning framework by introducing a discriminator $D(\cdot)$, parameterized by a multi-layer neural network. Specifically, the discriminator $D(\cdot)$ tries to classify whether the embeddings are from the target (label 0) or source (label 1) embedding distributions. Formally, discriminator $D(\cdot)$ aims at minimizing the discriminator adversarial loss, as in Eq. (8),

$$\mathcal{L}_D = -\log(D(\mathbf{e}_S)) - \log(D(1 - \mathbf{e}_T\mathbf{W})), \tag{8}$$

where $\mathbf{e}_S$ ($\mathbf{e}_T$) represents the source (target) code embedding sampled randomly from the code embedding matrix $\mathbf{E}_S$ ($\mathbf{E}_T$), and $\mathbf{W}$ maps the target embedding to the source embedding space via $\mathbf{e}_T\mathbf{W}$.

The mapping matrix $\mathbf{W}$ acts as the generator and tries to deceive the discriminator $D(\cdot)$. Formally, we try to minimize the generator adversarial loss, as in Eq. 9,

$$\mathcal{L}_G = -\log(D(\mathbf{e}_T\mathbf{W})). \tag{9}$$

Theoretically, the discriminator $D(\cdot)$ and mapping matrix $\mathbf{W}$ learn to align two coding systems as an adversarial game. Since the minimization happens at the distribution level, we do not require code mapping pairs to supervise training.

Teacher B: Backbone Here, the backbone model $F(\cdot)$ is leveraged to optimize the ultimate prediction performance based on the transformed target code embeddings. Formally, we aim at minimizing the following classification loss

$$\mathcal{L}_{\text{cls}}(F([v^{(i)}]_{i=1}^n, \mathbf{E}_T\mathbf{W}), \mathbf{y}_T), \tag{10}$$

where the transformed target code embeddings $\mathbf{E}_T\mathbf{W}$ are used to encode patient visits $[v^{(i)}]_{i=1}^n$.

In summary, the mapping matrix $\mathbf{W}$ is fine-tuned by minimizing the combined loss

$$\mathcal{L}_W = \mathcal{L}_{\text{cls}} + \alpha\mathcal{L}_G, \tag{11}$$

where α is a hyper-parameter. The pseudo-code can be found in the appendix.

5 Experiment

5.1 Experimental Setting

We will briefly introduce the experimental settings. Detailed information can be found in the appendix. The code of AutoMap is publicly available[6].

Data We evaluate the performance of AutoMap extensively with two publicly accessible datasets: eICU [25] and MIMIC-III [13]. eICU [25] is a multi-center database with intensive care unit (ICU) records for over 200K admissions to over 200 hospitals across the United States. MIMIC-III [13] is a single-center database containing 53K ICU records from Beth Israel Deaconess Medical Center.

Baselines We compare AutoMap with multiple baseline methods ranging from simple methods such as **Direct Training** and **Transfer Learning**, standard method leveraging **Mapping Tools**, to cross-lingual translation methods like **MUSE [9]** and **VecMap [2]**. We also conduct an ablation study of our AutoMap with **Step 1 Only**, **Step 1 Only + Random Ontology**, and **Step 2 Only**.

[6] https://github.com/zzachw/AutoMap

Backbone Models As `AutoMap` is a general framework that can apply to different backbone models, we incorporate `AutoMap` with the following backbone deep learning models: **MLP**, **RNN**, **RETAIN** [5], **GCT** [8], **BEHRT** [14].

Table 1. Results with limited labeled data (100 patients) in the target site. Dataset is eICU [25]. The average scores of two mapping directions between ICD-9 and ICD-10 codes are reported. * indicates that `AutoMap` achieves significant improvement over the best baseline method (i.e., p-value is smaller than 0.05). Experiment results show that `AutoMap` can adapt different backbone models to the target site with limited labeled data.

Backbone	**Method**	**Mortality**		**Length-of-Stay**	
		AUC-PR	AUC-ROC	F1	AUC-ROC
MLP	*Full-Label*	*0.2819*	*0.6531*	*0.5033*	*0.2819*
	Direct Training	0.2524	0.6191	0.2835	0.5345
	Transfer Learning	0.2551	0.6240	0.4584	0.6095
	MUSE	0.2506	0.6276	0.4905	0.6240
	VecMap	0.2820	0.6502	0.4947	0.6341
	`AutoMap`	**0.2934***	**0.6631***	**0.4952**	**0.6350**
RNN	*Full-Label*	*0.2818*	*0.6539*	*0.5030*	*0.2818*
	Direct Training	0.2074	0.5547	0.1222	0.4427
	Transfer Learning	0.2536	0.6234	0.4662	0.6166
	MUSE	0.2455	0.6260	0.4933	0.6367
	VecMap	0.2780	0.6488	**0.5019**	0.6416
	`AutoMap`	**0.2875***	**0.6627***	0.4996	**0.6487***
RETAIN	*Full-Label*	*0.2648*	*0.6190*	*0.4447*	*0.2648*
	Direct Training	0.2031	0.5466	0.1222	0.4427
	Transfer Learning	0.2269	0.5732	0.4455	0.5395
	MUSE	0.2374	0.5838	0.4217	0.5831
	VecMap	0.2744	0.6315	0.4264	0.5963
	`AutoMap`	**0.2835***	**0.6528***	**0.4779***	**0.6007***
GCT	*Full-Label*	*0.2814*	*0.6533*	*0.4986*	*0.2814*
	Direct Training	0.1836	0.5402	0.2680	0.4865
	Transfer Learning	0.2103	0.5967	0.4748	0.5718
	MUSE	0.2242	0.6016	0.4866	0.6129
	VecMap	0.2491	0.6291	0.4863	0.6085
	`AutoMap`	**0.2707***	**0.6539***	**0.4940***	**0.6363***
BEHRT	*Full-Label*	*0.2652*	*0.6673*	*0.3657*	*0.2652*
	Direct Training	0.1740	0.5438	0.3063	0.4730
	Transfer Learning	0.2320	0.6190	0.3291	0.5609
	MUSE	0.2155	0.6040	0.3493	0.5869
	VecMap	**0.2786**	**0.6740**	0.3612	0.6044
	`AutoMap`	0.2712	0.6737	**0.3744***	**0.6328***

Table 2. Results for the scenario where the backbone model is trained on diagnosis code and deployed on medication codes. Dataset is MIMIC-III [13]. Experiment results show that AutoMap can adapt to target data coded in a completely different system.

Method	Mortality AUC-PR	Length-of-Stay F1
Full-Label	*0.7149*	*0.3057*
Direct Training	0.4701	**0.3158**
Transfer Learning	0.5642	0.2999
MUSE	0.4905	0.3022
VecMap	0.3553	0.3014
AutoMap	**0.5902***	0.3022

5.2 Q1: Target Data with Limited Labels

We first evaluate AutoMap in a common setting where the target site has limited labeled data (100 patients). For reference, we also report the performance of the model trained with the fully-labeled target data, as *"Full-Label"* in the table. This can be viewed as an "upper bound" of the model performance. Descriptions of the metrics can be found in the appendix. Results can be found in in Tab. 1.

First, we find that the two simple baselines: direct training and transfer learning methods do not work very well. In most cases, they are much worse compared to the full-label performance. This is expected as the amount of labeled data is insufficient to train or fine-tune the backbone models. Next, code-level mapping methods MUSE [9] and VecMap [2] achieve some improvements, but they are not stable. In some cases, they perform even worse than the two simple baselines. This may because ICD-9 and ICD-10 have different degrees of specificity (e.g., 10K codes in ICD-9 v.s. 68K codes in ICD-10), and directly mapping them at code level does not work very well. Finally, we observe that AutoMap achieves significant improvement over the baseline and can match the full-label performance in most cases. Specifically, AutoMap achieves up to 8.7% relative improvement on AUC-PR score for mortality prediction; for length-of-stay estimation, AutoMap achieves up to and 3.7% relative improvement on F1 score. This demonstrates the effectiveness of coarse-to-fine mapping and the versatility of AutoMap.

5.3 Q2: Completely Different Codes

We then evaluate AutoMap in the challenging case where we train the backbone model on diagnosis code (ICD-9) and deploy it on medication codes (NDC). Due to the limited space, for the rest of the experiments, we only report AUC-PR for mortality and F1 for length-of-stay with backbone model BEHRT [14] using 100 labeled patients in the target data. Results can be found in in Tab. 2.

First, we note that since these two coding systems are so different, no existing mapping tools is available. For mortality prediction, as shown in Tab. 2, the code-level mapping methods perform even worse than direct training and transfer

Table 3. Results for the scenario where the backbone model is trained and deployed in hospitals from different regions. Dataset is eICU [25]. Experiment results show that `AutoMap` can adapt to target hospitals from a completely region.

Method	**Mortality** AUC-PR	**Length-of-Stay** F1
Full-Label	*0.2578*	*0.4560*
Direct Training	0.1434	**0.4334**
Transfer Learning	0.1860	0.3924
MUSE	0.1314	0.3988
VecMap	0.1305	0.3801
`AutoMap`	**0.1990***	0.4290

learning. This may due to the large gap between these two coding systems. On the contrary, `AutoMap` can still give acceptable results, outperforming all baseline methods with 4.6%−66.1% statistically significant improvements. This shows the superiority of `AutoMap`'s coarse-to-fine mapping strategy. For the length-of-stay estimation task, all five methods perform pretty similar to full-label performance. This may indicate that medication codes are not so informational for length-of-stay estimation.

5.4 Q3: Completely Different Hospitals

We next challenge `AutoMap` under the scenario where we train the backbone model in hospitals from Midwest region (with ICD-9 code) and deploy it in hospitals from South region (with ICD-10 code). Results can be found in in Tab. 3.

For mortality prediction, mapping based methods (MUSE [9] and VecMap [2]) achieve the worst results. This is expected as methods from cross-lingual word mapping do not consider the domain gap between different regions. This also explains why transfer learning perform slightly better (as its training scheme can accommodate some domain gap). Benefit from the refinement step, `AutoMap` achieves the best result with $7.0\% - 52.5\%$ statistically significant relative improvements. This shows that `AutoMap` can adapt to hospitals from different regions. For length-of-stay estimation, all pre-training based methods perform worse than direct training. This may indicate that different hospitals have different decision rules on ICU length-of-stay. As a result, transferring knowledge from other hospitals may not help. Despite this, `AutoMap` still achieves the best results among all pre-training based methods.

5.5 Q4: Mapping Accuracy

We further evaluate the accuracy of the learnt mapping. The ICD code mapping in the eICU [25] dataset is used as the ground truth. As shown in Tab. 4, VecMap [2] and `AutoMap` achieve much better performance than MUSE [9].

Table 4. Accuracy of mapping for diagnosis codes (ICD-9 and ICD-10). Dataset is eICU [25]. The average scores of two mapping directions are reported. Experiment results show that AutoMap can learn accurate mapping across medical coding systems.

Method	Similarity	Hit@10
MUSE	0.1633	0.0600
VecMap	0.4612	0.5974
AutoMap	**0.4992***	**0.6657***

Table 5. Ablation study. Dataset is eICU [25]. The average scores of two mapping directions between ICD-9 and ICD-10 codes are reported. R.O. denotes random ontology. Experiment results demonstrate the importance of AutoMap's 2-step coarse-to-fine mapping.

Method	Mortality AUC-PR	Length-of-Stay F1
Step 1 Only	0.2680	0.3623
Step 1 Only + R.O.	0.2054	0.3631
Step 2 Only	0.2038	0.3306
AutoMap	**0.2712***	**0.3744***

This supports the isometric assumption used in both methods. Further, AutoMap achieves the best results with statistical significance. This demonstrates that the proposed coarse-to-fine mapping can better map coding systems with different granularities.

5.6 Ablation study

Finally, we compare AutoMap with three ablated versions. As shown in Tab. 5, only performing step 2 (code-level refinement) gives the worst results. This is reasonable as the model will easily over-fit the target data with limited labels. Also, since the mapping matrix $\mathbf{W}$ is randomly generated, the adversarial learning module will even harm the downstream tasks. Next, we can see that performing step 1 (ontology-level alignment) only gives better results. This indicates that step 1 contributes most to AutoMap's improvements. This may because the isometric assumption and medical ontology can act as a strong prior to guide the model learning process. This point can also be supported by the performance with randomly-generated ontology. Lastly, AutoMap achieves the best results. This shows the importance of refining the mapping at code-level after the coarse ontology alignment.

6 Conclusion

We propose AutoMap for automatic medical code mapping across different hospitals EHR systems. Benefit from the coarse-to-fine mapping, AutoMap can better align coding systems at different granularities. We evaluate AutoMap extensively using different backbone models with two real-world EHR datasets. Experimental results show that AutoMap outperforms existing solutions on multiple prediction tasks when mapping solutions exist and provides a mapping strategy when conventional solutions do not exist.

References

1. Artetxe, M., Labaka, G., Agirre, E.: Learning bilingual word embeddings with (almost) no bilingual data. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 451–462. Association for Computational Linguistics, Vancouver, Canada (Jul 2017). https://doi.org/10.18653/v1/P17-1042, https://www.aclweb.org/anthology/P17-1042
2. Artetxe, M., Labaka, G., Agirre, E.: A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 789–798. Association for Computational Linguistics, Melbourne, Australia (Jul 2018). https://doi.org/10.18653/v1/P18-1073, https://www.aclweb.org/anthology/P18-1073
3. Birkhead, G.S., Klompas, M., Shah, N.R.: Uses of electronic health records for public health surveillance to advance public health. Annual Review of Public Health **36**(1), 345–359 (2015). https://doi.org/10.1146/annurev-publhealth-031914-122747, https://doi.org/10.1146/annurev-publhealth-031914-122747, pMID: 25581157
4. Bodenreider, O.: The Unified Medical Language System (UMLS): integrating biomedical terminology. Nucleic Acids Res **32**(Database issue), D267–270 (Jan 2004)
5. Choi, E., Bahadori, M.T., Kulas, J.A., Schuetz, A., Stewart, W.F., Sun, J.: Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. p. 3512–3520. NIPS'16, Curran Associates Inc., Red Hook, NY, USA (2016)
6. Choi, E., Bahadori, M.T., Schuetz, A., Stewart, W.F., Sun, J.: Doctor ai: Predicting clinical events via recurrent neural networks. In: Doshi-Velez, F., Fackler, J., Kale, D., Wallace, B., Wiens, J. (eds.) Proceedings of the 1st Machine Learning for Healthcare Conference. Proceedings of Machine Learning Research, vol. 56, pp. 301–318. PMLR, Northeastern University, Boston, MA, USA (18–19 Aug 2016), https://proceedings.mlr.press/v56/Choi16.html
7. Choi, E., Bahadori, M.T., Searles, E., Coffey, C., Thompson, M., Bost, J., Tejedor-Sojo, J., Sun, J.: Multi-layer representation learning for medical concepts. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 1495–1504. KDD '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2939672.2939823, https://doi.org/10.1145/2939672.2939823
8. Choi, E., Xu, Z., Li, Y., Dusenberry, M., Flores, G., Xue, E., Dai, A.: Learning the graphical structure of electronic health records with graph convolutional transformer. Proceedings of the AAAI Conference on Artificial Intelligence **34**, 606–613 (04 2020). https://doi.org/10.1609/aaai.v34i01.5400
9. Conneau, A., Lample, G., Ranzato, M., Denoyer, L., Jégou, H.: Word translation without parallel data (2018)
10. Gupta, P., Malhotra, P., Narwariya, J., Vig, L., Shroff, G.: Transfer learning for clinical time series analysis using deep neural networks (2019)
11. Harutyunyan, H., Khachatrian, H., Kale, D.C., Ver Steeg, G., Galstyan, A.: Multitask learning and benchmarking with clinical time series data. Scientific Data **6**(1) (Jun 2019). https://doi.org/10.1038/s41597-019-0103-9, http://dx.doi.org/10.1038/s41597-019-0103-9

12. Hripcsak, G., Duke, J.D., Shah, N.H., Reich, C.G., Huser, V., Schuemie, M.J., Suchard, M.A., Park, R.W., Wong, I.C., Rijnbeek, P.R., van der Lei, J., Pratt, N., Norén, G.N., Li, Y.C., Stang, P.E., Madigan, D., Ryan, P.B.: Observational Health Data Sciences and Informatics (OHDSI): Opportunities for Observational Researchers. Stud Health Technol Inform **216**, 574–578 (2015)
13. Johnson, A.E., Pollard, T.J., Shen, L., Lehman, L.H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L.A., Mark, R.G.: Mimic-iii, a freely accessible critical care database. Scientific data **3**, 160035 (2016)
14. Li, Y., Rao, S., Solares, J.R.A., Hassaine, A., Ramakrishnan, R., Canoy, D., Zhu, Y., Rahimi, K., Salimi-Khorshidi, G.: BEHRT: Transformer for Electronic Health Records. Scientific Reports **10**(1), 7155 (Dec 2020). https://doi.org/10.1038/s41598-020-62922-y, http://www.nature.com/articles/s41598-020-62922-y
15. Luo, J., Ye, M., Xiao, C., Ma, F.: HiTANet: Hierarchical Time-Aware Attention Networks for Risk Prediction on Electronic Health Records, p. 647–656. Association for Computing Machinery, New York, NY, USA (2020), https://doi.org/10.1145/3394486.3403107
16. Ma, F., Chitta, R., Zhou, J., You, Q., Sun, T., Gao, J.: Dipole: Diagnosis prediction in healthcare via attention-based bidirectional recurrent neural networks. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 1903–1911. KDD '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3097983.3098088, https://doi.org/10.1145/3097983.3098088
17. Ma, L., Gao, J., Wang, Y., Zhang, C., Wang, J., Ruan, W., Tang, W., Gao, X., Ma, X.: Adacare: Explainable clinical health status representation learning via scale-adaptive feature extraction and recalibration. In: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020. pp. 825–832. AAAI Press (2020), https://aaai.org/ojs/index.php/AAAI/article/view/5427
18. Ma, L., Ma, X., Gao, J., Zhang, C., Yu, Z., Jiao, X., Ruan, W., Wang, Y., Tang, W., Wang, J.: Covidcare: Transferring knowledge from existing emr to emerging epidemic for interpretable prognosis (2020)
19. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: in ICML Workshop on Deep Learning for Audio, Speech and Language Processing (2013)
20. Mandel, J.C., Kreda, D.A., Mandl, K.D., Kohane, I.S., Ramoni, R.B.: SMART on FHIR: a standards-based, interoperable apps platform for electronic health records. J Am Med Inform Assoc **23**(5), 899–908 (09 2016)
21. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space (2013)
22. Mikolov, T., Le, Q.V., Sutskever, I.: Exploiting similarities among languages for machine translation (2013)
23. Nguyen, P., Tran, T., Wickramasinghe, N., Venkatesh, S.: `Deepr`: A convolutional net for medical records. IEEE Journal of Biomedical and Health Informatics **21**(1), 22–30 (2017). https://doi.org/10.1109/JBHI.2016.2633963
24. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543 (2014), http://www.aclweb.org/anthology/D14-1162

25. Pollard, T.J., Johnson, A.E.W., Raffa, J.D., Celi, L.A., Mark, R.G., Badawi, O.: The eICU Collaborative Research Database, a freely available multi-center database for critical care research. Scientific Data **5**(1), 180178 (Sep 2018). https://doi.org/10.1038/sdata.2018.178, https://doi.org/10.1038/sdata.2018.178
26. Rajkomar, A., Oren, E., Chen, K., Dai, A.M., Hajaj, N., Hardt, M., Liu, P.J., Liu, X., Marcus, J., Sun, M., Sundberg, P., Yee, H., Zhang, K., Zhang, Y., Flores, G., Duggan, G.E., Irvine, J., Le, Q., Litsch, K., Mossin, A., Tansuwan, J., Wang, D., Wexler, J., Wilson, J., Ludwig, D., Volchenboum, S.L., Chou, K., Pearson, M., Madabushi, S., Shah, N.H., Butte, A.J., Howell, M.D., Cui, C., Corrado, G.S., Dean, J.: Scalable and accurate deep learning with electronic health records. npj Digital Medicine **1**(1), 18 (Dec 2018). https://doi.org/10.1038/s41746-018-0029-1, http://www.nature.com/articles/s41746-018-0029-1
27. Schönemann, P.H.: A generalized solution of the orthogonal procrustes problem. Psychometrika **31**(1), 1–10 (Mar 1966). https://doi.org/10.1007/BF02289451, https://doi.org/10.1007/BF02289451
28. Shang, J., Xiao, C., Ma, T., Li, H., Sun, J.: Gamenet: Graph augmented memory networks for recommending medication combination. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019. pp. 1126–1133. AAAI Press (2019). https://doi.org/10.1609/aaai.v33i01.33011126, https://doi.org/10.1609/aaai.v33i01.33011126
29. Søgaard, A., Ruder, S., Vulić, I.: On the limitations of unsupervised bilingual dictionary induction. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 778–788. Association for Computational Linguistics, Melbourne, Australia (Jul 2018). https://doi.org/10.18653/v1/P18-1072, https://www.aclweb.org/anthology/P18-1072
30. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research **15**(56), 1929–1958 (2014), http://jmlr.org/papers/v15/srivastava14a.html
31. Tang, S., Davarmanesh, P., Song, Y., Koutra, D., Sjoding, M.W., Wiens, J.: Democratizing EHR analyses with FIDDLE: a flexible data- driven preprocessing pipeline for structured clinical data. Journal of the American Medical Informatics Association **0**(0), 14 (2020)
32. Wojcik, B.E., Stein, C.R., Devore, R.B., Hassell, L.H.: The Challenge of Mapping between Two Medical Coding Systems. Military Medicine **171**(11), 1128–1136 (Nov 2006). https://doi.org/10.7205/MILMED.171.11.1128, https://academic.oup.com/milmed/article/171/11/1128-1136/4578127
33. Xiao, C., Choi, E., Sun, J.: Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review. Journal of the American Medical Informatics Association **25**(10), 1419–1428 (06 2018). https://doi.org/10.1093/jamia/ocy068, https://doi.org/10.1093/jamia/ocy068
34. Zhang, C., Gao, X., Ma, L., Wang, Y., Wang, J., Tang, W.: Grasp: Generic framework for health status representation learning based on incorporating knowledge from similar patients. Proceedings of the AAAI Conference on Artificial Intelligence **35**(1), 715–723 (May 2021), https://ojs.aaai.org/index.php/AAAI/article/view/16152

35. Zhang, H., Dullerud, N., Seyyed-Kalantari, L., Morris, Q., Joshi, S., Ghassemi, M.: An empirical framework for domain generalization in clinical settings. In: Proceedings of the Conference on Health, Inference, and Learning. p. 279–290. CHIL '21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3450439.3451878, https://doi.org/10.1145/3450439.3451878