MedTPE: Compressing Long EHR Sequence for LLM-based Clinical Prediction with Token-Pair Encoding

Mingcheng Zhu*
mingcheng.zhu@eng.ox.ac.uk
University of Oxford
Oxford, UK

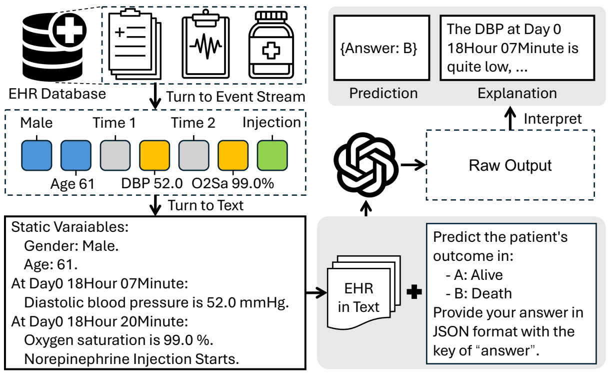
Zhiyao Luo
University of Oxford
Oxford, UK

Yu Liu
University of Oxford
Oxford, UK

Tingting Zhu
University of Oxford
Oxford, UK

Abstract

Electronic health records (EHRs) capture detailed, longitudinal patient data essential for healthcare. While large language models (LLMs) offer strong language understanding, their use in clinical applications with EHRs remains limited due to two key challenges: (1) the excessive input lengths of EHRs, which often exceed the context windows of even long-context LLMs; and (2) the high prevalence of domain-specific or out-of-distribution tokens in clinical text. To address these issues, we propose **Medical Token-Pair Encoding (MedTPE)**, a lightweight, domain-adaptive token compression method built on top of the Byte Pair Encoding (BPE) framework. MedTPE identifies frequently co-occurring token patterns in real-world EHR corpora and replaces them with medically coherent composite tokens, without increasing the vocabulary size or embedding parameters. Crucially, MedTPE modifies only 3.3% of the original BPE vocabulary using a dependency-aware replacement strategy, making it both efficient and compatible with existing pretrained models. Without requiring additional labels, MedTPE achieves up to 49% reduction in token sequence length on two clinical prediction tasks, while maintaining predictive performance across all evaluated settings. Our method offers a scalable solution for compressing EHR inputs, substantially enhancing the practicality of LLMs in clinical environments.



Static Varaiables:
Gender: Male.
Age: 61.
At Day0 18Hour 07Minute:
Diastolic blood pressure is 52.0 mmHg.
At Day0 18Hour 20Minute:
Oxygen saturation is 99.0 %.
Norepinephrine Injection Starts.

Predict the patient's outcome in:
- A: Alive
- B: Death
Provide your answer in JSON format with the key of "answer".

Figure 1: Illustration of LLM-based clinical prediction with EHR data. Structured EHRs are first converted into event streams and then translated into natural language, incorporating both static patient characteristics and time-stamped clinical events. The resulting text is concatenated with a prompt describing the prediction task. The combined text and prompt are tokenised and fed into the LLM, which generates output that is interpreted as both a structured prediction and a free-text explanation.

1 Introduction

Electronic health records (EHRs) contain comprehensive, longitudinal information about patient care, encompassing diagnoses, laboratory tests, medications, and procedures. Leveraging these multi-modal and temporally structured data sources is crucial for advancing predictive modelling and clinical decision support in healthcare systems [16, 22]. Especially, recent work has demonstrated that large language models (LLMs) can effectively perform a range of clinical prediction tasks in zero-shot and few-shot settings by representing EHRs as token sequences and learning patterns across patient timelines [7, 23]. The common paradigm involves converting structured EHR events into text and giving instructions for downstream tasks, as shown in Figure 1. In addition to the zero-shot predictive capability, LLMs are capable of generating human-readable explanations for clinical decisions [29].

To support reliable clinical predictions, LLMs must process extensive and complex patient histories. However, EHRs for a few days can lead to tokenised sequences over 64,000 tokens, far surpassing the context window of most existing LLM families [9, 27, 30]. This results in significant computational overhead and can reduce inference efficiency. A key contributing factor to this inefficiency lies in the subword tokenisation algorithms (e.g., Byte-Pair Encoding, WordPiece, and SentencePiece [15, 24, 25]). While these algorithms are effective for general-purpose language modelling, they are not optimised for the long and specialised medical terms commonly found in EHRs. As a result, compound clinical phrases are frequently fragmented into multiple short subword tokens, which unnecessarily elongates input sequences and increases computation [33].

One potential solution is to retrain a domain-specific tokeniser on medical corpora, which can yield more compact representations [4, 14, 23]. However, this approach often requires re-training or continuing pre-training from a base model to accommodate the new vocabulary, making it prohibitively resource-intensive. In addition, the retraining can destroy LLMs' core capabilities, such as instruction following and reasoning, that are already embedded in the pre-trained weights. Recent efforts modify the existing vocabularies through token remapping or adapter layers to avoid model retraining but introduce additional computational cost and integration complexity [10, 19]. Overall, current approaches in medical applications face significant challenges of resource demands, compatibility, and computational efficiency. There remains a pressing need for a medical-specific tokenisation approach that reduces sequence length while preserving compatibility with existing LLMs and maintaining inference efficiency.

To address these challenges, we propose MedTPE, an efficient tokenisation method built on top of any existing BPE tokeniser. Instead of discarding the pre-trained tokeniser, MedTPE improves it by merging frequently co-occurring BPE token pairs from real EHR corpora into new composite tokens. Unlike methods that simply add new tokens, MedTPE uses a dependency-aware replacement strategy to substitute a small portion of the original BPE tokens with the most informative new tokens. This enables significant compression without increasing the vocabulary size or embedding layer parameters. With only minimal fine-tuning of the newly introduced embeddings, MedTPE significantly reduces the length of the token sequence while maintaining, and sometimes even improving, predictive performance. Our main contributions are as follows:

- **Substantial token savings with near-zero performance drop.** MedTPE reduces input sequence lengths by up to 49% across two clinical prediction tasks in the publicly available MIMIC-IV dataset [13], with marginal degradation and even improvement in predictive performance.
- **Label-free adaptation.** MedTPE introduces a light-weighted token compression scheme that modifies only the embedding layer, requiring no labelled data and no re-training or fine-tuning of the full LLM. The embeddings for new tokens are initialised and adapted via lightweight self-supervised learning.
- **Efficient and compatible design.** MedTPE uses a dependency-aware replacement algorithm that replaces only 3.3% of original tokens without altering the vocabulary size or embedding matrix. This algorithm preserves the encoding mechanism of the BPE, making MedTPE fully compatible with any BPE framework and thus enabling seamless integration with pre-trained LLMs.

Overall, MedTPE offers a practical and scalable approach to using LLMs in real-world clinical environments, where efficiency and compatibility with existing models are critical.

2 Related Work

2.1 LLM-based Prediction on EHR

Recent studies have begun to exploit LLMs for clinical prediction from EHRs, evolved from time series models [6, 9, 21, 31]. Niu et al. [21] extract a targeted subset of medical events and laboratory results from EHR databases, rendering them as concise, narrative-style summaries for LLM input. Chen et al. [6] convert diagnosis, procedure, and medication codes into full descriptive sentences to enrich the clinical context of the model. In contrast, Fleming et al. [9] serialises the patient's entire event history into an XML-formatted text for direct ingestion. Similarly, Wu et al. [31] linearises all EHR events into a continuous text sequence and then adopts ClinicalBERT [1] to embed the events before feeding them into the LLM. Despite their success, these methods struggle with overly long token sequences. Niu et al. and Chen et al. reduce length by selecting a hand-picked subset of events, risking the omission of important history [6, 21]. Wu et al. employs per-event compression models to shorten descriptions, adding computational overhead [31]. Fleming et al. simply truncates the event stream, which speeds up inference, but discards a lot of the patient record [9]. In contrast, our method addresses the sequence length at the tokenisation level. By merging frequently co-occurring token pairs into single, medical tokens, we preserve the patient trajectory within the hospital while substantially reducing input length and inference cost.

2.2 Tokenisation

Modern LLMs employ subword tokenisation to strike a balance between a fixed-size vocabulary and the need to represent rare or out-of-vocabulary words. BPE begins with a base vocabulary of individual characters and repeatedly merges the most frequent pairs of adjacent characters into new sub-word units [24]. WordPiece extends this idea by scoring candidate merges according to their impact on the overall likelihood of a language model, resulting in a vocabulary that maximises predictability at the token level [25]. In contrast, SentencePiece treats tokenisation as inference under a probabilistic finite vocabulary: it starts from a large seed vocabulary and removes low-probability tokens iteratively, optimising for both coverage and compression without relying on explicit merge operations [15]. These general-purpose tokenisers excel on everyday text, but can over-segment specialised terminology. In the medical domain, long drug names and ontological identifiers can be split into dozens of sub-words, increasing the length of the sequence, computing the cost, and affecting the semantic cohesion [12]. Whereas conventional approaches map text to tokens, we reverse the lens: TPE merges high-utility token pairs from the medical corpus into single domain-specific units. This compresses trajectories without discarding the mature BPE vocabulary or the checkpoints that depend on it.

2.3 Domain-specific Vocabulary

The choice of vocabulary directly shapes the embedding space of an LLM and drives the inference costs. Therefore, a lot of work has been done to develop a domain-specific vocabulary. Bolton et al. [4] introduce a custom BPE tokeniser tailored to a biomedical corpus to preserve healthcare-specific terms as single tokens. Kim et al. [14] similarly design a customised tokeniser for mental health records. Renc et al. [23] construct an EHR-event vocabulary that takes medical events as words to facilitate the encoding of clinical events. Although these domain-specific vocabularies enhance token efficiency, they require training on a specialised corpus from scratch, thereby forfeiting the advantages of pre-trained general-purpose LLMs. In contrast, our method augments existing BPE vocabularies, preserving all pre-trained embeddings and model parameters, while markedly reducing the sequence length in clinical prediction.

Recently, vocabulary adaptation methods emerged to enable domain-specific vocabulary integration to pre-trained LLMs. Han et al. [10] propose appending new tokens to the original vocabulary and adapting embeddings via an additional adapter module. However, this approach increases the vocabulary size and the number of model parameters. Nakash et al. [19] instead replace less useful original tokens with domain-specific ones, yet require dynamic token merging during inference, which increases the computational

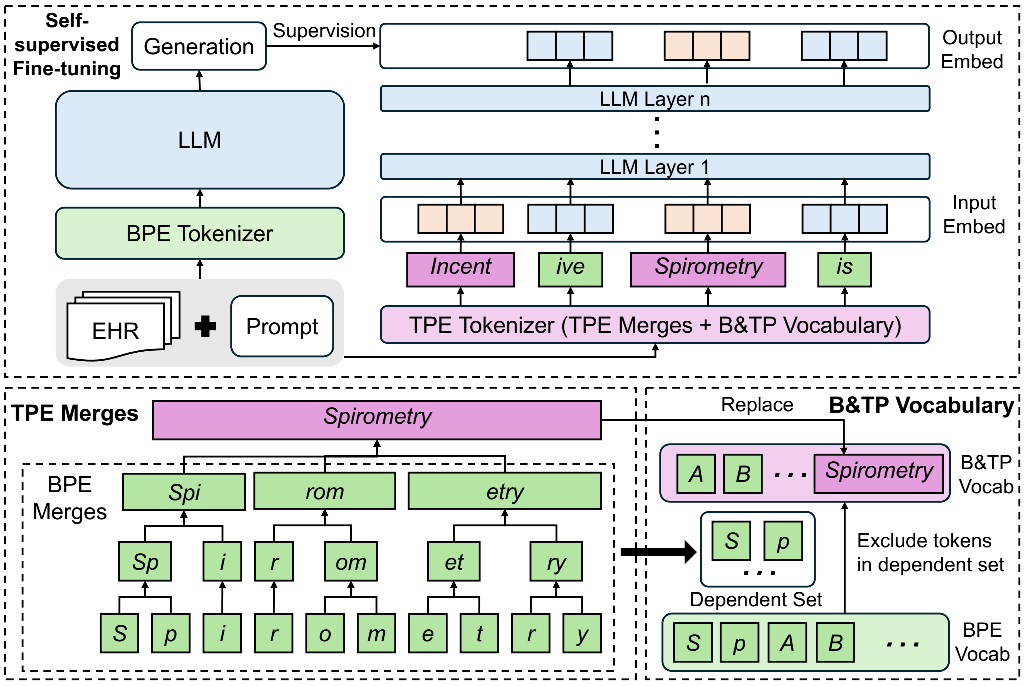


Figure 2: Illustration of the MedTPE tokenisation and encoding process. Raw EHR data and clinical prompts are first tokenised into subwords using the original BPE tokeniser, as well as passing through the TPE Tokeniser. The latter consists of the TPE Merges mechanism and the Byte and Token-Pair (B&TP) vocabulary. The TPE tokeniser then applies a domain-specific merge table to combine frequent adjacent BPE tokens into composite medical tokens (e.g., *Spirometry*). The resulting TPE-encoded sequence is passed to the LLM for efficient clinical prediction. Embeddings for new TPE tokens are initialised based on their BPE constituents and further fine-tuned via self-supervised fine-tuning, aligning the compressed vocabulary with the original LLM outputs without requiring human labels.

complexity of tokenisation. In contrast, our method neither increases vocabulary size nor model parameters, while maintaining tokenisation efficiency comparable to the original BPE approach.

3 Preliminaries

We represent the EHR of a patient $i \in \{1, \dots, N\}$ as a sequence of timestamped clinical events:

$$\mathbf{E}^{(i)} = \{e_1^{(i)}, e_2^{(i)}, \dots, e_{T^{(i)}}^{(i)}\}, \tag{1}$$

where each event $e_j^{(i)} = (c_j^{(i)}, o_j^{(i)}, t_j^{(i)})$ consists of a clinical concept $c_j^{(i)}$ (e.g., diagnosis, lab test, medication), an observation value $o_j^{(i)}$ (e.g., blood glucose level or test result), and a timestamp $t_j^{(i)}$. $T^{(i)}$ denotes the number of clinical events for patient i. Events are chronologically ordered, though multiple events may share the same timestamp.

Each structured event is converted into a natural language fragment using a data-to-text generation module:

$$s_j^{(i)} = \phi(e_j^{(i)}), \tag{2}$$

where $\phi : \mathcal{C} \times \mathcal{O} \times \mathbb{R} \rightarrow \mathcal{S}$ maps structured tuples into human-readable text. This yields an ordered sequence $\{s_1^{(i)}, s_2^{(i)}, \dots, s_{T^{(i)}}^{(i)}\}$, representing the patient's trajectory in natural language.

The text fragments are first concatenated to form a single text $s^{(i)} \in \mathcal{S}$ for each patient:

$$s^{(i)} = s_1^{(i)} \oplus s_2^{(i)} \oplus \dots \oplus s_{T^{(i)}}^{(i)}, \tag{3}$$

where $\oplus$ denotes concatenation.

This sequence is then tokenised using a subword tokeniser $\tau : \mathcal{S} \rightarrow \mathcal{V}$, yielding a token sequence:

$$\mathcal{X}^{(i)} = \{x_1^{(i)}, x_2^{(i)}, \dots, x_{L^{(i)}}^{(i)}\}, \quad x_n^{(i)} \in \mathcal{V}, \tag{4}$$

where $\mathcal{V}$ is the subword vocabulary and $L^{(i)}$ is the length of the token sequence for patient i.

To perform clinical prediction, a natural language prompt s_{prt} (e.g., "What is the most likely discharge diagnosis?") is appended and tokenised to M tokens:

$$\mathcal{P} = \{p_1, p_2, \dots, p_M\}, \quad p_m \in \mathcal{V}. \tag{5}$$

The complete tokenised model input becomes:

$$\mathcal{X}'^{(i)} = \{x_1^{(i)}, \dots, x_{L^{(i)}}^{(i)}, p_1, \dots, p_M\}. \tag{6}$$

A pretrained autoregressive language model f with parameters θ defines a conditional distribution over output sequences:

$$\mathcal{G}^{(i)} = f(\mathcal{X}'^{(i)}; \theta) = \prod_{k=1}^{K^{(i)}} p(g_k^{(i)} \mid \mathcal{X}'^{(i)}, g_{<k}^{(i)}; \theta), \tag{7}$$

where $\mathcal{G}^{(i)} = \{g_1^{(i)}, g_2^{(i)}, \ldots, g_{K^{(i)}}^{(i)}\}$, $\mathcal{G}^{(i)} \subset \mathcal{V}$ denotes the generated output tokens, and k indexes positions in the output sequence.

Finally, a task-specific extraction function $ext : \mathcal{V} \rightarrow \mathcal{Y}$ maps the model output to a structured prediction:

$$\hat{y}^{(i)} = ext(G^{(i)}), \tag{8}$$

where $\hat{y}^{(i)}$ denotes the clinical prediction (e.g., diagnostic codes, binary outcomes, or treatment recommendations).

For clarity, we use x_j to denote tokens in the subsequent sections.

4 Methodology

This section details the MedTPE framework designed to efficiently compress medical token sequences for pre-trained language models (LLMs). As illustrated in Fig. 2, we first identify frequent medical-specific tokens from tokenised medical corpora and construct a TPE vocabulary and tokeniser (Section 4.1). To incorporate these newly defined tokens without increasing the size of the vocabulary and embedding matrix, we employ a dependency-aware vocabulary replacement mechanism (Section 4.2). The embeddings for the new TPE tokens are then initialised to integrate into the pre-drilled embedding space (Section 4.3). Finally, we use self-supervised fine-tuning (SSFT), optimising only the new embeddings to preserve the general knowledge captured by the pre-trained LLM while benefiting from the domain-specific tokenisation (Section 4.4).

4.1 Token–Pair Encoding

TPE is built upon a pre-trained BPE tokeniser, treating complete BPE tokens as its fundamental units. It identifies and merges frequently co-occurring sequences of $n \geq 2$ tokens into domain-specific n-gram symbols. For any contiguous span of length $n_j \in \{2, \ldots, n_{\max}\}$, a composite TPE token is constructed as:

$$d_j = x_1 \oplus x_2 \oplus \cdots \oplus x_{n_j}, \quad x_i \in \mathcal{V}, \quad d_j \in \mathcal{V}_{\text{TPE}} \tag{9}$$

where $\mathcal{V}_{\text{TPE}}$ denotes the TPE vocabulary.

For each composite token d_j, a corresponding merge rule is defined and added to a *token-pair merge table*, which is ordered by the frequency of each token in the medical corpus. To define the merge process for d_j, we traverse its constituent sub-tokens from left to right to construct the *merge path* $MP(d_j)$:

$$MP(d_j) = \Big[(x_1, x_2),\ (x_1x_2, x_3), \\ \ldots,\ (x_1 \ldots x_{n_{j-1}}, x_{n_j})\Big]. \tag{10}$$

The merge path for each candidate token is appended to a unified token-pair merge table, with duplicate entries omitted. The final TPE merge table is constructed by concatenating the merge paths of all candidate tokens:

$$\mathcal{M}_{\text{TPE}} = \big[\, MP(d_1) \| MP(d_2) \| \ldots \| MP(d_J)\, \big], \tag{11}$$

where $\|$ denotes list concatenation with duplicate removal. The resulting merge table, together with the defined composite tokens, constitutes the TPE vocabulary $\mathcal{V}_{\text{TPE}}$.

The TPE tokenisation $\tau^\star$ operates in two steps: the input text is first tokenised using the original BPE tokenizer to produce an intermediate BPE token sequence, and subsequently, the TPE tokenizer applies the constructed merge table $\mathcal{M}_{\text{TPE}}$ to further merge frequent adjacent BPE tokens into more efficient, domain-specific TPE tokens. This layered tokenisation provides substantial compression of medical token sequences while maintaining the same computational complexity as standard BPE tokenisation, i.e., $O(N)$, where N is the length of the input token sequence.

4.2 Dependency-Aware Token Replacement

To integrate the new TPE vocabulary with the BPE vocabulary of the pre-trained LLM, it is necessary to combine these into a unified set. A straightforward solution, taking their union $\mathcal{V} \cup \mathcal{V}_{\text{TPE}}$, would increase the size of the embedding matrix and diminish efficiency. Instead, we propose to maintain the original vocabulary size by replacing the least useful BPE tokens with the most beneficial TPE tokens. Specifically, we employ a length-aware frequency score, which rewards tokens that both occur frequently and significantly compress sequences:

$$\text{score}(d_j) = \text{freq}(d_j) \cdot |d_j|_{\text{BPE}}, \tag{12}$$

where $\text{freq}(d_j)$ is the raw corpus frequency of token d_j, and $|d_j|_{\text{BPE}}$ is the number of composited BPE tokens.

We select the top-M TPE tokens ranked by this score to form the insertion set $\mathcal{I}$. However, directly replacing BPE tokens with TPE tokens can disrupt the underlying BPE merge paths needed for consistent tokenisation. Each TPE token $c \in \mathcal{I}$ depends on a BPE merge path $MP(d_j)$ resolving recursively to byte-level tokens. Thus, to ensure deterministic tokenisation, we must retain all BPE tokens involved in these paths, including any recursively required tokens. We explicitly define this *dependent set* $\mathcal{D}$ as:

$$\mathcal{D} = \left\{ x \;\middle|\; \begin{array}{l} \exists\, d_j \in \mathcal{I} \text{ such that } x \in MP(d_j), \\ \text{or} \\ \exists\, x' \in MP(d_j) \text{ such that } x \in MP(x') \end{array} \right\}. \tag{13}$$

Given the dependent set $\mathcal{D}$, we identify tokens eligible for replacement by forming the unprotected set $\mathcal{U} = \mathcal{V} \setminus \mathcal{D}$. We then select the M least frequent tokens from $\mathcal{U}$ according to corpus frequency, forming the eviction set $\mathcal{E} \subseteq \mathcal{U}$. Combining removals and insertions gives the B&TP vocabulary:

$$\mathcal{V}^\star = (\mathcal{V} \setminus \mathcal{E}) \cup \mathcal{I}, \quad \text{with} \quad |\mathcal{I}| = |\mathcal{E}| = M. \tag{14}$$

By explicitly protecting all dependent tokens in $\mathcal{P}$, the merge paths from the original BPE merge table $\mathcal{M}_{\text{BPE}}$ and the new TPE merge table $\mathcal{M}_{\text{TPE}}$ remain valid. This procedure ensures deterministic tokenisation, introduces only minor modifications to the pre-trained LLM's embedding matrix.

4.3 TPE Embedding Initialisation

Each new TPE token requires an embedding vector that is compatible with the pre-trained embedding space of the original LLM. For a given TPE token d_j composed of n_j constituent BPE tokens, we initialise its embedding vector $\mathbf{e}_{d_j}$ by computing the arithmetic mean of the pre-trained embeddings of its constituent tokens:

$$\tilde{\mathbf{e}}_{d_j} = \frac{1}{n_j} \sum_{i=1}^{n_j} \mathbf{e}_{x_i}. \tag{15}$$

To maintain numerical stability and prevent excessively strong initial embeddings, we further normalise this vector to half of the average embedding norm of the pre-trained BPE vocabulary. Specifically, the initial embedding is given by:

$$\mathbf{e}_{d_j} = 0.5\,\mu\,\frac{\tilde{\mathbf{e}}_{d_j}}{\|\tilde{\mathbf{e}}_{d_j}\|}, \quad \mu = \frac{1}{|\mathcal{V}|}\sum_{x\in\mathcal{V}}\|\mathbf{e}_x\|, \tag{16}$$

where μ denotes the average embedding norm across the original BPE vocabulary $\mathcal{V}$.

This embedding initialisation method places new TPE tokens within the pre-trained embedding distribution, enabling quick adaptation during the subsequent fine-tuning. It introduces only M additional embedding rows, preserving the structure of the pre-trained model and the efficiency of the parameters.

4.4 Self-Supervised Fine-tuning

After introducing new TPE tokens, we propose SSFT to optimise their embeddings for the medical domain, without requiring any human-annotated labels. Specifically, we leverage the pretrained LLM to generate supervision as follows: Given a medical text $s^{(i)}$ and LLM f with BPE tokeniser τ, we train the LLM's parameters $\theta^\star$ with TPE tokeniser $\tau^\star$ to minimise the cross-entropy loss:

$$\begin{aligned} \mathcal{Y}_{\text{ssft}} &= f\left(\tau(s^{(i)});\theta\right), \\ \mathcal{L}_{\text{ssft}} &= -\sum_{t=1}^{T}\mathcal{Y}_{\text{ssft}}\log f\left(\tau^\star(s^{(i)});\theta^\star\right). \end{aligned} \tag{17}$$

To maintain the efficiency and integrity of the pre-trained LLM, we freeze all original model parameters and update only the embeddings corresponding to the newly added TPE tokens. We implement a dedicated embedding split module (see Appendix A) to ensure that gradients are computed exclusively for the new embeddings, avoiding unnecessary memory usage and computation. By aligning the new TPE embeddings with the outputs of the pretrained model, this self-supervised approach preserves the LLM's original capability while enabling efficient adaptation to the domain-specific tokenisation, without any labelled clinical data.

5 Experiments and Discussion

5.1 Experiment Settings

5.1.1 Dataset & Processing. To evaluate the effectiveness of MedTPE, we conducted experiments on MIMIC-IV [13], a large-scale EHR dataset collected from Beth Israel Deaconess Medical Centre. MIMIC-IV includes detailed records for patients admitted to intensive care units, including diagnoses, procedures, medications, laboratory results, and demographic information. This diverse and comprehensive dataset enables the assessment of clinical prediction tasks. For feature modelling, we transform the raw EHR data into structured event streams following the Medical Event Data Standard (MEDS) [3]. The patient's record is represented as a chronologically ordered sequence of events, each consisting of a timestamp and a clinical measurement. The dataset is divided at the patient level into training, validation, and test sets in an 8:1:1 ratio. The implementation is available via an anonymous link[1].

[1]https://anonymous.4open.science/r/MedTPE-DA2E/README.md

5.1.2 Tasks & Metrics. We evaluated model performance on two standard clinical prediction tasks in line with established benchmarks [11]: (1) **ICU Mortality**, a binary classification task that predicts whether a patient will die during their ICU stay based on the first 24 hours of data after admission to the ICU; and (2) **Phenotyping**, a multi-label classification task that predicts the presence of 25 clinical phenotypes for each patient using the 24 hours of data after admission to the ICU.

We evaluate LLMs on three key capabilities:

- **Prediction Capability**: We assess predictive performance of the model using the F1 score, following standard practice in LLM-based healthcare benchmarks [5, 17]. For ICU mortality, we report the binary F1 score; for phenotyping, we use the macro-F1 score.
- **Format-Compliance Capability**: This metric evaluates how often the model produces outputs in the required format. We report the Format Compliance Rate (FCR) [20], defined as the proportion of outputs that strictly match the expected format. Outputs that do not meet this criterion are considered uninterpretable and are treated as incorrect predictions.
- **Inference Efficiency**: To quantify efficiency, we report both the input token count for EHR and medical text, and the model's average inference time.

5.2 Evaluation of LLMs with MedTPE

This experiment assesses whether MedTPE can compress input sequences without compromising predictive performance in clinical prediction tasks and its contribution to improving inference efficiency. We set the maximum sequence length to 8,192 tokens (6,144 for input and 2,048 for output). Both standard prompting and Chain of Thought (CoT) prompting [28] are compared using the baseline BPE tokeniser and MedTPE. To evaluate the effect of self-supervised fine-tuning (SSFT), we also report results for MedTPE without SSFT. Table 1 presents the F1 scores, format compliance rate (FCR), and inference time for all evaluated settings.

The results show that MedTPE maintains or even improves predictive performance compared to the original LLM with BPE tokenisation. For standard prompting, MedTPE preserves F1 and macro-F1 scores while reducing inference time by 69.3% for phenotyping. In the CoT setting, MedTPE significantly improved F1 scores and FCR for both clinical prediction tasks. Inference latency is also sharply reduced, with a 95.9% decrease for the prediction of ICU mortality.

These findings demonstrate that MedTPE substantially improves LLM inference efficiency without compromising predictive performance. Furthermore, SSFT enables the model to understand the new tokens. Without SSFT, both predictive performance and format compliance collapse, as the model cannot interpret the new TPE tokens that frequently appear in clinical inputs. This also results in increased inference time, as the LLM, unable to understand the input, generates more tokens in an attempt to resolve the prompt, which leads to significantly longer inference durations [18]. Overall, these results highlight the necessity of adapting the embedding when introducing domain-specific tokens.

Interestingly, we observe that small language models like Qwen2.5-1.5B[32] struggle with CoT prompting when input sequences are long (6K), resulting in poor format compliance and low F1 scores. This is likely due to their limited attention span, causing them to

Table 1: Assessment of Prediction, Format Compliance, and Inference Efficiency of LLMs with MedTPE. Mean and standard deviation of F1 scores are reported, calculated by bootstrapping the test set 1,000 times. Time is reported in minutes, and percentage change is calculated relative to the inference time of the original LLM without MedTPE.

LLM (Prompting)	ICU Mortality			Phenotyping		
	F1 (std)↑	FCR↑	Time (Changed %)↓	macro-F1 (std)↑	FCR↑	Time (Changed %)↓
Qwen2.5-1.5B (-)	**0.123 (0.006)**	0.997	43.483 (-)	**0.178 (0.004)**	0.972	41.491 (-)
Qwen2.5-1.5B (-) + MedTPE	0.122 (0.006)	**1.0**	**18.306 (-57.9%)**	0.174 (0.004)	**0.997**	**12.717 (-69.3%)**
Qwen2.5-1.5B (-) + MedTPE w.o SSFT	0.001 (0.0)	0.041	59.369 (+36.5%)	0.004 (0.001)	0.055	85.283 (+105.5%)
Qwen2.5-1.5B (CoT)	0.061 (0.005)	0.603	484.411 (-)	0.039 (0.002)	0.581	303.230 (-)
Qwen2.5-1.5B (CoT) + MedTPE	**0.122 (0.006)**	**0.999**	**19.729 (-95.9%)**	**0.127 (0.004)**	**0.775**	**62.94 (-79.2%)**
Qwen2.5-1.5B (CoT) + MedTPE w.o SSFT	0.001 (0.001)	0.036	129.389 (-73.3%)	0.001 (0.0)	0.018	161.145(-46.9%)

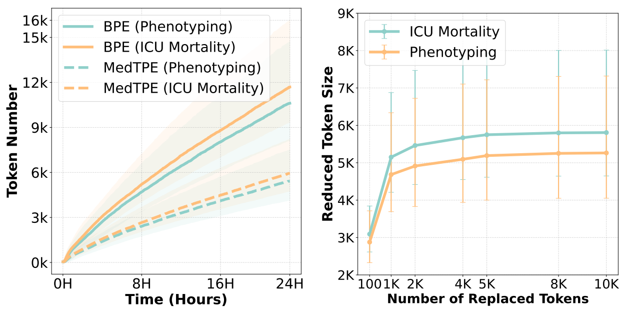
lose track of task instructions and prediction targets[2]. By compressing input sequences, MedTPE alleviates this limitation, helping LLMs retain essential information and consistently produce outputs in the required format. As a result, even small models benefit from CoT prompting when equipped with MedTPE, showing clear improvements in both accuracy and instruction-following capability.

5.3 Token Compression Analysis

This experiment assesses the effectiveness of MedTPE in compressing model input sequences for clinical prediction. We analysed patient cohorts for ICU mortality and MIMIC-IV phenotyping tasks, focusing on the first 24 hours of each patient's stay in the ICU. For each case, the input sequences were tokenised using both the Qwen2.5 BPE tokeniser[32] and our proposed MedTPE tokeniser. We then compared the resulting token counts across the test set, reporting the median and interquartile range to illustrate the distribution and magnitude of compression achieved. Figure 3(a) illustrates that, for the ICU mortality cohort, MedTPE reduces the median token count by 49.1% compared to the baseline BPE tokeniser. Similarly, MedTPE achieves a reduction of 49.0% in the phenotyping cohort. These substantial reductions in input sequence length are achieved without increasing the size of the vocabulary or expanding the embedding layer of the LLM. MedTPE accomplishes this by employing a dependency-aware token replacement strategy, which ensures that high-utility TPE tokens replace only the least useful BPE tokens, while strictly preserving all necessary dependencies required for deterministic tokenisation. As a result, MedTPE delivers efficient compression while maintaining the fixed vocabulary budget and parameter count of the pre-trained language model.

5.4 Analysis of Token Replacement Budget

This experiment investigates how the replacement budget, defined as the number of BPE tokens substituted with TPE tokens, affects the compression of the sequence in MedTPE. We vary the replacement budget M and measure the resulting token counts by encoding the first 24 hours of ICU stay in the MIMIC-IV test set. The resulting token sequence lengths are evaluated across both the ICU mortality and phenotyping cohorts, with the median and interquartile ranges reported. The results, as illustrated in Figure 3(b), reveal a clear trend: As the replacement budget increases from 100 to 2,000 tokens, the median token count in the test set decreases rapidly, indicating substantial compression gains. Beyond 2,000 replacements, the improvements become more gradual, and after 5,000 tokens have



(a) Compression Capability **(b) Token Replacement Budget**

Figure 3: Assessment of MedTPE's token sequence compression. The first 24 hours of ICU stay from the MIMIC-IV test set are used, with the tokeniser of Qwen2.5 [32]. (a) MedTPE achieves approximately 49% compression in token count for both the phenotyping and ICU mortality cohorts. The solid line shows the median, while the shaded region represents the interquartile range (25th to 75th percentiles) across the test set. (b) Analysis of the token replacement budget: the median value of reduced token size increases sharply as the number of replaced tokens increases from 100 to 2,000, with only marginal improvement beyond 5,000 replacements. Points indicate medians, and error bars show the interquartile range across the test set.

been replaced, further compression gains are minimal. These observations are consistent across both cohorts. These findings suggest that a replacement budget of 5000, which is 3.3% compared to the original Qwen2.5 vocabulary, offers an effective trade-off between sequence compression and vocabulary management in MedTPE. This setting achieves near-optimal token reduction while avoiding unnecessary increases in the risk of discarding potentially useful subwords from the original BPE vocabulary. As such, 5,000 replaced tokens are adopted as the default budget, balancing compression capability and efficiency in further fine-tuning.

6 Conclusion

In this work, we introduced MedTPE, an efficient two-level tokenisation method for compressing EHR token sequences for clinical prediction with pre-trained LLMs. MedTPE augments the BPE vocabulary with carefully selected domain-specific token pairs, achieving up to 49% sequence compression without compromising predictive performance. Our dependency-aware replacement algorithm ensures deterministic tokenisation and compatibility with existing models. Furthermore, we show that SSFT is essential for enabling the model to utilise the new tokens. Together, MedTPE addresses the challenges of long EHR trajectories, making LLMs more practical and efficient for healthcare applications.

Despite these benefits, there are still areas for further improvement. Although this study focuses on sequence compression and predictive performance, a more comprehensive evaluation of different embedding initialisation strategies could further enhance the adaptability of MedTPE, especially for rapid fine-tuning. Additionally, MedTPE is currently implemented as an offline pre-processing step and can not dynamically adapt to new or evolving clinical terms that may arise post-deployment. In future work, we plan to extend MedTPE to support test-time training [26], allowing dynamic vocabulary updates in response to novel medical terms and enabling personalised vocabulary learning for individual patients.

References

[1] Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. 2019. Publicly Available Clinical. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*. Association for Computational Linguistics.

[2] Chenxin An, Jun Zhang, Ming Zhong, Lei Li, Shansan Gong, Yao Luo, Jingjing Xu, and Lingpeng Kong. 2024. Why Does the Effective Context Length of LLMs Fall Short? *arXiv preprint arXiv:2410.18745* (2024).

[3] Bert Arnrich, Edward Choi, Jason Alan Fries, Matthew BA McDermott, Jungwoo Oh, Tom Pollard, Nigam Shah, Ethan Steinberg, Michael Wornow, and Robin van de Water. 2024. Medical event data standard (MEDS): Facilitating machine learning for health. In *ICLR 2024 Workshop on Learning from Time Series For Health*. 03–08.

[4] Elliot Bolton, Abhinav Venigalla, Michihiro Yasunaga, David Hall, Betty Xiong, Tony Lee, Roxana Daneshjou, Jonathan Frankle, Percy Liang, Michael Carbin, et al. 2024. Biomedlm: A 2.7 b parameter language model trained on biomedical text. *arXiv preprint arXiv:2403.18421* (2024).

[5] Yan Cai, Linlin Wang, Ye Wang, Gerard de Melo, Ya Zhang, Yanfeng Wang, and Liang He. 2024. Medbench: A large-scale chinese benchmark for evaluating medical large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 17709–17717.

[6] Canyu Chen, Jian Yu, Shan Chen, Che Liu, Zhongwei Wan, Danielle Bitterman, Fei Wang, and Kai Shu. 2024. ClinicalBench: Can LLMs Beat Traditional ML Models in Clinical Prediction? *arXiv preprint arXiv:2411.06469* (2024).

[7] Hejie Cui, Zhuocheng Shen, Jieyu Zhang, Hui Shao, Lianhui Qin, Joyce C Ho, and Carl Yang. 2025. Llms-based few-shot disease predictions using ehr: A novel approach combining predictive agent reasoning and critical agent instruction. In *AMIA Annual Symposium Proceedings*, Vol. 2024. 319.

[8] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2022. 8-bit Optimizers via Block-wise Quantization. In *International Conference on Learning Representations*. https://openreview.net/forum?id=shpkpVXzo3h

[9] Scott L Fleming, Alejandro Lozano, William J Haberkorn, Jenelle A Jindal, Eduardo Reis, Rahul Thapa, Louis Blankemeier, Julian Z Genkins, Ethan Steinberg, Ashwin Nayak, et al. 2024. Medalign: A clinician-generated dataset for instruction following with electronic medical records. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 22021–22030.

[10] HyoJung Han, Akiko Eriguchi, Haoran Xu, Hieu Hoang, Marine Carpuat, and Huda Khayrallah. 2025. Adapters for Altering LLM Vocabularies: What Languages Benefit the Most?. In *The Thirteenth International Conference on Learning Representations*.

[11] Hrayr Harutyunyan, Hrant Khachatrian, David C Kale, Greg Ver Steeg, and Aram Galstyan. 2019. Multitask learning and benchmarking with clinical time series data. *Scientific data* 6, 1 (2019), 96.

[12] Abul Hasan, Jinge Wu, Quang Ngoc Nguyen, Salomé Andres, Imane Guellil, Huayu Zhang, Arlene Casey, Beatrice Alex, Bruce Guthrie, and Honghan Wu. 2024. Infusing clinical knowledge into tokenisers for language models. *arXiv preprint arXiv:2406.14312* (2024).

[13] Alistair EW Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, et al. 2023. MIMIC-IV, a freely accessible electronic health record dataset. *Scientific data* 10, 1 (2023), 1.

[14] Dae-young Kim, Rebecca Hwa, and Muhammad Mahbubur Rahman. 2024. mhGPT: A lightweight generative pre-trained transformer for mental health text analysis. *arXiv preprint arXiv:2408.08261* (2024).

[15] Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 66–71.

[16] Terrence C Lee, Neil U Shah, Alyssa Haack, and Sally L Baxter. 2020. Clinical implementation of predictive models embedded within electronic health record systems: a systematic review. In *Informatics*, Vol. 7. MDPI, 25.

[17] Jin Li, Yiyan Deng, Qi Sun, Junjie Zhu, Yu Tian, Jingsong Li, and Tingting Zhu. 2024. Benchmarking large language models in evidence-based medicine. *IEEE Journal of Biomedical and Health Informatics* (2024).

[18] Yufei Li, Zexin Li, Wei Yang, and Cong Liu. 2023. RT-LM: Uncertainty-Aware Resource Management for Real-Time Inference of Language Models. In *2023 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 158–171.

[19] Itay Nakash, Nitay Calderon, Eyal Ben David, Elad Hoffer, and Roi Reichart. 2025. AdaptiVocab: Enhancing LLM Efficiency in Focused Domains through Lightweight Vocabulary Adaptation. *arXiv preprint arXiv:2503.19693* (2025).

[20] Minheng Ni, Zhengyuan Yang, Linjie Li, Chung-Ching Lin, Kevin Lin, Wangmeng Zuo, and Lijuan Wang. 2025. Point-RFT: Improving Multimodal Reasoning with Visually Grounded Reinforcement Finetuning. *arXiv preprint arXiv:2505.19702* (2025).

[21] Shuai Niu, Jing Ma, Liang Bai, Zhihua Wang, Li Guo, and Xian Yang. 2024. EHR-KnowGen: Knowledge-enhanced multimodal learning for disease diagnosis generation. *Information Fusion* 102 (2024), 102069.

[22] Alvin Rajkomar, Eyal Oren, Kai Chen, Andrew M Dai, Nissan Hajaj, Michaela Hardt, Peter J Liu, Xiaobing Liu, Jake Marcus, Mimi Sun, et al. 2018. Scalable and accurate deep learning with electronic health records. *NPJ digital medicine* 1, 1 (2018), 18.

[23] Pawel Renc, Yugang Jia, Anthony E Samir, Jaroslaw Was, Quanzheng Li, David W Bates, and Arkadiusz Sitek. 2024. Zero shot health trajectory prediction using transformer. *NPJ Digital Medicine* 7, 1 (2024), 256.

[24] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1715–1725.

[25] Xinying Song, Alex Salcianu, Yang Song, Dave Dopson, and Denny Zhou. 2021. Fast WordPiece Tokenization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2089–2103.

[26] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. 2020. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*. PMLR, 9229–9248.

[27] Ryan Synk, Monte Hoover, John Kirchenbauer, Neel Jain, Alex Stein, Manli Shu, Josue Melendez Sanchez, Ramani Duraiswami, and Tom Goldstein. 2025. Exploiting Sparsity for Long Context Inference: Million Token Contexts on Commodity GPUs. *arXiv preprint arXiv:2502.06766* (2025).

[28] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* 35 (2022), 24824–24837.

[29] Christopher YK Williams, Brenda Y Miao, Aaron E Kornblith, and Atul J Butte. 2024. Evaluating the use of large language models to provide clinical recommendations in the Emergency Department. *Nature Communications* 15, 1 (2024), 8236.

[30] Michael Wornow, Suhana Bedi, Miguel Angel Fuentes Hernandez, Ethan Steinberg, Jason Alan Fries, Christopher Re, Sanmi Koyejo, and Nigam Shah. 2025. Context Clues: Evaluating Long Context Models for Clinical Prediction Tasks on EHR Data. In *The Thirteenth International Conference on Learning Representations*.

[31] Zhenbang Wu, Anant Dadu, Michael Nalls, Faraz Faghri, and Jimeng Sun. 2024. Instruction Tuning Large Language Models to Understand Electronic Health Records. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[32] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024. Qwen2 Technical Report. *CoRR* abs/2407.10671 (2024). https://doi.org/10.48550/arXiv.2407.10671

[33] Min-Yung Yu. 2025. MedSeg: A Statistical Approach to Tokenization Assessment in Medical NLP. *Journal of Information Systems Engineering and Management* 10 (04 2025), 698–704. https://doi.org/10.52783/jisem.v10i37s.6506

A Embedding Split Module

To enable supervised fine-tuning of only a subset of embedding vectors, we introduce the embedding split module. Fine-tuning a subset of embeddings within a single unified embedding matrix can result in unnecessary gradient computation for the frozen embeddings, leading to increased memory usage and computational overhead. Our module addresses this by explicitly partitioning the embedding set into two disjoint subsets:

$$\mathbf{E} = \mathbf{E}_{\text{BPE, fixed}} \cup \mathbf{E}_{\text{TPE, trainable}},$$

where $\mathbf{E}_{\text{BPE, fixed}}$ denotes the pre-trained BPE embeddings, which remain fixed, and $\mathbf{E}_{\text{TPE, trainable}}$ is the embeddings of the newly introduced TPE tokens, which are trainable.

During the forward pass, embeddings are retrieved by separately indexing both subsets and concatenating the results to form the complete embedding set $\mathbf{E}$. During the backward pass, gradient updates are restricted to the trainable subset $\mathcal{E}_{\text{TPE, trainable}}$, substantially improving memory and computational efficiency.

B Training Details

Table 2: Hyperparameter Settings

Hyperparameter	Value
Learning rate (LR)	5×10^{-5}
Batch size	2
Number of training epochs	1
Maximum sequence length	4,096
Steps to accumulate gradients	2
Proportion of total steps for LR warmup	0.1
Maximum gradient norm for clipping	1.0

The fine-tuning was run on an A6000-Ada 48GB GPU. We train using the 8-bit AdamW optimiser[8] to reduce memory consumption. The optimiser is configured with default hyperparameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The learning rate schedule consists of a linear warmup for the first 10% of total steps, followed by a cosine decay for the remaining steps. To prevent gradient explosion, we apply gradient clipping by norm at each update. During training, we evaluate the model's validation loss every 1,000 steps. An early stopping mechanism halts training if validation performance does not improve for three consecutive evaluations. The hyperparameters for fine-tuning are detailed in Table 2.

C Prompts

C.1 ICU Mortality Prompt

```
## Background
You are an ICU clinician with exceptional
    expertise in analyzing Electronic Health
    Records (EHR).  Your task is to predict ICU
    mortality for a patient based on their first
    24 hours of ICU stay. The data provided
    includes:
- **Patient context**: Demographic and baseline
    health information.
- **Clinical events**: A chronological sequence of
     events extracted from their EHR, such as
    diagnoses, medications, procedures, vitals,
    and ICU interventions.

## Event Descriptions
Patient's trajectory in hospital(s) is presented
    by clinical events. The event types are as
    follows:
1. HOSPITAL_ADMISSION: Admission to a hospital.
2. HOSPITAL_DISCHARGE: Discharge from a hospital.
3. DIAGNOSIS: A disease diagnosis for a patient.
4. MEDICATION: Administering or prescribing
    medication to a patient.
5. GENDER: Recording a patient's gender.
6. AGE: Recording a patient's age.
7. RACE: Recording a patient's race or ethnicity.
8. MEDS_BIRTH: Recording medications administered
    at or present during the patient's birth.
9. MEDS_DEATH: Recording medications administered
    around the time of the patient's death.
10. ICU_ADMISSION: Admission to an Intensive Care
    Unit (ICU).
11. ICU_DISCHARGE: Discharge from an ICU.
12. LAB: Recording laboratory test results.
13. VITALS: Recording vital signs of a patient.
14. INFUSION: Administering an infusion to a
    patient.
15. SUBJECT_WEIGHT_AT_INFUSION: Recording a
    patient's weight at the time of infusion.
16. ADMISSION_DIAGNOSIS: Primary diagnosis
    recorded at the time of hospital admission.
17. CAREPLAN_GENERAL: Documenting general care
    plans for patient management.
18. CAREPLAN_GOAL: Outlining specific goals in the
     patient's care plan.
19. CAREPLAN_EOL: Documenting end-of-life care
    plans and directives.
20. CAREPLAN_INFECTIOUS_DISEASE: Care plans
    specifically addressing infectious disease
    management.
21. ALLERGY: Recording a patient's known allergies
    .

## Instructions
1. Review and Analyse Clinical Events:
   Carefully examine the clinical events from the
       EHR data to understand the patient's
       general health profile. Pay attention to
       key events such as diagnoses, medications,
       procedures, lab tests, and vital signs.
   Analyze the clinical events to identify
       patterns or indicators that may predict ICU
        mortality.
2. Make a Prediction:
   Based on your analysis, predict the patient's
       outcome by selecting one of the following
       options:
   - A: Alive
   - B: Death
   Provide your prediction in the specified format
       by placing your answer (A or B) in json
       format with the key of "answer".
   {"answer": "Your Answer"}
   For example, if your prediction is Alive, your
       submission should be like this:
    {"answer": "A"}
```

C.2 Phenotyping Prompt

```
## Background
You are an ICU clinician with exceptional
    expertise in analyzing Electronic Health
    Records (EHR). Your task is to predict disease
     phenotypes for a patient upon ICU discharge
    based on their first 24 hours of ICU stay. The
     data provided includes:
- **Patient context**: Demographic and baseline
    health information.
- **Clinical events**: A chronological sequence of
     events extracted from their EHR, such as
    diagnoses, medications, procedures, vitals,
    and ICU interventions.

## Event Descriptions
Patient's trajectory in hospital(s) is presented
    by clinical events. The event types are as
    follows:
1. ED_REGISTRATION: Registration to an emergency
    department.
2. ED_OUT: Discharge or exit from the emergency
    department.
3. HOSPITAL_ADMISSION: Admission to a hospital.
4. HOSPITAL_DISCHARGE: Discharge from a hospital.
5. DIAGNOSIS: A disease diagnosis for a patient.
6. DRG: Assigning a Diagnosis-Related Group (DRG)
    code for billing and classification purposes.
7. MEDICATION: Administering or prescribing
    medication to a patient.
8. HCPCS: Recording Healthcare Common Procedure
    Coding System (HCPCS) codes for procedures and
     services.
9. LAB: Recording laboratory test results.
10. OMR: Recording Objective Medical Records (OMR)
    , such as structured clinical data.
11. GENDER: Recording a patient's gender.
16. PROCEDURE: A medical procedure.
17. TRANSFER_TO: Transferring a patient to another
     department or facility.
18. ICU_ADMISSION: Admission to an Intensive Care
    Unit (ICU).
19. ICU_DISCHARGE: Discharge from an ICU.
20. VITALS: Recording vital signs of a patient.
21. INFUSION_START: Starting an infusion.
22. INFUSION_END: Ending an infusion.
23. SUBJECT_WEIGHT_AT_INFUSION: Recording a
    patient's weight at the time of infusion.
24. SUBJECT_FLUID_OUTPUT: Recording a patient's
    fluid output.

## Phenotype Options
A: Acute and unspecified renal failure
B: Acute cerebrovascular disease
C: Acute myocardial infarction
D: Cardiac dysrhythmias
E: Chronic kidney disease
F: Chronic obstructive pulmonary disease and
    bronchiectasis
G: Complications of surgical procedures or medical
     care
H: Conduction disorders
I: Congestive heart failure; nonhypertensive
J: Coronary atherosclerosis and other heart
    disease
K: Diabetes mellitus with complications
L: Diabetes mellitus without complication
M: Disorders of lipid metabolism
N: Essential hypertension
O: Fluid and electrolyte disorders
P: Gastrointestinal hemorrhage
Q: Hypertension with complications and secondary
    hypertension
R: Other liver diseases
S: Other lower respiratory disease
T: Other upper respiratory disease
U: Pleurisy; pneumothorax; pulmonary collapse
V: Pneumonia
W: Respiratory failure; insufficiency; arrest
X: Septicemia (except in labor)
Y: Shock

## Instructions
1. Review and Analyse Clinical Events:
   Carefully examine the clinical events from the
       EHR data to understand the patient's health
        status.
   Analyze the events to identify indicators that
       suggest specific disease phenotypes from
       the available options.

2. Make Predictions:
   Based on your analysis, predict which
       phenotypes the patient will have upon
       discharge by selecting from the available
       options.
   A patient may have zero, one, or multiple
       phenotypes.

   Provide your prediction in JSON format with:
   - "answer": A list of the selected option
       letters (A, B, C, etc.)

   Example format:
   {"answer": ["A", "C"]}

   For instance, if you predict the patient will
       have Pneumonia (V) and Septicemia (X), your
        submission should look like:
   {"answer": ["V", "X"]}

   Important:
   - Only use the option letters (A, B, C, etc.)
       in the "answer" list
```