RECOMMEND LABORATORY ITEMS USING DISCRETE-TIME DYNAMIC GRAPHS

Anonymous due to double-blind
ANONYMOUS
x@ANONYMOUS

ABSTRACT

Laboratory items, whose results are the basis of clinical decisions, give essential information to clinicians. However, due to individual differences in professional knowledge and experience, cognitive bias happening among clinicians leads to unscheduled laboratory items that should not have been neglected, which even cause fatal consequences sometimes. Traditional approaches have attempted to prevent bias by educating clinicians and using cognitive psychology strategies, which are subjective and prone to be influenced by personal factors. Therefore, we create an objective model to provide supplementary advice for clinicians by recommending what laboratory items need to do next based on patients' historical results, which are logged in the Electronic Medical Records (EMR) form. We exploit the natural heterogeneity of EMR data to transform the recommendation problem into the edges prediction problem on *discrete-time dynamic graphs*. The evaluation results conducted on a public dataset prove the efficiency and practicality of our laboratory items recommendation model.

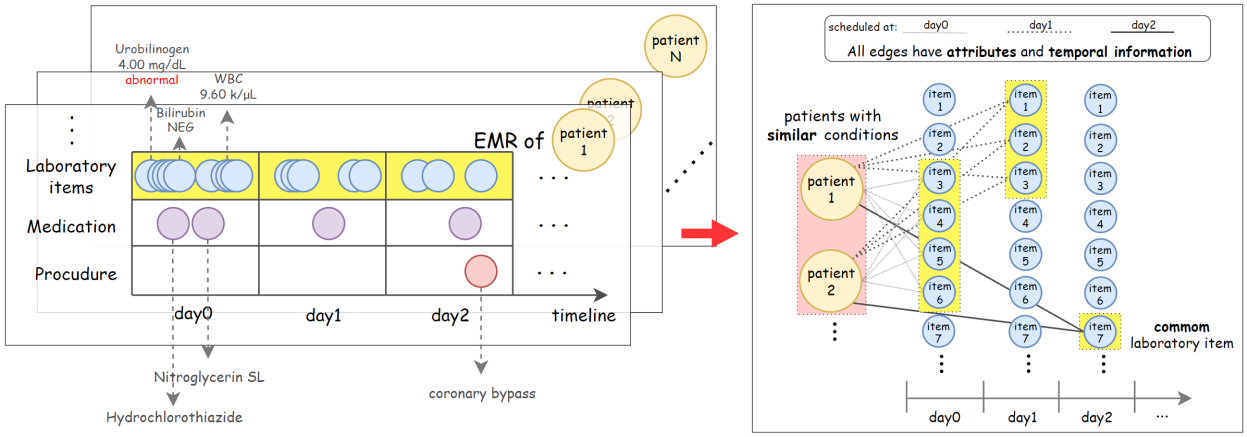
***K*eywords** Laboratory items · Electronic Medical Records · Heterogeneous Graph · Sequential Recommendation Systesm · Discrete-Time Dynamic Graph

Laboratory items are tests performed by medical laboratories on human blood, urine, body fluids, tissues, cells, and other biological samples, which provide references for evaluating human health status, diagnosing diseases, monitoring treatment effects and prognosis, etc. Moreover, clinical decisions are made tightly relying on the results of laboratory items which give essential information to clinicians. For example, the evaluation for Heart Failure (HF) is performed using the results of blood tests and HF-specific laboratory tests (BNP and NT-proBNP)[1]; diagnostic procedures of AML contain many laboratory tests of morphology, immunophenotyping, molecular genetic testing, and so on[2].

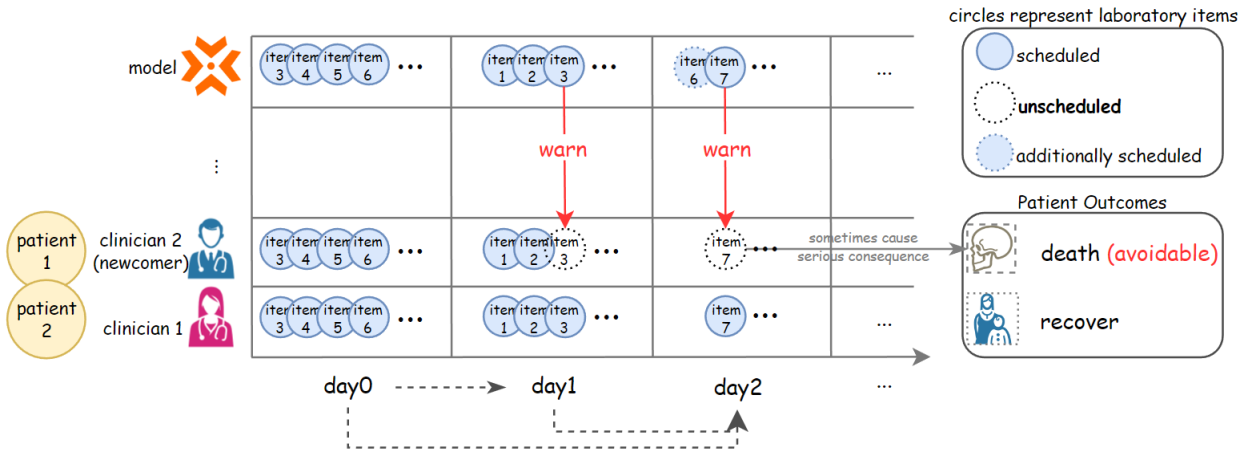
However, due to individual differences in professional knowledge and experience among clinicians, clinical decisions vary from one to another even when they both get the same information about laboratory items. In practice, individual differences are usually called cognitive bias, which sometimes would have fatal consequences[3]. For example, a case study about misdiagnosis[4] reported that a patient died of an undiscovered perforated colonic diverticulum in the pelvis caused by the clinicians' cognitive bias toward the key result of laboratory items[1]. Even if any clinician performed corresponding laboratory items on the patient's pelvis, she would not have died. Another study[5] also reported that the performance and interpretation of diagnostic tests contributed to about 35% of MDOs (missed diagnostic opportunities). As *Reason* has stated[6]: "Our propensity for certain types of error is the price we pay for the brain's remarkable capacity to think and act intuitively." Therefore, if we can create a model to provide supplementary advice for clinicians by recommending what laboratory items need to do next based on patients' historical results, we can not only reduce the probability of cognitive bias happens among clinicians but also help patients get the correct treatments which would even save many lives!

Then, the following critical question is, *how to implement the recommendations?* We start thinking about this question in terms of the characteristics of patients' history results. See the left part of Fig.1a, after a patient is admitted to the hospital, a series of Electronic Medical Records (EMR) will be generated, which contain the results of the laboratory items scheduled by the clinicians and other kinds of results (e.g. medication, procedure). Extending the range to all patients, we got an EMR dataset which is naturally heterogeneous as it records different relationships between different entities (patients, laboratory items, medications, procedures, and so on). Then, if we only consider the relationship between patients and laboratory

[1]In this case, the key result was "100 leucocytes per high power field but yielded no bacterial growth" found in the MSU (mid-stream specimen of urine) test.



(a) EMR dataset and its corresponding heterogeneous graph. Left: the EMR dataset that contains the historical results of patients. Right: the corresponding heterogeneous graph which only considers the relationship between patients and laboratory items; in this heterogeneous graph, we demonstrate that patients with similar generally scheduled common laboratory items, which is the so-called *semantics*.



(b) Illustration for explaining how our laboratory items recommendation model provides supplementary advice to clinicians. The newcomer or inexperienced clinician(s) can be warned by our model if there is an item that should not be unscheduled, which helps to avoid cognitive bias and its serious consequences.

Figure 1: Overview

items, we can construct the corresponding heterogeneous graph[2] of EMR dataset, which has rich semantics to be discovered and exploited[7]. For example, the right part of Fig.1a demonstrates that, by attaching the results of items as edges attributes and associating the released time of results, patients with similar conditions are generally scheduled common laboratory items, and their results are generally similar. From this perspective, the laboratory items scheduled for patients at someday can also be regarded as the edges connected between patients and items on that day; the work of clinicians to schedule laboratory items for patients based on their historical results can also be regarded as predicting which edges will connect next by utilizing the semantics existing in the previous days' edges.

Hence, we seek inspiration for our problem from other domains whose data can be represented with heterogeneous graphs. Coincidentally, we got inspiration from the SRSs (Sequential Recommendation Systems) domain which is widely used in e-commerce platforms.

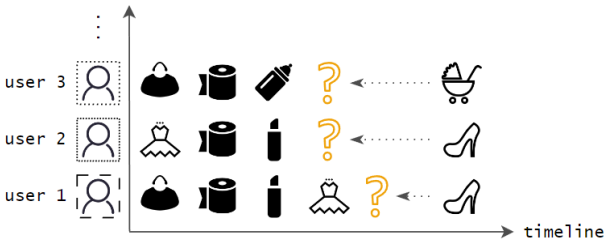
SRSs recommend items that may be of interest to a user by mainly modeling the sequential dependencies over the user-item interactions (e.g., viewing or purchasing items on an online shopping platform) in a sequence, besides, SRSs also treat the user-item interactions as a dynamic sequence and take the sequential dependencies into account to capture the current and recent preferences of a user for more accurate recommendations[8]. Different from the conventional recommender systems (RSs) including collaborative filtering and content-based filtering[9], SRSs

[2]For the convenience of display, in the right part of Fig.1a, we use different lines to represent the connection at different times and draw the items repeatedly.

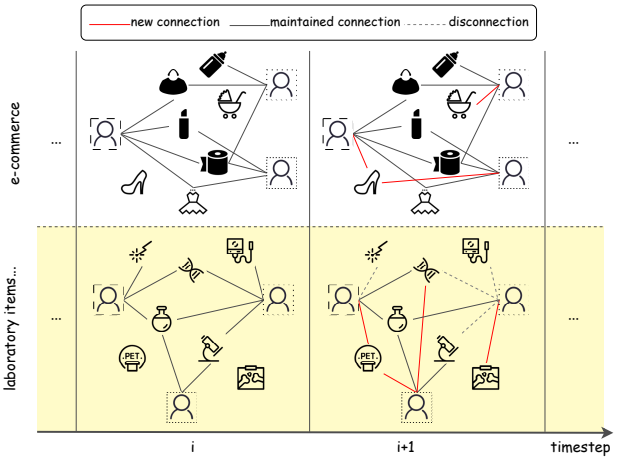
try to understand and model the sequential user behaviors, the interactions between users and items, and the evolution of users' preferences and item popularity over time[10]. Recent studies of SRSs mainly focus on personalized recommendations, where users' demands are not only conditioned by their profile but also by their recent browsing behaviors as well as periodical purchases made some time ago (e.g. Jin et al. proposed a novel framework that can capture the evolving demands of users over time through a unified search-based time-aware model[11]). For example, in Fig.2a, since the products purchased by user 1 and user 2 are related to women's clothing and cosmetics, SRSs tend to recommend high heels to them; and since user 3 has recently purchased baby products such as feeding bottles, SRSs are more inclined to recommend strollers to her than high-heeled shoes.

The user-item interactions data of SRSs is also naturally heterogeneous like the EMR data, as there existing relationships between different entities (users, items). However, many works in SRSs only model the users' and items' sequences individually[11, 12], while ignoring the graph structure that naturally exists among them and the relationships that exist in the graph structure along the timeline (these are exactly the *semantics* of the heterogeneous graph we mentioned above). This ignorance would have serious consequences in our clinical scenario, which is unacceptable for us. But, we noticed a novel work[13] which takes advantage of a technique called *discrete-time dynamic graph*[14] (annotated as $\mathcal{DTDG}$ next) which can model the graph structure among users-items sequences and capture the relationships along the timeline. As shown in Fig.2b, by using $\mathcal{DTDG}$s to represent both e-commerce and laboratory item recommendation scenarios, both their natural heterogeneous graph structure and relationships along the timeline are modeled. But there is a difference between the two scenarios, that is, in the e-commerce scenario, we only need to recommend the single next item; while in the laboratory item recommendation scenario, we need to recommend multiple items (comparing to the previous timestep, some are still needed, some are newly added and some are not needed). Hence, by representing our EMR data in this $\mathcal{DTDG}$ form, we can create a model that predicts which edges will connect next by utilizing the semantics existing in the previous days' edges to feasibly solve the question of implementing the laboratory items recommendation.

For understanding more clearly what role the recommendation model plays in providing supplementary advice to clinicians, we illustrate it with Fig.1b. Based on the information gained from the results of laboratory items so far, clinicians schedule the next day's laboratory items for patients with similar conditions. For example, day 1's scheduled laboratory items are based on day 0's information; day 2's are based on day $0 \sim 1$'s. The recommendation model which is trained from a large number of $\mathcal{DTDG}$s representing the EMR does the same thing for patients. However, due to the existence of cognitive bias, clinicians may fail to schedule some laboratory items that should



(a) An illustrative example of using SRSs in a typical e-commerce platform scenario. Users with similar preferences will be recommended similar items by SRSs, but SRSs will also consider the items recently purchased by the user(s) to make recommendations.



(b) Illustration of similarities and difference between e-commerce scenarios and laboratory items recommendation scenarios using discrete-time dynamic graphs.

Figure 2: Schematic diagram of the inspiration we got from the SRSs domain for implementing our laboratory items recommendation.

be scheduled for patients, resulting in the missing of key information about patients. The consequences of missing will be magnified over time, causing more unscheduled laboratory items and ultimately the death of patients (as in the case reported in [4]). Hence, the recommendation model can act as an assistant to help clinicians to check and find unscheduled important laboratory items in time, so as to supplement the clinical decisions and even avoid catastrophic consequences. Besides, it is worth mentioning that even if the model recommends redundant laboratory item(s), it only means additional information (sometimes even beneficial) to clinicians, and it is harmless to patients.

In this work, we convert the laboratory items recommendation problem to the edges prediction problem on $\mathcal{DTDG}$s equivalently, then create a model to recommend laboratory items using these $\mathcal{DTDG}$s. The model mainly consists of two parts: one is the encoders that use GNN for capturing the graph structure information in $\mathcal{DTDG}$s, and another is the decoders that use *masked self-attention mechanism* to

learn the relationships of history time-sequential information. Our major contributions are summarized as follows:

- We transform the laboratory items recommendation problem to the edges prediction problem on $\mathcal{DTDG}$s equivalently.
- We create a generic model that takes advantage of GNN's powerful spatial semantics acquisition capability and self-attention mechanism's powerful sequential information modeling capability.
- We conduct comprehensive evaluation experiments on a public medical dataset mimic-iii[15] to demonstrate the effectiveness and practicality of our proposed model.

Results

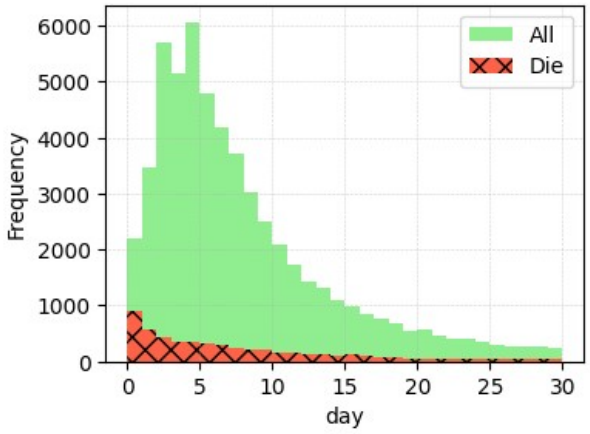
Statistical Analysis of Data

Since the raw data is finally converted into $\mathcal{DTDG}$s (stated from appendix A.2 to appendix A.3), and the laboratory items recommendation problem is equal to predict the edges connectivity between the patient nodes and the laboratory item nodes of $\mathcal{DTDG}$s (stated in appendix A.1), statistical analysis was performed on data relevant to patients and laboratory items.

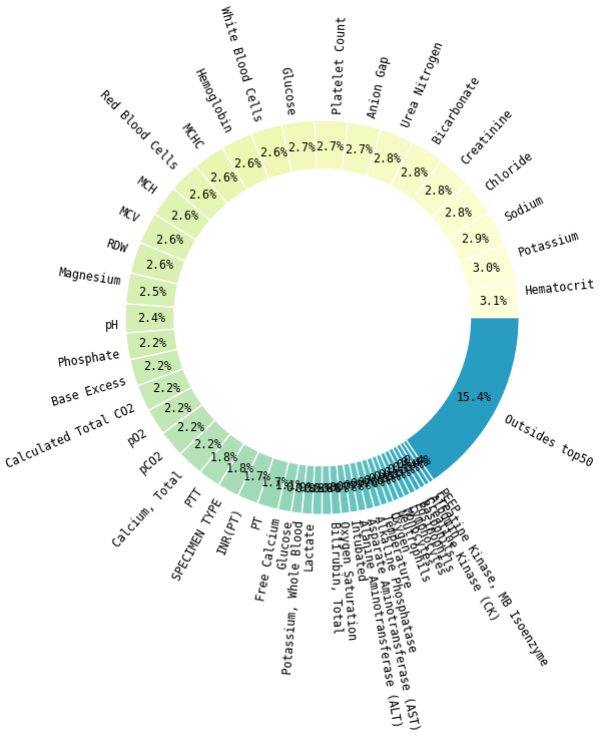
For all $58,976$ patients[3], we first counted the length of each patient's admission stay and drew the corresponding histogram to visualize the distribution. Usually, the length of admission stay depends on the patient's condition, so the corresponding histogram tends to show a skewed distribution which is due to the fact that the majority of patients do not stay too long. Evidence for this is shown in Fig.3a, for all patients, the mean length of admission stay is approximately 10.134 days, and the corresponding standard deviation is approximately 12.457 days. In addition, we also drew the histogram of the length of admission stay for patients whose final outcomes are death. For dead patients, their mean length of admission stay is approximately 10.127 days, and the corresponding standard deviation is approximately 13.928 days.

For all 753 laboratory items, we first performed statistics about categories. There are 423 items belonging to hematology, 293 items belonging to chemistry, and 37 items belonging to blood gas. If we differentiate these laboratory items by the sampling source, there are 374 items using blood, 109 items using urine, 91 items using other body fluid, 40 items using bone marrow, 38 items using ascites, 35 items using pleural, 31 items using cerebrospinal fluid, 28 items using joint fluid, and 7 items using stool. In addition, we counted the percentage of each laboratory item among the $22,245,034$ laboratory item records and listed the Top50 most frequently performed items according to the size of their percentage in Fig.3b.

[3]It should be clarified here that we treat each admission as an individual patient.



(a) Histogram of the length of admission stays within 30 days.

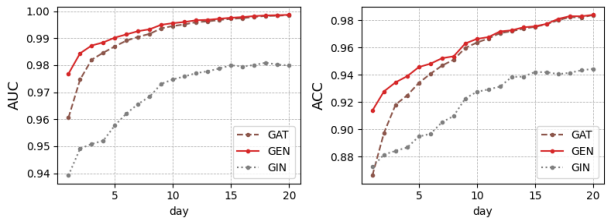


(b) Top50 most frequently performed laboratory items.

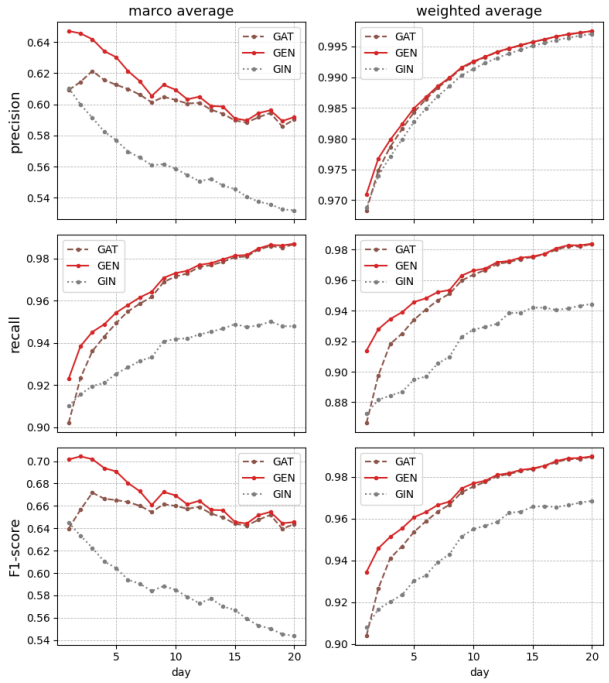
Figure 3: Statistical analysis of patients and laboratory items.

Evaluation of model performance

Owing to the laboratory items recommendation problem is converted to the edges predictions problem on the $\mathcal{DTDG}$ equivalently (stated in appendix A.1) which is essentially a binary classification problem, we use metrics including *Area Under the Curve (AUC), Accuracy (ACC), Precision, Recall, F1-score* to evaluate the model performance on validating set. Moreover, because the type of GNN encoders has multiple options (stated in appendix A.4.2) which will lead to different model performances and it would be more



(a) AUC and Accuracy performance at each timestep.



(b) Precision, Recall, and F1-score performance at each timestep calculated with different average methods. Note: the data points in the left half are calculated with macro-average which "averages the unweighted mean per label" (each class is assigned with equal weight), and the data points in the right half are calculated with weighted-average which "averages the support-weighted mean per label" (class with more proportion has greater weight).

Figure 4: Model Performance under different metrics (day $1 \sim t_{max} = 20$).

helpful to evaluate the model performance at each timestep individually rather than using all timesteps, we performed multiple experiments[4] with different types of GNN encoders and visualize their performances at each timestep individually (see Fig.4).

For AUC and Accuracy metrics, the lines plotted in both figures (see Fig.4a) show an upward trend along timesteps, which means the model performs better and better. We attribute this trend to the use of *masked self-attention mechanism* (stated in appendix A.4.3). From the perspective of the average, all GNN encoder variants achieve high AUC (≥ 0.9691) and Accuracy scores ($\geq 91.87\%$), and the GEN variant achieves the highest AUC (0.9935) and Accuracy scores (96.07%), these results mean the model's strong abilities to discriminate and make correct predictions. We also noticed that the GIN variant does not perform as well as the other two, which is mainly due to the fact that the GIN variant we built only uses a single-layer MLP when updating node features, while the other two use more complex mechanisms (e.g. GEN uses a 2-layer MLPs by default).

However, just the AUC and Accuracy scores do not provide a complete picture of the model's performance. It's essential to consider other evaluation metrics including *precision, recall*, and *F1-score* to get a comprehensive understanding of the model's effectiveness and practicality. Hence, we also visualize model performance on these metrics alone timesteps.

The precision score measures the ability of the model to avoid false positive results. In Fig.4b, if we only look at the weighted-average line on the top right, it appears that all variants achieve very high precision; however, the macro-average line drawn on the top left shows that the variants' precision score is not as high as the right, which is mainly due to the serious imbalance between positive and negative categories. In the prediction of model, the number of positive examples (required laboratory items) is far less than that of negative examples(non-required laboratory items). Therefore, little prediction errors in the positive examples will have a huge impact on the precision of the positive category, which is reflected in the final precision score calculated using macro-average[5]. But, in our case, a false positive prediction does not mean that it is bad; because more scheduled laboratory items do not harm patients like the misuse of drugs or surgery, it just gives additional, harmless information to clinicians.

The recall score measures the proportion of true positive predictions among the actual positive instances. In the corresponding two figures (mid of Fig.4b), the performance of all variants shows a high degree of agreement, which means that the prediction results of the positive and negative examples almost cover the ground truth. Meanwhile, as the information available to the model increases with time, the coverage becomes more and more complete. In clinical practice, this means that the positive predictions of the model are more and more consistent with the laboratory items actually scheduled by clinicians; in other words, our model rarely misses the laboratory items that should have been scheduled.

The F1-score considers both precision and recall, it indicates how effective a model is in correctly classifying instances into their respective classes while minimizing both false positives and false negatives. All variants perform well in the weighted-average plots but seem to perform poorly in macro-average plots (see bottom of Fig.4b). But,

[4]Details about the experiment settings are stated in appendix A.5

[5]The results in the weighted-average also support this interpretation. We can view this discussion on the web for more details about *macro/weighted average*: https://datascience.stackexchange.com/questions/65839

this poorness is due to the low and declining precision values calculated based on the macro-average (we have explained above why it's declining and why this decline isn't necessarily a bad thing). Hence, we conclude that the performance shown in the weighted-average plots is closer to the model's actual performance.

Altogether, through the experiments that comprehensively evaluate model performance using these metrics, we have a comprehensive understanding of the power of our proposed model which can implement laboratory items recommendations.

Discussion

In this work, we create a model to recommend laboratory items using $\mathcal{DTDG}$s. We aim to provide supplementary advice for clinicians when scheduling the items, which not only helps to reduce the probability of cognitive bias happening among clinicians but also helps to take the correct treatments for patients which would even save many lives! Specifically speaking, our model mainly consists of two parts: one is the encoders that use GNN for capturing the graph structure information in $\mathcal{DTDG}$s, and another is the decoders that use *masked self-attention mechanism* to learn the relationships of history time-sequences. The evaluation results prove the efficiency and practicality of our laboratory items recommendation model in the mimic-iii dataset. If we use the best-performing variant which takes GEN as GNN encoders for illustration, approximately 96.07% of the laboratory items in the first 20 days of hospitalization can be predicted correctly on average. Even more surprising, this variant also achieves approximately 96.77% recall score (calculated with macro-average), which means that the positive predictions of the model are highly consistent with the laboratory items actually scheduled by clinicians. Besides, although the false positive prediction (laboratory items that predicted not-scheduled but scheduled actually) rarely made by the model leads to a decline in the macro-average precision score, it will not harm the patient like the misuse of drugs or surgery but can bring additional and harmless (sometimes even useful for diagnosis) information to clinicians.

However, our model has several limitations deriving from its implement method (stated in appendix A). The first limitation is the lower model performance in the early days (can be observed in Fig.4). It's caused by the use of *masked self-attention mechanism* which restricts only previous information that can be taken into consideration. So, when using our model in real clinical scenarios, the laboratory items predicted as negative (not scheduled) in the early few days are not actually not-required; we suggest clinicians pay close attention to the laboratory items that are predicted to be positive (scheduled) by the model while missed by them, at the same time, supplement some necessary laboratory items with their own professional knowledge and experience. The second limitation is the neglect of other influencing factors in EMR. As shown in Fig.5, there are many other factors (such as medication, and procedure) that have an influence on the scheduling of laboratory items for patients. Let's illustrate this with a concrete example: in intensive postremission therapy of AML, the postremission strategies comprise intensive chemotherapy (medication) and high-dose therapy followed by autologous or allogeneic HCT (procedure); so, assessment of residual disease by RT-qPCR or MFC (specific laboratory items) is critical in monitoring patients in morphological remission to inform further therapy[2]. Our model does not take these factors into account, which explains why performance does not hit the upper bound; fortunately, due to the powerful modeling capabilities of discrete-time dynamic graphs, we can easily incorporate these factors in future work. The third limitation is about the currently determined $t_{max} = 20$ setting, which restricts our model can only process the first 20 timesteps of $\mathcal{DTDG}$s (stated in appendix A.4.2). Although the currently determined number is a trade-off based on the figure of the average number of scheduled laboratory items per patient per day during hospitalization (see Fig.10), we must point out that it is not a fixed number; in other words, there are alternatives of different values. The fourth limitation is the unknown generalization ability of our model. We only train and validate the model on the available mimic-iii dataset, which is a single-center database comprising information relating to patients admitted to critical care units[15]. While ICU patients generate mode EMR data than non-ICU patients, it is still uncertain how well techniques derived from this dataset will generalize to broader populations[16]. So, we expected there could be more available EMR dataset to test and verify the generalization ability of our model in the future.

Even though the improvable limitations mentioned above exist, the potential clinical applications of our laboratory items recommendation model are promising. Cognitive biases in clinical practice have a significant impact on care, often in negative ways[3]. Traditional approaches in the medical profession have attempted to prevent bias by educating clinicians about cognitive biases[17] and using cognitive psychology strategies (such as guided reflection[18], or cognitive forcing strategies[19]). However, the subjective nature of these approaches means that their practical effectiveness will vary greatly with the differences in expertise and experience of clinicians. In contrast, our proposed model is objective, and it will unbiasedly provide supplementary advice to clinicians about what laboratory items need to be scheduled next based on patients' historical results. Besides, as a review stated, the ability to take advantage of information stored in HIS (Hospital Information Systems) event logs can generate benefits associated with process efficiency, such as improving the quality of provided services[20]; another article envisages the ideal LIS (Laboratory Information System) could use patients' information, previous test results, and clinician input to suggest appropriate tests, test frequency, and interpretative criteria[21]. Their expectations coincided with our work. Therefore, we believe that our proposed model can play an

important role if it could be integrated into HIS or LIS. Final but not least, we also realized that our approach which uses the $\mathcal{DTDG}$s to represent EMR data can be extended to many other meaningful clinical tasks (e.g. the medication recommendation[22] or patient condition inference). Hence, perhaps this work can serve as a reference for various possible future applications that represent EMR with $\mathcal{DTDG}$s.

References

[1] Arati A Inamdar and Ajinkya C Inamdar. Heart failure: diagnosis, management and utilization. *Journal of clinical medicine*, 5(7):62, 2016.

[2] Hartmut Döhner, Elihu Estey, David Grimwade, Sergio Amadori, Frederick R Appelbaum, Thomas Büchner, Hervé Dombret, Benjamin L Ebert, Pierre Fenaux, Richard A Larson, et al. Diagnosis and management of aml in adults: 2017 eln recommendations from an international expert panel. *Blood, The Journal of the American Society of Hematology*, 129(4):424–447, 2017.

[3] Tiffany S Doherty and Aaron E Carroll. Believing in overcoming cognitive biases. *AMA Journal of Ethics*, 22(9):773–778, 2020.

[4] Graham Neale, Helen Hogan, and Nick Sevdalis. Misdiagnosis: analysis based on case record review with proposals aimed to improve diagnostic processes. *Clinical Medicine*, 11(4):317, 2011.

[5] Sudeh Cheraghi-Sohi, Fiona Holland, Hardeep Singh, Avril Danczak, Aneez Esmail, Rebecca Lauren Morris, Nicola Small, Richard Williams, Carl de Wet, Stephen M Campbell, et al. Incidence, origins and avoidable harm of missed opportunities in diagnosis: longitudinal patient record review in 21 english general practices. *BMJ Quality & Safety*, 30(12):977–985, 2021.

[6] James Reason. *Human error*. Cambridge university press, 1990.

[7] Yizhou Sun and Jiawei Han. Mining heterogeneous information networks: A structural analysis approach. *SIGKDD Explor. Newsl.*, 14(2):20–28, apr 2013.

[8] Wang-Cheng Kang, Mengting Wan, and Julian McAuley. Recommendation through mixtures of heterogeneous item relationships. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1143–1152, 2018.

[9] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

[10] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z Sheng, and Mehmet Orgun. Sequential recommender systems: challenges, progress and prospects. *Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[11] Jiarui Jin, Xianyu Chen, Weinan Zhang, Junjie Huang, Ziming Feng, and Yong Yu. Learn over past, evolve for future: Search-based time-aware recommendation with sequential behavior data. In *Proceedings of the ACM Web Conference 2022*, pages 2451–2461, 2022.

[12] Qijie Shen, Hong Wen, Wanjie Tao, Jing Zhang, Fuyu Lv, Zulong Chen, and Zhao Li. Deep interest highlight network for click-through rate prediction in trigger-induced recommendation. In *Proceedings of the ACM Web Conference 2022*, pages 422–430, 2022.

[13] Mengqi Zhang, Shu Wu, Xueli Yu, Qiang Liu, and Liang Wang. Dynamic graph neural networks for sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[14] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *The Journal of Machine Learning Research*, 21(1):2648–2720, 2020.

[15] Alistair E.W. Johnson, Tom J. Pollard, Lu Shen, Li wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3, 2016. https://doi.org/10.1038/sdata.2016.35.

[16] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. *Nature medicine*, 25(1):24–29, 2019.

[17] Pat Croskerry. The importance of cognitive errors in diagnosis and strategies to minimize them. *Academic medicine*, 78(8):775–780, 2003.

[18] Silvia Mamede and Henk G Schmidt. The structure of reflective practice in medicine. *Medical education*, 38(12):1302–1308, 2004.

[19] Pat Croskerry. Cognitive forcing strategies in clinical decisionmaking. *Annals of emergency medicine*, 41(1):110–120, 2003.

[20] Eric Rojas, Jorge Munoz-Gama, Marcos Sepúlveda, and Daniel Capurro. Process mining in healthcare: A literature review. *Journal of biomedical informatics*, 61:224–236, 2016.

[21] Jorge L Sepulveda and Donald S Young. The ideal laboratory information system. *Archives of Pathology and Laboratory Medicine*, 137(8):1129–1140, 2013.

[22] Rui Wu, Zhaopeng Qiu, Jiacheng Jiang, Guilin Qi, and Xian Wu. Conditional generation net for medication recommendation. In *Proceedings of the ACM Web Conference 2022*, pages 935–945, 2022.

[23] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[24] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

[25] Gilbert Reibnegger and Walter Schrabmair. Optimum binary cut-off threshold of a diagnostic test: comparison of different methods using monte carlo technique. *BMC medical informatics and decision making*, 14:1–9, 2014.

A Methods

In this section, we describe how we implement the laboratory items recommendations in detail.

A.1 Problem Formulation

We first describe the laboratory item recommendation problem using formula language, so that readers can have a general view first and facilitate the following description of specific methods. For convenience, we summarize the notations used here and their corresponding interpretations in Table 1.

Notation	Interpretations
$\mathcal{CTDG}$	continuous-time dynamic graph
$\mathcal{G}$	a static graph representing the initial state of a dynamic graph at time t_0
$\mathcal{O}$	a set of observations/events where each observation is a tuple *(event type, event, timestamp)*
$\mathcal{DTDG}$	discrete-time dynamic graph
$\mathcal{G}^i$	snapshots sampling at a specified time interval t_i from $\mathcal{CTDG}$
$\mathcal{V}$	set of nodes $\{v_0, \dots, v_i, \dots, v_j, \dots\}$
$\mathcal{E}$	set of edges e, such as e_{ij} that connects between node v_i and v_j
$\mathcal{T}$	set of timesteps
$\mathcal{P}$	set of patients
$\mathcal{I}$	set of laboratory items
$S^{\mathcal{I}}$	patient's laboratory items sequence
$T^{\mathcal{I}}$	corresponding timesteps sequence of $S^{\mathcal{I}}$

Table 1: Notations appearing in appendix A.1 and their corresponding interpretations.

A.1.1 Discrete-Time Dynamic Graph

Since the static graph can only reflect the state of multiple objects at a certain moment, it is not capable of constantly changing real scenarios. Therefore, dynamic graphs are proposed, which can be roughly divided into the following two categories[14]:

- *continuous-time dynamic graph*, annotated as $\mathcal{CTDG} = (\mathcal{G}, \mathcal{O})$, where $\mathcal{G}$ is a static graph representing the initial state of a dynamic graph at time t_0, $\mathcal{O}$ is a set of observations/events where each observation is a tuple *(event type, event, timestamp)*. An *event type* can be an edge addition, edge deletion, node addition, node deletion, node splitting, node merging, etc. An *event* is the operation corresponding to *event type*. Then the *timestamp* records when this observation/event happened. Hence, we can view the static graph as a snapshot obtained by updating the $\mathcal{G}$ according to $\mathcal{O}$ from t_0 to current *timestamp* t_n.
- *discrete-time dynamic graph*, annotated as $\mathcal{DTDG} = \{\mathcal{G}^1, \mathcal{G}^2, \dots, \mathcal{G}^n\}$ is a sequence of snapshots sampling at a specified time interval from $\mathcal{CTDG}$. A $\mathcal{DTDG}$ can be defined using triplet $(\mathcal{V}, \mathcal{E}, \mathcal{T})$, where $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$ is the nodes set and $e \in \mathcal{E}$ represents the interaction between V_i and v_j at timestep $t \in \mathcal{T}$, so edge e_{ij} is generally represented by triplet (v_i, v_j, t). By recording the timestep over all edges, the $\mathcal{DTDG}$ can capture the evolution of the relationship between nodes[13].

A.1.2 Laboratory Items Recommendation Problem

Because developing a model for $\mathcal{DTDG}$ is more feasible than $\mathcal{CTDG}$ in engineering, we organized sequential medical data (also called EMR, example is shown in Fig.5) to the $\mathcal{DTDG}$ form by a 24 hours time interval (reason for choosing this interval will be stated in appendix A.2.2). In this work, only records of laboratory items type are considered. So, each patient p belongs to the set of all patients $\mathcal{P} = (p_0, p_1, p_2, \dots)$ has a laboratory items sequence $S^{\mathcal{I}} = (i_0, i_1, i_2, \dots)$, where each i represents an item that belongs to the set of all laboratory items $\mathcal{I}$, and $T^{\mathcal{I}} = (t_0, t_1, t_2, \dots)$ is the corresponding timestep sequence of $S^{\mathcal{I}}$. The sets $\mathcal{P}$ and $\mathcal{I}$ are two different kinds of nodes $\mathcal{V}$ in $\mathcal{DTDG}$[6], the laboratory items sequences $S^{\mathcal{I}}$ of all patients indicate the edges $\mathcal{E}$ in $\mathcal{DTDG}$ and the corresponding timestep sequences $S^{\mathcal{I}}$ indicate the timesteps $\mathcal{T}$ of all edges in $\mathcal{DTDG}$. The laboratory items recommendation problem aims to predict the edges $\mathcal{E}$ at timestep t_{i+1} by using all past information contained in $\mathcal{DTDG}$ from timestep $t_0 \sim t_i$.

A.2 Processes of raw data

Because EMR (Electronic Medical Records) in hospitals usually contain personal information about patients, there is hardly any available EMR dataset on the web. However, we hope that our method is reproducible so that it can be applied to real hospital EMR databases. To our best knowledge, we only found the mimic-iii (Medical Information Mart for Intensive Care) dataset[15], which is de-identified and available on the web. As its original paper[15] states, mimic-iii is a large, single-center database comprising information relating to patients admitted to critical care units

[6] It is worth mentioning that this kind of graph with different types of nodes and the connections between them is called heterogeneous graph[7].

at a large tertiary care hospital. For each admission, many EMR will be logged, which include vital signs, medications, laboratory measurements, observations and notes charted by care providers, fluid balance, procedure codes, diagnostic codes, imaging reports, hospital length of stay, survival data, and so on (Fig.5 gives an illustrative example). In this work, only records related to laboratory items are considered; but their initial form is not suitable for our recommendation problem. Therefore, we need to perform the following processes in order to convert them to the $\mathcal{DTDG}$ form.

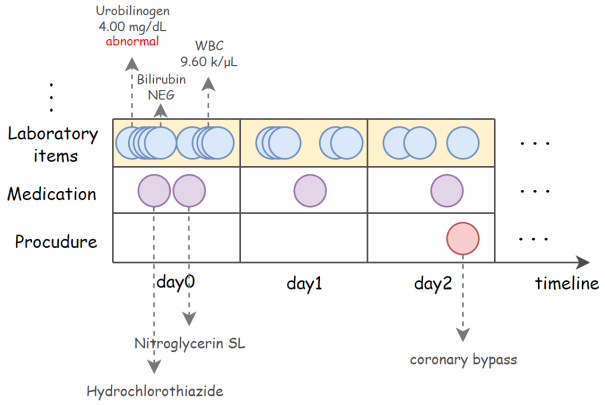


Figure 5: An illustrative example of the EMR logged during an admission.

A.2.1 Preprocessing

Data preprocessing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it, and it directly affects the learning ability of our model. Therefore, it is extremely important that we preprocess our data before feeding it into our model. There are 3 tables that we performed preprocessing on:

- *ADMISSIONS.csv.gz*: storing hospitalization information, including each patient's unique id-number, time of admission and discharge (will have the time of death if die), language, religion, and a lot of other information about this patient.
- *D_LABITEMS.csv.gz*: storing laboratory item codes, indexing every laboratory item.
- *LABEVENTS.csv.gz*: storing laboratory measurements, recording the results of laboratory items scheduled for patients.

In the first two tables, we mainly handled the categorical variables. In the tabular data whose raw file format is *".csv"*, there are many columns that store the nominal categorical variables (NCVs). For example, the *INSURANCE* column in table *ADMISSIONS.csv.gz* contains NCVs *"Private"* and *"Medicare"*, and these NCVs appear at different frequencies. But these NCVs are in string format, which is incomprehensible for the model. Therefore, we need to turn these NCVs into float format by encoding them in order of frequency, then the most frequent NCV will be encoded to 1, the second most frequent NCV will be encoded to 2, and so on (we use 0 to fill *"NaN"*). We did this encoding preprocessing for feature columns (will be used as nodes' feature in appendix A.3) in the first two tables.

In the last *LABEVENTS.csv.gz* table, we mainly handled the "multiple types of values" problem in the single *VALUE* column. As shown in Fig. 5, there are numeric laboratory item entries, such as "WBC" with a numeric value 9.60 and a unit "k/μL". However, there are also non-numeric laboratory item entries, such as "Bilirubin" with a non-numeric value "NEG". The root cause of this problem is the natural difference in the types of values among different laboratory items, some items naturally produce numeric values, while others are not. Therefore, we need to handle this problem in different ways depending on the value types of different laboratory items:

- *pure numeric value type* laboratory items: we need to do standardization, as the values of results among different items are not on the same scale. The standardized value reflects the outlier degree of the value, in other words, reflects the danger degree of the value. So, we performed standardization according to the formula $z = \frac{x_i - \mu}{\sigma}$ on each pure numeric value of each laboratory item, where μ and σ are mean and standard deviation respectively. Actually, μ and σ are not calculated on values of all entries of a pure numeric value laboratory item. Because we found that the existence of extreme outliers causes huge bias in calculations. So we calculated μ and σ only on selected entries whose *FLAG* column $\neq$ "abnormal" (based on the fact that the reference value of a laboratory item is generated based on the values of normal people). Meanwhile, we apply the 3σ principle to sort out outliers among these selected entries, so that we can avoid the mis-logged entries (extreme outliers whose *FLAG* column was not set to "abnormal").
- *non-numeric value type* laboratory items: just encoding these values as NCVs do, then record them in a new column *CATAGORY* (use 0 to fill "NaN" entries, which means that a pure numeric value type entry's *CATAGORY* column will be 0).
- *mix-types* laboratory items: just drop these entries. Why we do this way is because we found these entries were produced by the meaningless log into pure numeric value type laboratory items, such as "LESS THAN 0.1" or "$<$ 0.10", and so on.

A.2.2 Converting Timestamps

For sequential medical dataset like mimic-iii, we need to convert each entry's original absolute timestamp (e.g. "2117-09-11 08:22:00") in table *LABEVENTS.csv.gz* to a relative timestep (such as 1 or 3, and so on), which is an indispensable component for constructing our $\mathcal{DTDG}$s. Roughly speaking, a relative timestep represents how many intervals it has passed since the start time of the earliest entry of a patient was logged.

Let's illustrate with a concrete example: a patient with unique id=100001 logged an earliest laboratory items entry at absolute timestamp "2117-09-11 08:22:00". Then, if we choose 24 hours (a day) as the interval, the start time will be "2117-09-11 00:00:00", therefore, the relative timestep of the earliest entry would be 0. Along this line, patient 100001's last entry with absolute timestamp "2117-09-17 05:45:00" would have a relative timestep 6. Because 6 intervals have passed since the start time at "2117-09-11 00:00:00".

Note that choice of the interval will influence the final number of relative timesteps. So we need to choose a reasonable interval that is in line with the real situation. Finally, we chose 24 hours as the interval based on the following reasons:

- The workflows of clinicians are usually carried out day by day.
- The instruments used in the vast majority of laboratory items are not running all the time, they are usually started, run for a period of time to process samples, and then shut down during the day.
- The work and rest of patients also follow the day as the unit, most of them wake up in the morning, go through a series of medical services (containing the laboratory items that we focus on), and finally rest at night, so back and forth.

Moreover, we conducted the timestamp converting with the determined 24 hours time interval for all entries of each patient in table *LABEVENTS.csv.gz* (all timesteps below refer to relative timesteps). After conversion, we plotted the histogram of new relative timesteps among all entries for a simple check view. As shown in Fig.6, the number of days in hospital increases, while the number of laboratory items perform every day decreases.

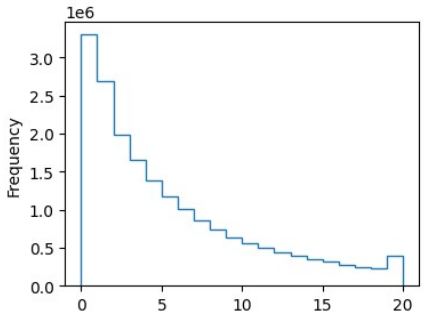


Figure 6: Histogram of the number of laboratory items per relative timestep. Here interval $=$ 24 hours, and the last bin in the x-axis contains all counts whose relative timestep ≥ 20.

A.2.3 Outdated Entries

The chosen interval of timestamp conversion brought another problem: if a patient did the same laboratory item multiple times during an interval (as shown in Fig.7), there would be outdated entries that cause trouble in constructing $\mathcal{DTDG}$ later in appendix A.3. According to the definition of $\mathcal{DTDG}$ in appendix A.1.1, the $\mathcal{G}^i$ should reflect the latest snapshot of $\mathcal{CTDG}$ during the i-th interval which it belongs to; besides, each heterogeneous graph $\mathcal{G}^i$ that is input into the model should not have redundant edges between the same *patient-item* nodes pair, which would cause ambiguity and make the model go into error. Therefore, we should drop the outdated entries, while keeping the latest entry.

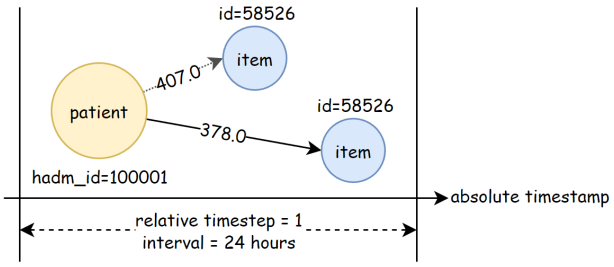


Figure 7: An illustrative example of outdated entries: a patient did a laboratory item twice at different absolute timestamps during an interval, therefore producing 2 entries that have different values. Only the latest entry in the interval is needed.

A.3 Graphs Constructing

In this section, we describe how the discrete-time dynamic graphs ($\mathcal{DTDG}$s) were constructed using the preprocessed data (stated from appendix A.2.1 to appendix A.2.3).

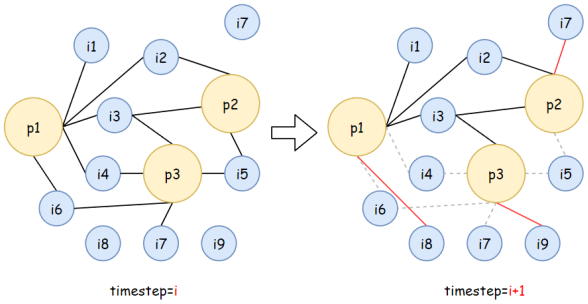


Figure 8: Simplified illustration of $\mathcal{DTDG}$. The $\mathcal{G}^i$ at t_i has patient nodes $\{p_1, p_2, p_3\}$, laboratory item nodes $\{i_1, i_2, \ldots, i_9\}$, and undirected edges between these nodes. Then, these edges change in the $\mathcal{G}^{i+1}$ at t_{i+1}: some are disconnected (gray dotted line), some are newly connected (red solid line), and some remain unchanged (black solid line).

overview Our final constructed $\mathcal{DTDG}$s are very similar to the simplified illustration of $\mathcal{DTDG}$ shown in Fig.8. There are two node types: p represents the patient, and i represents the laboratory item. In all $\mathcal{G}$s during all timestep ts, nodes remain the same, only the undirected edges be-

tween them change. These entities in $\mathcal{G}$ have their own node features or edge attributes, such as each patient node p has features $f^{(p)} \in \mathbb{R}^{d^{n(p)}}$, each laboratory item node i has features $f^{(i)} \in \mathbb{R}^{d^{n(i)}}$, and each undirected edge e has attributes $a^{(e)} \in \mathbb{R}^{d^{n(e)}}$.

Constructing When a laboratory item i was done by a patient p at timestep t with result r, an edge e was established between p and i. This edge e can be further represented by the triplet (p, i, t) we defined in appendix A.1.1, where p is a patient node with features $f^{(p)}$ comes from preprocessed *ADMISSIONS* table, i is a laboratory item with features $f^{(i)}$ comes from preprocessed *D_LABITEMS* table, the result r acts as edge e's attributes $a^{(e)}$, and t records the existing time of e. Due to every entry in the preprocessed *LABEVENTS* table corresponding to such a triplet, we can use these entries to form $\mathcal{DTDG} = \{(p, i, t) | p \in \mathcal{P}, i \in \mathcal{I}\}$, where $\mathcal{P}$ are set of all patients and $\mathcal{I}$ are set of all laboratory items. In actual implementation, instead of using all entries to form an incomparably huge $\mathcal{DTDG}$, we used subsets of patients $\mathcal{P}$ to form many $\mathcal{DTDG}$s which are more suitable for inputting into model and optimization. Then, for every $\mathcal{DTDG}$, we split it into $\{\mathcal{G}^0, \mathcal{G}^1, \dots, \mathcal{G}^i, \dots\}$ by the timestep before inputting it into model.

A.4 Architecture of The Recommendation Model

Review our goal that recommending what laboratory items need to do in the next day by using patients' history results. We designed a model to meet our goal. As shown in Fig.9, our model feeds a set of heterogeneous graphs $\{\mathcal{G}^0, \mathcal{G}^1, \dots, \mathcal{G}^i, \dots\}$ which come from splitting $\mathcal{DTDG}$ by timesteps into GNN encoders to capture the information from neighbor nodes and their corresponding edges (or called *semantics*), then it uses masked self-attention decoders to learn the relationships of history time-sequential information, then finally recommends which laboratory items need to do next (equivalent to edges prediction problem) using the output of decoders.

A.4.1 Aligning and Embedding

In order to make the dimensions of the features uniform and convenient for inputting into the encoders next, the dimension of each node features $f^{(p)}$ and $f^{(p)}$, each edge attributes $a^{(e)}$ are aligned into dimension d_h using trainable projection matrices respectively (in PyTorch, these matrices is achieved through fully connected layer).

Besides, due to each $\mathcal{DTDG}$ being constructed using the full laboratory items set $\mathcal{I}$, every laboratory item i is embedded with a unique vector representation ($\in \mathbb{R}^{d_h}$). These vector representations can bring additional, useful information to differentiate between laboratory items for our model.

A.4.2 GNN Encoders

As abundant information is contained in each $\mathcal{G}^i$, we need to capture it in order to make correct edges prediction. For example, in the $\mathcal{G}^i$ shown in Fig.8, patients nodes p_1, p_2, p_3 have the common laboratory item nodes i_2, i_3, i_4 revealing that these patients are in the same condition to some extent and these laboratory items probably are routines. Besides, the particular laboratory item node i_7 of patient node p_3 may imply i_7 is specially requested by doctors due to p_3's history condition. Therefore, enough powerful GNN model is needed for adapting to the complexity of information.

GIN[26] was the first powerful GNN model we noticed, and its source paper shows that GIN is *the most powerful GNN* by their theory. But since each $\mathcal{G}^i$ contains edge attribute $a^{(e)}$ which original GIN can not handle, we turned to use GINE[27] that modifies GIN in order to include the edge attribute $a^{(e)}$. Therefore, each node features f_i will be iteratively updated by the following formula:

$$f_i' = h_\Theta\left((1+\epsilon)\cdot f_i + \sum_{j \in \mathcal{N}(i)} \text{ReLU}(f_j + a_{j,i}^{(e)})\right)$$

where h_Θ is a neural network. We constructed every encoder with 2 layers of GINE, which means the update operation will perform 2 iterations to capture the 2-hop neighborhood information of each node. Then, every $\mathcal{G}^i$ is assigned an encoder to capture the neighborhood information.

Besides, we also noticed other two powerful GNN models that take edge attribute $a^{(e)}$ into consideration. The first one is called GATConv, which leverages masked self-attentional layers to specify different weights to different nodes in a neighborhood[28]. Each node features f_i will be iteratively updated by the GATConv's formula:

$$f_i' = \alpha_{i,i}\Theta f_i + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j}\Theta f_j$$

where $\alpha_{i,j}$ is the attention coefficient, and Θ is linear transformation parametrized by a weight matrix. And the second one is called GENConv, which proposes a suite of novel techniques (Generalized Aggregation Function, modified skip connection, graph normalization) to achieve SOTA performance[29]. Each node features f_i will be iteratively updated by the GENConv's formula:

$$f_i' = \text{MLP}\left(f_i + \sum_{j \in \mathcal{N}(i)} \text{AGG}\big(\text{ReLU}(f_j + e_{ji}) + \epsilon\big)\right)$$

where ϵ is the epsilon value of the message construction function. Every encoder of these two variants is also constructed with 2 layers in order to capture the 2-hop neighborhood information of each node.

However, due to our data having been constructed to $\mathcal{DTDG}$ form which has a set of heterogeneous graphs $\{\mathcal{G}^0, \mathcal{G}^1, \dots, \mathcal{G}^i, \dots\}$ of different timesteps, each $\mathcal{G}$ needs

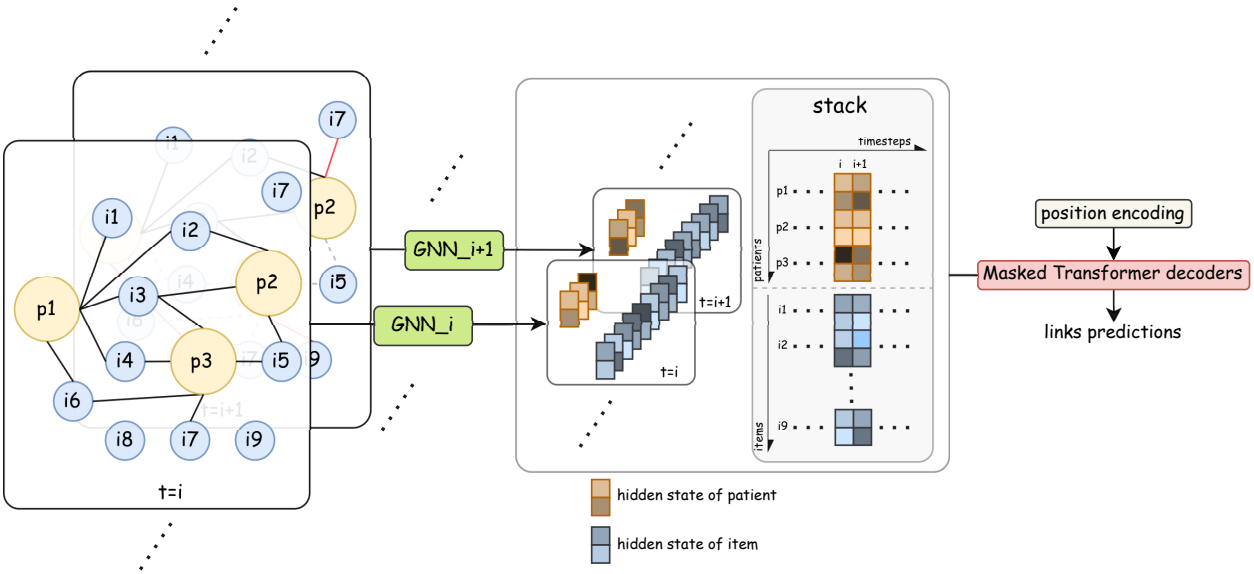


Figure 9: Overview of our laboratory items recommendation model. It mainly consists of two parts: one is the GNN encoders part which is used for capturing the graph structure information in discrete-time dynamic graphs, and another is the masked self-attention decoders part which is used to learn the relationships of history time-sequential information.

an encoder to capture information lying in it. So, in implementation, we can not increase the number of encoders without limit as the timestep has an obvious *marginal effect*: as Fig.10 shows, after 20 days, the average number of laboratory items per patient per day remained at a very low level, and after 50 days, the numbers are almost 0. Simultaneously, this emphmarginal effect is consistent with what we observed in Fig.3a: the length of hospital stay for most patients is usually not too long. So, we limit the max number of timesteps with parameter t_{max}. Correspondingly, the number of encoders is equal to t_{max}.

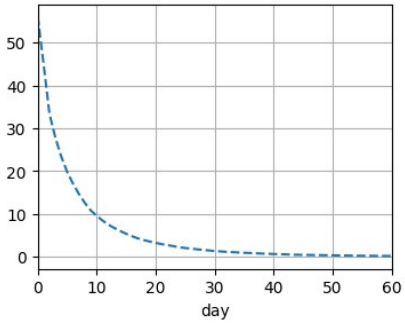


Figure 10: Curve of the average number of laboratory items performed on a patient per day during hospitalization. The line plotted in this figure shows us a very clear marginal effect.

A.4.3 Masked Self-Attention Decoders

After encoding the $\{\mathcal{G}^0, \mathcal{G}^1, \ldots, \mathcal{G}^i, \ldots\}$ series, the key question that how to compute the dependencies among different $\mathcal{G}$ at the different timestep emerges. As we know, for a patient, what laboratory items need to do next are depend on the history results, moreover, results from earlier or later days have different impacts. Hence, similar to translation tasks, this key question can also be regarded as a sequence modeling task. Therefore, a mechanism that relates different positions of a single sequence in order to compute a representation of the sequence is needed.

The previous alternatives are to use RNN, GRU[30], or LSTM[31]; DGSR[13] used them to model user's long- and short-term interest to items. But we didn't choose these mechanisms, as the Transformer-based model GPT-4[32] achieved unprecedented success which proves the self-attention mechanism's strong capabilities in sequence modeling. Besides, compared to previous mechanisms, there are 3 desiderata that motivate us to finally use the self-attention mechanism. First is the total computational complexity, the second is the parallel computing capabilities, and the third is the path length between long-range dependencies[7].

By using the self-attention mechanism, the attention scores Z at different positions in the sequence would be calculated by:

$$Z = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \tag{1}$$

and outputs O of the self-attention layer would be:

$$O = ZV \tag{2}$$

where d_k is scaling factor, and Q, K, V are Query, Key, Value matrices respectively. These Q, K, V matrices are obtained by packing input embeddings into a matrix

[7] can be found in section 4 of Transformer's original paper[33]

X and multiplying them with trainable weight matrices W^Q, W^K, W^V respectively.

In classical translation tasks, the X is the packed word embeddings with position encoding of a sentence[8]. In our case, the X is a node's series of hidden states packed along the timestep axis with position encoding (as shown in Fig.9's middle part). For example, if we set the t_{max} to 5 (not actual values, just for illustration here), then the shape of X for patient node p_1 would be $(5, d_k)$, as node p_1's hidden state along $\{\mathcal{G}^0, \ldots, \mathcal{G}^4\}$ are packed.

However, like the decoder side of Transformer which uses *Masked* operation when doing sequential translation (Only after the i-th word is translated, the i+1-th word can be translated), we should also use *Masked* operation to prevent node's hidden state at timestep i from getting information from timestep $> i$. As shown in Fig.11, if we continue with the X example whose shape is $(5, d_h)$, the QK^T matix would have a shape of $(5, 5)$, and the element $QK^T_{i,j}$ represents for the attention score ($\neq$ final Z_{ij}) of i-th to j-th position. Then the *Masked* operation is performed by using an upper triangular matrix of $-\infty$ to QK^T. Finally, the *softmax* function would set the attention score of these masked positions to 0.

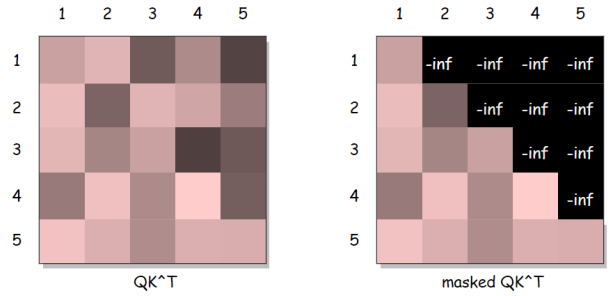


Figure 11: *Masked* operation performs on QK^T.

In order to model the dependencies of patients and laboratory items along different timesteps separately, we constructed 2 decoders respectively. And each decoder is composed of *L* self-attention layers, where the first layer is the masked self-attention layer and the others are common self-attention layers. After decoding, the output of patients $O^{(p)}$ and output of laboratory items $O^{(i)}$ would have shape of $(t_{max}, l^{(p)}, d_k)$ and $(t_{max}, l^{(i)}, d_k)$ respectively, where the $l^{(p)}$ and $l^{(i)}$ are length of $\mathcal{P}$ and $\mathcal{I}$ (described in appendix A.3) respectively[9].

A.4.4 Recommendation and Optimization

Due to the original data having been converted to $\mathcal{DTDG} = \{\mathcal{G}^0, \ldots, \mathcal{G}^i, \mathcal{G}^{i+1}, \ldots\}$ form, our original problem of *predicting what laboratory items need to do next* is equivalent to predicting which nodes will connect in the next $\mathcal{G}^{i+1}$ using historical $\{\mathcal{G}^0, \ldots, \mathcal{G}^i\}$ (typical edges prediction problem).

After decoders, $O^{(p)}$ and $O^{(i)}$ will be obtained. The recommendation R is calculated by:

$$R = O^{(p)} {O^{(i)}}^T. \tag{3}$$

Therefore, the R would have shape $(t_{max}, l^{(p)}, l^{(i)})$ which can be viewed as adjacency matrices distributed along the timestep axis. So, the natural ground true labels T of R are the adjacency matrices of $\{\mathcal{G}^1, \ldots, \mathcal{G}^{t_{max}}\}$ (note that $\{\mathcal{G}^0, \ldots, \mathcal{G}^{t_{max}-1}\}$ are input into our model). To learn the model's parameters, we optimize the *binary cross-entropy* loss of recommendation R and ground true labels T.

A.5 Experiment Settings

To comprehensively assess the efficiency and practicality of our model and inspect how the model performs at different timesteps, we conducted experiments on the validation set[10]. Here we list the parameter settings of experiments:

Dataset Partition All patients that exist in the dataset was partitioned into training and validating subsets to assess the model performance. The partitioning was performed using an 80-20 strategy, ensuring the representative distribution of data across both subsets. Before partitioning, a random shuffle will be done on the set $\mathcal{P}$ of all patients to guarantee the randomness. Then,

- Training set: 80% $(47,180/58,976)$ of the dataset was allocated for training. This subset was used to optimize the model's parameters and learn the underlying patterns in the data.
- Validating set: 20% $(11,796/58,976)$ of the dataset was reserved for validating the trained model's performance. This subset served as an independent evaluation set to measure the model's generalization ability and assess its performance on unseen data.

Parameters of Model We implemented all model variants in *PyG* Library. In the GNN encoders part, the hidden states dimension d_h is fixed to 64, the max number of timestep t_{max} is set to 20, and all variants have the same layers number 2. On the decoders side, the numbers of layers L in each decoder are both set to 6, and we reduced the dimension d_k of the feedforward network in each layer to 512[11]. Except for the model parameters mentioned above, we used the default settings for the rest of the model parameters.

hyperparameters All model variants are trained using AdamW[23] optimizer with a 0.0003 learning rate. In

[8] This blog gives a very understandable explanation: https://jalammar.github.io/illustrated-transformer

[9] Note: in PyTorch's default "nn.TransformerDecoder", the second dimension is batch.

[10] All experiments were conducted on a computer server with two NVIDIA GeForce RTX3090 (24GB) and 27 Intel i9-10940X CPUs.

[11] In PyTorch, the dimension is set to 2048 in default "nn.TransformerDecoderLayer", which is redundant for our cases.

addition, due to what we have mentioned in appendix A.3: instead of using all entries to form an incomparably huge $\mathcal{DTDG}$, we used subsets of patients $\mathcal{P}$ to form many $\mathcal{DTDG}$s which are more suitable for inputting into model and optimization. This approach is very similar to the *batch size* hyperparameter in the field of computer vision, which needs to be set to an appropriate size for the model to achieve the best possible performance under a fixed learning rate[24]. So, in order to find an appropriate size of each $\mathcal{P}$, we performed experiments with sizes equal to $\{128, 256, 512\}$, fixing the GINE[27] as GNN encoders and using AUC as the evaluation metric. Then we observe in Fig.12 that compared to the other two sizes, the model achieves best performance when the size of each $\mathcal{P}$ equals to 128[12]. So, under our fixed 0.0003 learning rate, choosing 128 as the size of each subset of patients $\mathcal{P}$ is appropriate.

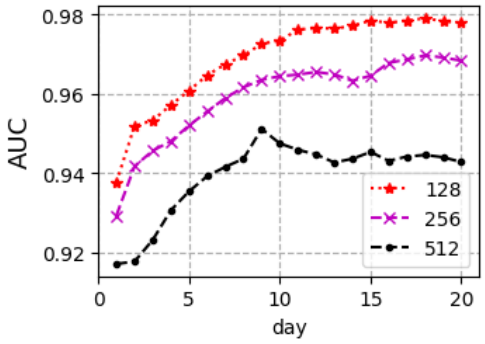


Figure 12: The experiment result of different sizes of $\mathcal{P}$. Note that the data is plotted from day 1 on the horizontal axis since our model predicts day i using information from the previous day(s).

Metrics To assess the model's performance of our binary classification task (edges prediction), we employed the following metrics:

- **Area Under the Curve (AUC)**: This metric is a widely used metric for evaluating the performance of binary classification models. It quantifies the model's ability to discriminate between positive and negative instances across different probability thresholds. A higher AUC value indicates a better-performing model.
- **Accuracy**: This metric measures the overall correctness of the predictions made by the model. It is calculated as the ratio of correct predictions to the total number of instances in the dataset.
- **Precision**: It quantifies the proportion of true positive predictions among the predicted positive instances. It is calculated as the ratio of true positives to the sum of true positives and false positives. Precision provides insights into the model's ability to avoid false positives.
- **Recall**: It is also known as sensitivity or true positive rate, which measures the proportion of true positive predictions among the actual positive instances. It is calculated as the ratio of true positives to the sum of true positives and false negatives. Recall indicates the model's effectiveness in capturing positive instances.
- **F1-score**[13]: It is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance by considering both precision and recall. F1-score is particularly useful when there is an imbalance between the positive and negative classes.

All these metrics are calculated in *sklearn.metrics* between model's recommendations R and ground ture labels L (stated in appendix A.4.4). Hence, they provide a comprehensive assessment of the model's performance, considering various aspects such as correctness, precision, recall, balance, and discrimination ability.

Data Availability

As the database contains detailed information regarding the clinical care of patients, it must be treated with appropriate care and respect. Researchers are required to formally request access via a process documented on the MIMIC website[14]. There are two key steps that must be completed before access is granted: first, the researcher must complete a recognized course in protecting human research participants that includes Health Insurance Portability and Accountability Act (HIPAA) requirements; second, the researcher must sign a data use agreement, which outlines appropriate data usage and security standards, and forbids efforts to identify individual patients. Approval requires at least a week. Once an application has been approved the researcher will receive emails containing instructions for downloading the database from PhysioNetWorks, a restricted access component of PhysioNet[15].

Code Availability

In this version for double-blind peer review, we attach the project code.

Appendix References

[26] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[12] Besides, we can also observe that model performs better on later days than on earlier days, and an increasing trend of AUC in Fig.12. This can be attributed to the *masked decoder mechanism* (mentioned in appendix A.4.3) which limits the model to only obtain information from previous times. In other words, the more information the model can obtain over time, the better the model will perform.

[13] Note: the thresholds we use when calculating the F1-score are got basing on *Youden Index*[25]

[14] https://mimic.mit.edu/

[27] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

[28] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[29] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. Deepergcn: All you need to train deeper gcns, 2020.

[30] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[31] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[32] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.