`RETAIN`: An Interpretable Predictive Model for Healthcare using Reverse Time Attention Mechanism

Edward Choi[*]**, Mohammad Taha Bahadori**[*]**, Joshua A. Kulas**[*]**,**
Andy Schuetz[†]**, Walter F. Stewart**[†]**, Jimeng Sun**[*]
[*] Georgia Institute of Technology   [†] Sutter Health
{mp2893,bahadori,jkulas3}@gatech.edu,
{schueta1,stewarwf}@sutterhealth.org, jsun@cc.gatech.edu

Abstract

Accuracy and interpretability are two dominant features of successful predictive models. Typically, a choice must be made in favor of complex black box models such as recurrent neural networks (RNN) for accuracy versus less accurate but more interpretable traditional models such as logistic regression. This tradeoff poses challenges in medicine where both accuracy and interpretability are important. We addressed this challenge by developing the REverse Time AttentIoN model (`RETAIN`) for application to Electronic Health Records (EHR) data. `RETAIN` achieves high accuracy while remaining clinically interpretable and is based on a two-level neural attention model that detects influential past visits and significant clinical variables within those visits (e.g. key diagnoses). `RETAIN` mimics physician practice by attending the EHR data in a reverse time order so that recent clinical visits are likely to receive higher attention. `RETAIN` was tested on a large health system EHR dataset with 14 million visits completed by 263K patients over an 8 year period and demonstrated predictive accuracy and computational scalability comparable to state-of-the-art methods such as RNN, and ease of interpretability comparable to traditional models.

1 Introduction

The broad adoption of Electronic Health Record (EHR) systems has opened the possibility of applying clinical predictive models to improve the quality of clinical care. Several systematic reviews have underlined the care quality improvement using predictive analysis [7, 22, 5, 18]. EHR data can be represented as temporal sequences of high-dimensional clinical variables (e.g., diagnoses, medications and procedures), where the sequence ensemble represents the documented content of medical visits from a single patient. Traditional machine learning tools summarize this ensemble into aggregate features, ignoring the temporal and sequence relationships among the feature elements. The opportunity to improve both predictive accuracy and interpretability is likely to derive from effectively modeling temporality and high-dimensionality of these event sequences.

Accuracy and interpretability are two dominant features of successful predictive models. There is a common belief that one has to trade accuracy for interpretability using one of three types of traditional models [6]: 1) identifying a set of rules (*e.g.* via decision trees [24]), 2) case-based reasoning by finding similar patients (*e.g.* via k-nearest neighbors [16] and distance metric learning [33]), and 3) identifying a list of risk factors (*e.g.* via LASSO coefficients [15]). While interpretable, all of these models rely on aggregated features, ignoring the temporal relation among features inherent to EHR data. As a consequence, model accuracy is sub-optimal. Latent-variable time-series models, such as [31, 32], account for temporality, but often have limited interpretation due to abstract state variables.

Recently, recurrent neural networks (RNN) have been successfully applied in modeling sequential EHR data to predict diagnoses [27] and model encounter sequences [11, 14]. But, the gain in accuracy
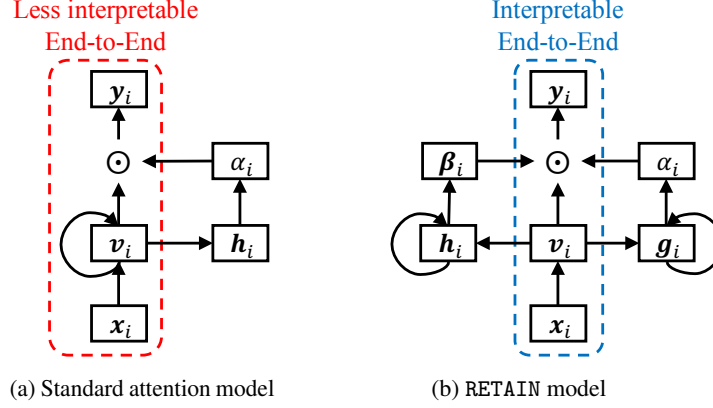
Figure 1: Common attention models vs. `RETAIN`, using folded diagrams of RNNs. (a) Standard attention mechanism: the recurrence on the hidden state vector $\mathbf{v}_i$ hinders interpretation of the model. (b) Attention mechanism in `RETAIN`: The recurrence is on the attention generation components ($\mathbf{h}_i$ or $\mathbf{g}_i$) while the hidden state $\mathbf{v}_i$ is generated by a simpler more interpretable output.

from use of RNNs is at the cost of model output that is notoriously difficult to interpret. While there have been several attempts at directly interpreting RNNs [17, 23, 8], these methods are not sufficiently developed for application in clinical care.

We have addressed this limitation using a modeling strategy known as `RETAIN`, a two-level neural attention model for sequential data that provides detailed interpretation of the prediction results while retaining the prediction accuracy comparable to RNN. To this end, `RETAIN` relies on an attention mechanism modeled to represent the behavior of physicians during an encounter. A distinguishing feature of `RETAIN` (see Figure 1) is to leverage sequence information using an attention generation mechanism, while learning an interpretable representation. And emulating physician behaviors, `RETAIN` examines a patient's past visits in reverse time order, facilitating a more stable attention generation. As a result, `RETAIN` identifies the most meaningful visits and quantifies visit specific features that contribute to the prediction.

`RETAIN` was tested on a large health system EHR dataset with 14 million visits completed by 263K patients over an 8 year period. We compared predictive accuracy of `RETAIN` to traditional machine learning methods and to RNN variants using a case-control dataset to predict a future diagnosis of heart failure. The comparative analysis demonstrates that `RETAIN` achieves comparable performance to RNN in both accuracy and speed and significantly outperforms traditional models. Moreover, using a concrete case study and visualization method, we demonstrate how `RETAIN` offers an intuitive interpretation.

2   Methodology

We first describe the structure of sequential EHR data and our notation, then follow with a general framework for predictive analysis in healthcare using EHR, followed by details of the `RETAIN` method.

EHR Structure and our Notation. The EHR data of each patient can be represented as a time-labeled sequence of multivariate observations. Assuming we use r different variables, the n-th patient of N total patients can be represented by a sequence of $T^{(n)}$ tuples $(t_i^{(n)}, \mathbf{x}_i^{(n)}) \in \mathbb{R} \times \mathbb{R}^r, i = 1, \ldots, T^{(n)}$. The timestamps $t_i^{(n)}$ denotes the time of the i-th visit of the n-th patient and $T^{(n)}$ the number of visits of the n-th patient. To minimize clutter, we describe the algorithms for a single patient and have dropped the superscript (n) whenever it is unambiguous. The goal of predictive modeling is to predict the label at each time step $\mathbf{y}_i \in \{0, 1\}^s$ or at the end of the sequence $\mathbf{y} \in \{0, 1\}^s$. The number of labels s can be more than one.

For example, in encounter sequence modeling (ESM) [11], each visit (*e.g.* encounter) of a patient's visit sequence is represented by a set of varying number of medical codes $\{c_1, c_2, \ldots, c_n\}$. c_j is the j-th code from the vocabulary $\mathcal{C}$. Therefore, in ESM, the number of variables $r = |\mathcal{C}|$ and input

$\mathbf{x}_i \in \{0,1\}^{|\mathcal{C}|}$ is a binary vector where the value one in the j-th coordinate indicates that c_j was documented in i-th visit. Given a sequence of visits $\mathbf{x}_1, \ldots, \mathbf{x}_T$, the goal of ESM is, for each time step i, to predict the codes occurring at the next visit $\mathbf{x}_2, \ldots, \mathbf{x}_{T+1}$, with the number of labels $s = |\mathcal{C}|$.

In case of learning to diagnose (L2D) [27], the input vector $\mathbf{x}_i$ consists of continuous clinical measures. If there are r different measurements, then $\mathbf{x}_i \in \mathbb{R}^r$. The goal of L2D is, given an input sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$, to predict the occurrence of a specific disease ($s = 1$) or multiple diseases ($s > 1$). Without loss of generality, we will describe the algorithm for ESM, as L2D can be seen as a special case of ESM where we make a single prediction at the end of the visit sequence.

In the rest of this section, we will use the abstract symbol RNN to denote any recurrent neural network variants that can cope with the vanishing gradient problem [3], such as LSTM [21], GRU [9], and IRNN [26], with any depth (number of hidden layers).

2.1 Preliminaries on Neural Attention Models

Attention based neural network models are being successfully applied to image processing [1, 29, 19, 34], natural language processing [2, 20, 30] and speech recognition [12]. The utility of the attention mechanism can be seen in the language translation task [2] where it is inefficient to represent an entire sentence with one fixed-size vector because neural translation machines finds it difficult to translate the given sentence represented by a single vector.

Intuitively, the attention mechanism for language translation works as follows: given a sentence of length S in the original language, we generate $\mathbf{h}_1, \ldots, \mathbf{h}_S$, to represent the words in the sentence. To find the j-th word in the target language, we generate attentions α_i^j for $i = 1, \ldots, S$ for each word in the original sentence. Then, we compute the context vector $\mathbf{c}_j = \sum_i \alpha_i^j \mathbf{h}_i$ and use it to predict the j-th word in the target language. In general, the attention mechanism allows the model to focus on a specific word (or words) in the given sentence when generating each word in the target language.

We rely on a conceptually similar temporal attention mechanism to generate interpretable prediction models using EHR data. Our model framework is motivated by and mimics how doctors attend to a patient's needs and explore the patient record, where there is a focus on specific clinical information (e.g., key risk factors) working from the present to the past.

2.2 Reverse Time Attention Model RETAIN

Figure 2 shows the high-level overview of our model, where a central feature is to delegate a considerable portion of the prediction responsibility to the process for generating attention weights. This is intended to address, in part, the difficulty with interpreting RNNs where the recurrent weights feed past information to the hidden layer. Therefore, to consider both the visit-level and the variable-level (individual coordinates of $\mathbf{x}_i$) influence, we use a linear embedding of the input vector $\mathbf{x}_i$. That is, we define

$$\mathbf{v}_i = \mathbf{W}_{emb} \mathbf{x}_i, \qquad \text{(Step 1)}$$

where $\mathbf{v}_i \in \mathbb{R}^m$ denotes the embedding of the input vector $\mathbf{x}_i \in \mathbb{R}^r$, m the size of the embedding dimension, $\mathbf{W}_{emb} \in \mathbb{R}^{m \times r}$ the embedding matrix to learn. We can alternatively use more sophisticated yet interpretable representations such as those derived from multilayer perceptron (MLP) [13, 25]. MLP has been used for representation learning in EHR data [10].

We use two sets of weights, one for the visit-level attention and the other for variable-level attention, respectively. The scalars $\alpha_1, \ldots, \alpha_i$ are the visit-level attention weights that govern the influence of each visit embedding $\mathbf{v}_1, \ldots, \mathbf{v}_i$. The vectors $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_i$ are the variable-level attention weights that focus on each coordinate of the visit embedding $v_{1,1}, v_{1,2}, \ldots, v_{1,m}, \ldots, v_{i,1}, v_{i,2}, \ldots, v_{i,m}$.

We use two RNNs, RNN_α and RNN_β, to separately generate α's and $\boldsymbol{\beta}$'s as follows,

$$\mathbf{g}_i, \mathbf{g}_{i-1}, \ldots, \mathbf{g}_1 = \text{RNN}_\alpha(\mathbf{v}_i, \mathbf{v}_{i-1}, \ldots, \mathbf{v}_1),$$

$$e_j = \mathbf{w}_\alpha^\top \mathbf{g}_j + b_\alpha, \quad \text{for} \quad j = 1, \ldots, i$$

$$\alpha_1, \alpha_2, \ldots, \alpha_i = \text{Softmax}(e_1, e_2, \ldots, e_i) \qquad \text{(Step 2)}$$

$$\mathbf{h}_i, \mathbf{h}_{i-1}, \ldots, \mathbf{h}_1 = \text{RNN}_{\boldsymbol{\beta}}(\mathbf{v}_i, \mathbf{v}_{i-1}, \ldots, \mathbf{v}_1)$$

$$\boldsymbol{\beta}_j = \tanh\left(\mathbf{W}_{\boldsymbol{\beta}} \mathbf{h}_j + \mathbf{b}_{\boldsymbol{\beta}}\right) \quad \text{for} \quad j = 1, \ldots, i, \qquad \text{(Step 3)}$$
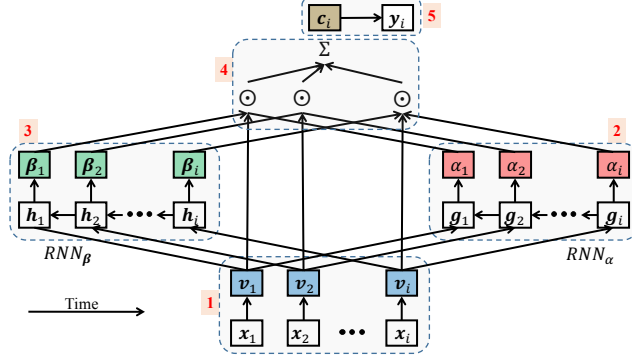
3

Figure 2: Unfolded view of RETAIN's architecture: Given input sequence $\mathbf{x}_1, \ldots, \mathbf{x}_i$, we predict the label $\mathbf{y}_i$. **Step 1**: Embedding, **Step 2**: generating α values using RNN_α, **Step 3**: generating $\boldsymbol{\beta}$ values using RNN_β, **Step 4**: Generating the context vector using attention and representation vectors, and **Step 5**: Making prediction. Note that in Steps 2 and 3 we use RNN in the reversed time.

where $\mathbf{g}_i \in \mathbb{R}^p$ is the hidden layer of RNN_α at time step i, $\mathbf{h}_i \in \mathbb{R}^q$ the hidden layer of RNN_β at time step i and $\mathbf{w}_\alpha \in \mathbb{R}^p, b_\alpha \in \mathbb{R}, \mathbf{W}_\beta \in \mathbb{R}^{m \times q}$ and $\mathbf{b}_\beta \in \mathbb{R}^m$ are the parameters to learn. The hyperparameters p and q determine the hidden layer size of RNN_α and RNN_β, respectively. Note that for prediction at each timestamp, we generate a new set of attention vectors α and $\boldsymbol{\beta}$. For simplicity of notation, we do not include the index for predicting at different time steps. In Step 2, we can use Sparsemax [28] instead of Softmax for sparser attention weights.

As noted, RETAIN generates the attention vectors by running the RNNs backward in time; i.e., RNN_α and RNN_β both take the visit embeddings in a reverse order $\mathbf{v}_i, \mathbf{v}_{i-1}, \ldots, \mathbf{v}_1$. Running the RNN in reversed time order also offers computational advantages since the reverse time order allows us to generate e's and β's that dynamically change their values when making predictions at different time steps $i = 1, 2, \ldots, T$. This ensures that the attention vectors are modified at each time step, increasing the computational stability of the attention generation process.[1]

Using the generated attentions, we obtain the context vector $\mathbf{c}_i$ for a patient up to the i-th visit as follows,

$$\mathbf{c}_i = \sum_{j=1}^{i} \alpha_j \boldsymbol{\beta}_j \odot \mathbf{v}_j, \tag{Step 4}$$

where $\odot$ denotes element-wise multiplication. We use the context vector $\mathbf{c}_i \in \mathbb{R}^m$ to predict the true label $\mathbf{y}_i \in \{0, 1\}^s$ as follows,

$$\widehat{\mathbf{y}}_i = \text{Softmax}(\mathbf{W}\mathbf{c}_i + \mathbf{b}), \tag{Step 5}$$

where $\mathbf{W} \in \mathbb{R}^{s \times m}$ and $\mathbf{b} \in \mathbb{R}^s$ are parameters to learn. We use the cross-entropy to calculate the classification loss as follows,

$$\mathcal{L}(\mathbf{x}_1, \ldots, \mathbf{x}_T) = -\frac{1}{N} \sum_{n=1}^{N} \frac{1}{T^{(n)}} \sum_{i=1}^{T^{(n)}} \left(\mathbf{y}_i^\top \log(\widehat{\mathbf{y}}_i) + (\mathbf{1} - \mathbf{y}_i)^\top \log(\mathbf{1} - \widehat{\mathbf{y}}_i) \right) \tag{1}$$

where we sum the cross entropy errors from all dimensions of $\widehat{\mathbf{y}}_i$. In case of real-valued output $\mathbf{y}_i \in \mathbb{R}^s$, we can change the cross-entropy in Eq. (1) to, for example, mean squared error.

Overall, our attention mechanism can be viewed as the inverted architecture of the standard attention mechanism for NLP [2] where the words are encoded by RNN and the attention weights are generated by MLP. In contrast, our method uses MLP to embed the visit information to preserve interpretability and uses RNN to generate two sets of attention weights, recovering the sequential information as well as mimicking the behavior of physicians. Note that we did not use the timestamp of each visit in our formulation. Using timestamps, however, provides a small improvement in the prediction performance. We propose a method to use timestamps in Appendix A.

---

[1]For example, feeding visit embeddings in the original order to RNN_α and RNN_β will generate the same e_1 and β_1 for every time step $i = 1, 2, \ldots, T$. Moreover, in many cases, a patient's recent visit records deserve more attention than the old records. Then we need to have $e_{j+1} > e_j$ which makes the process computationally unstable for long sequences.

3 Interpreting RETAIN

Finding the visits that contribute to prediction are derived using the largest α_i, which is straightforward. However, finding influential variables is slightly more involved as a visit is represented by an ensemble of medical variables, each of which can vary in its predictive contribution. The contribution of each variable is determined by $\mathbf{v}$, $\boldsymbol{\beta}$ and α, and interpretation of α alone informs which visit is influential in prediction but not why.

We propose a method to interpret the end-to-end behavior of RETAIN. By keeping α and $\boldsymbol{\beta}$ values fixed as the attention of doctors, we analyze changes in the probability of each label $y_{i,1}, \ldots, y_{i,s}$ in relation to changes in the original input $x_{1,1}, \ldots, x_{1,r}, \ldots, x_{i,1}, \ldots, x_{i,r}$. The $x_{j,k}$ that yields the largest change in $y_{i,d}$ will be the input variable with highest contribution. More formally, given the sequence $\mathbf{x}_1, \ldots, \mathbf{x}_i$, we are trying to predict the probability of the output vector $\mathbf{y}_i \in \{0,1\}^s$, which can be expressed as follows

$$p(\mathbf{y}_i | \mathbf{x}_1, \ldots, \mathbf{x}_i) = p(\mathbf{y}_i | \mathbf{c}_i) = \mathrm{Softmax}\left(\mathbf{W}\mathbf{c}_i + \mathbf{b}\right) \tag{2}$$

where $\mathbf{c}_i \in \mathbb{R}^m$ denotes the context vector. According to Step 4, $\mathbf{c}_i$ is the sum of the visit embeddings $\mathbf{v}_1, \ldots, \mathbf{v}_i$ weighted by the attentions α's and $\boldsymbol{\beta}$'s. Therefore Eq (2) can be rewritten as follows,

$$p(\mathbf{y}_i | \mathbf{x}_1, \ldots, \mathbf{x}_i) = p(\mathbf{y}_i | \mathbf{c}_i) = \mathrm{Softmax}\left(\mathbf{W}\Big(\sum_{j=1}^{i} \alpha_j \boldsymbol{\beta}_j \odot \mathbf{v}_j\Big) + \mathbf{b}\right) \tag{3}$$

Using the fact that the visit embedding $\mathbf{v}_i$ is the sum of the columns of $\mathbf{W}_{emb}$ weighted by each element of $\mathbf{x}_i$, Eq (3) can be rewritten as follows,

$$
\begin{aligned}
p(\mathbf{y}_i | \mathbf{x}_1, \ldots, \mathbf{x}_i) &= \mathrm{Softmax}\left(\mathbf{W}\Big(\sum_{j=1}^{i} \alpha_j \boldsymbol{\beta}_j \odot \sum_{k=1}^{r} x_{j,k} \mathbf{W}_{emb}[:,k]\Big) + \mathbf{b}\right) \\
&= \mathrm{Softmax}\left(\sum_{j=1}^{i} \sum_{k=1}^{r} x_{j,k}\, \alpha_j \mathbf{W}\Big(\boldsymbol{\beta}_j \odot \mathbf{W}_{emb}[:,k]\Big) + \mathbf{b}\right)
\end{aligned}
\tag{4}
$$

where $x_{j,k}$ is the k-th element of the input vector $\mathbf{x}_j$. Eq (4) can be completely deconstructed to the variables at each input $\mathbf{x}_1, \ldots, \mathbf{x}_i$, which allows for calculating the contribution ω of the k-th variable of the input $\mathbf{x}_j$ at time step $j \leq i$, for predicting $\mathbf{y}_i$ as follows,

$$\omega(\mathbf{y}_i, x_{j,k}) = \underbrace{\alpha_j \mathbf{W}(\boldsymbol{\beta}_j \odot \mathbf{W}_{emb}[:,k])}_{\text{Contribution coefficient}} \underbrace{x_{j,k}}_{\text{Input value}}, \tag{5}$$

where the index i of $\mathbf{y}_i$ is omitted in the α_j and $\boldsymbol{\beta}_j$. As we have described in Section 2.2, we are generating α's and $\boldsymbol{\beta}$'s at time step i in the visit sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$. Therefore the index i is always assumed for α's and $\boldsymbol{\beta}$'s. Additionally, Eq (5) shows that when we are using a binary input value, the coefficient itself is the contribution. However, when we are using a non-binary input value, we need to multiply the coefficient and the input value $x_{j,k}$ to correctly calculate the contribution.

4 Experiments

We compared performance of RETAIN to RNNs and traditional machine learning methods. Given space constraints, we only report the results on the learning to diagnose (L2D) task and summarize the encounter sequence modeling (ESM) in Appendix C. The RETAIN source code is publicly available at https://github.com/mp2893/retain.

4.1 Experimental setting

Source of data: The dataset consists of electronic health records from Sutter Health. The patients are 50 to 80 years old adults chosen for a heart failure prediction model study. From the encounter records, medication orders, procedure orders and problem lists, we extracted visit records consisting of diagnosis, medication and procedure codes. To reduce the dimensionality while preserving the clinical information, we used existing medical groupers to aggregate the codes into input variables. The details of the medical groupers are given in the Appendix B. A profile of the dataset is summarized in Table 1.

Table 1: Statistics of EHR dataset. (D:Diagnosis, R:Medication, P:Procedure)

# of patients	263,683	Avg. # of codes in a visit	3.03
# of visits	14,366,030	Max # of codes in a visit	62
Avg. # of visits per patient	54.48	Avg. # of Dx codes in a visit	1.83
# of medical code groups	615 (D:283, R:94, P:238)	Max # of Dx in a visit	42

Implementation details: We implemented `RETAIN` with Theano 0.8 [4]. For training the model, we used Adadelta [35] with the mini-batch of 100 patients. The training was done in a machine equipped with Intel Xeon E5-2630, 256GB RAM, two Nvidia Tesla K80's and CUDA 7.5.

Baselines: For comparison, we completed the following models.

- **Logistic regression (LR)**: We compute the counts of medical codes for each patient based on all her visits as input variables and normalize the vector to zero mean and unit variance. We use the resulting vector to train the logistic regression.
- **MLP**: We use the same feature construction as **LR**, but put a hidden layer of size 256 between the input and output.
- **RNN**: RNN with two hidden layers of size 256 implemented by the GRU. Input sequences $\mathbf{x}_1, \ldots, \mathbf{x}_i$ are used. Logistic regression is applied to the top hidden layer. We use two layers of RNN of to match the model complexity of `RETAIN`.
- **RNN+$\boldsymbol{\alpha}_M$**: One layer single directional RNN (hidden layer size 256) along time to generate the input embeddings $\mathbf{v}_1, \ldots, \mathbf{v}_i$. We use the MLP with a single hidden layer of size 256 to generate the visit-level attentions $\alpha_1, \ldots, \alpha_i$. We use the input embeddings $\mathbf{v}_1, \ldots, \mathbf{v}_i$ as the input to the MLP. This baseline corresponds to Figure 1a.
- **RNN+$\boldsymbol{\alpha}_R$**: This is similar to **RNN+$\boldsymbol{\alpha}_M$** but uses the reverse-order RNN (hidden layer size 256) to generate the visit-level attentions $\alpha_1, \ldots, \alpha_i$. We use this baseline to confirm the effectiveness of generating the attentions using reverse time order.

The comparative visualization of the baselines are provided in Appendix D. We use the same implementation and training method for the baselines as described above. The details on the hyper-parameters, regularization and drop-out strategies for the baselines are described in Appendix B.

Evaluation measures: Model accuracy was measured by:

- **Negative log-likelihood** that measures the model loss on the test set. The loss can be calculated by Eq (1).
- **Area Under the ROC Curve (AUC)** of comparing $\widehat{y}_i$ with the true label y_i. AUC is more robust to imbalanced positive/negative prediction labels, making it appropriate for evaluation of classification accuracy in the heart failure prediction task.

We also report the bootstrap (10,000 runs) estimate of the standard deviation of the evaluation measures.

4.2 Heart Failure Prediction

Objective: Given a visit sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$, we predicted if a primary care patient will be diagnosed with heart failure (HF). This is a special case of ESM with a single disease outcome at the end of the sequence. Since this is a binary prediction task, we use the logistic sigmoid function instead of the Softmax in Step 5.

Cohort construction: From the source dataset, 3,884 cases are selected and approximately 10 controls are selected for each case (28,903 controls). The case/control selection criteria are fully described in the supplementary section. Cases have index dates to denote the date they are diagnosed with HF. Controls have the same index dates as their corresponding cases. We extract diagnosis codes, medication codes and procedure codes in the 18-months window before the index date.

Training details: The patient cohort was divided into the training, validation and test sets in a 0.75:0.1:0.15 ratio. The validation set was used to determine the values of the hyper-parameters. See Appendix B for details of hyper-parameter tuning.

Table 2: Heart failure prediction performance of `RETAIN` and the baselines

Model	Test Neg Log Likelihood	AUC	Train Time / epoch	Test Time
LR	0.3269 ± 0.0105	0.7900 ± 0.0111	0.15s	0.11s
MLP	0.2959 ± 0.0083	0.8256 ± 0.0096	0.25s	0.11s
RNN	0.2577 ± 0.0082	0.8706 ± 0.0080	10.3s	0.57s
RNN+α_M	0.2691 ± 0.0082	0.8624 ± 0.0079	6.7s	0.48s
RNN+α_R	0.2605 ± 0.0088	$\mathbf{0.8717} \pm 0.0080$	10.4s	0.62s
`RETAIN`	$\mathbf{0.2562} \pm 0.0083$	0.8705 ± 0.0081	10.8s	0.63s

Results: Logistic regression and MLP underperformed compared to the four temporal learning algorithms (Table 2). `RETAIN` is comparable to the other RNN variants in terms of prediction performance while offering the interpretation benefit.

Note that RNN+α_R model are a degenerated version of `RETAIN` with only scalar attention, which is still a competitive model as shown in table 2. This confirms the efficiency of generating attention weights using the RNN. However, RNN+α_R model only provides scalar visit-level attention, which is not sufficient for healthcare applications. Patients often receives several medical codes at a single visit, and it will be important to distinguish their relative importance to the target. We show such a case study in section 4.3.

Table 2 also shows the scalability of `RETAIN`, as its training time (the number of seconds to train the model over the entire training set once) is comparable to RNN. The test time is the number of seconds to generate the prediction output for the entire test set. We use the mini-batch of 100 patients when assessing both training and test times. RNN takes longer than RNN+α_M because of its two-layer structure, whereas RNN+α_M uses a single layer RNN. The models that use two RNNs (RNN, RNN+α_R, `RETAIN`)[2] take similar time to train for one epoch. However, each model required a different number of epochs to converge. RNN typically takes approximately 10 epochs, RNN+α_M and RNN+α_R 15 epochs and `RETAIN` 30 epochs. Lastly, training the attention models (RNN+α_M, RNN+α_R and `RETAIN`) for ESM would take considerably longer than L2D, because ESM modeling generates context vectors at each time step. RNN, on the other hand, does not require additional computation other than embedding the visit to its hidden layer to predict target labels at each time step. Therefore, in ESM, the training time of the attention models will increase linearly in relation to the length of the input sequence.

4.3 Model Interpretation for Heart Failure Prediction

We evaluated the interpretability of `RETAIN` in the HF prediction task by choosing a HF patient from the test set and calculating the contribution of the variables (medical codes in this case) to diagnostic prediction. The patient suffered from skin problems, *skin disorder* (SD), *benign neoplasm* (BN), *excision of skin lesion* (ESL), for some time before showing symptoms of HF, *cardiac dysrhythmia* (CD), *heart valve disease* (HVD) and *coronary atherosclerosis* (CA), and then a diagnosis of HF (Figure 3). We can see that skin-related codes from the earlier visits made little contribution to HF prediction as expected. `RETAIN` properly puts much attention to the HF-related codes that occurred in recent visits.

To confirm `RETAIN`'s ability to exploit the sequence information of the EHR data, we reverse the visit sequence of Figure 3a and feed it to `RETAIN`. Figure 3b shows the contribution of the medical codes of the reversed visit record. HF-related codes in the past are still making positive contributions, but not as much as they did in Figure 3a. Figure 3b also emphasizes `RETAIN`'s superiority to interpretable, but stationary models such as logistic regression. Stationary models often aggregate past information and remove the temporality from the input data, which can mistakenly lead to the same risk prediction for Figure 3a and 3b. `RETAIN`, however, can correctly digest the sequence information and calculates the HF risk score of 9.0%, which is significantly lower than that of Figure 3a.

Figure 3c shows how the contributions of codes change when selected medication data are used in the model. We added two medications from day 219: *antiarrhythmics* (AA) and *anticoagulants* (AC), both of which are used to treat *cardiac dysrhythmia* (CD). The two medications make a negative contributions, especially towards the end of the record. The medications decreased the positive contributions of *heart valve disease* and *cardiac dysrhythmia* in the last visit. Indeed, the HF risk

---

[2]The RNN baseline uses two layers of RNN, RNN+α_R uses one for visit embedding and one for generating α, `RETAIN` uses each for generating α and $\boldsymbol{\beta}$
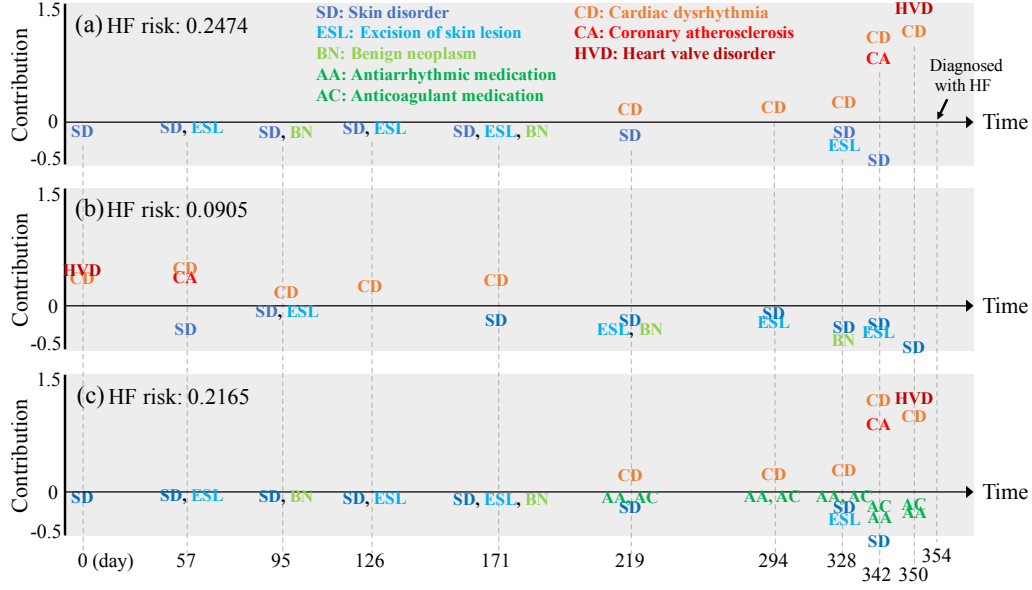
Figure 3: (a) Temporal visualization of a patient's visit records where the contribution of variables for diagnosis of heart failure (HF) is summarized along the x-axis (*i.e.* time) with the y-axis indicating the magnitude of visit and code specific contributions to HF diagnosis. (b) We reverse the order of the visit sequence to see if RETAIN can properly take into account the modified sequence information. (c) Medication codes are added to the visit record to see how it changes the behavior of RETAIN.

prediction (0.2165) of Figure 3c is lower than that of Figure 3a (0.2474). This suggests that taking proper medications can help the patient in reducing their HF risk.

5 Conclusion

Our approach to modeling event sequences as predictors of HF diagnosis suggest that complex models can offer both superior predictive accuracy and more precise interpretability. Given the power of RNNs for analyzing sequential data, we proposed RETAIN, which preserves RNN's predictive power while allowing a higher degree of interpretation. The key idea of RETAIN is to improve the prediction accuracy through a sophisticated attention generation process, while keeping the representation learning part simple for interpretation, making the entire algorithm accurate and interpretable. RETAIN trains two RNN in a reverse time order to efficiently generate the appropriate attention variables. For future work, we plan to develop an interactive visualization system for RETAIN and evaluating RETAIN in other healthcare applications.

References

[1] J. Ba, V. Mnih, and K. Kavukcuoglu. Multiple object recognition with visual attention. In *ICLR*, 2015.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[3] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994.

[4] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of SciPy*, 2010.

[5] A. D. Black, J. Car, C. Pagliari, C. Anandan, K. Cresswell, T. Bokun, B. McKinstry, R. Procter, A. Majeed, and A. Sheikh. The impact of ehealth on the quality and safety of health care: a systematic overview. *PLoS Med*, 8(1):e1000387, 2011.

[6] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *KDD*, 2015.

[7] B. Chaudhry, J. Wang, S. Wu, M. Maglione, W. Mojica, E. Roth, S. C. Morton, and P. G. Shekelle. Systematic review: impact of health information technology on quality, efficiency, and costs of medical care. *Annals of internal medicine*, 144(10):742–752, 2006.

[8] Z. Che, S. Purushotham, R. Khemani, and Y. Liu. Distilling knowledge from deep networks with applications to healthcare domain. *arXiv preprint arXiv:1512.03542*, 2015.

[9] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

[10] E. Choi, M. T. Bahadori, E. Searles, C. Coffey, and J. Sun. Multi-layer representation learning for medical concepts. In *KDD*, 2016.

[11] E. Choi, M. T. Bahadori, and J. Sun. Doctor ai: Predicting clinical events via recurrent neural networks. *arXiv preprint arXiv:1511.05942*, 2015.

[12] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. Attention-based models for speech recognition. In *NIPS*, pages 577–585, 2015.

[13] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 2009.

[14] C. Esteban, O. Staeck, Y. Yang, and V. Tresp. Predicting clinical events by combining static and dynamic information using recurrent neural networks. *arXiv preprint arXiv:1602.02685*, 2016.

[15] A. S. Fleisher, B. B. Sowell, C. Taylor, A. C. Gamst, R. C. Petersen, L. J. Thal, and f. t. A. D. C. Study. Clinical predictors of progression to Alzheimer disease in amnestic mild cognitive impairment. *Neurology*, 68(19):1588–1595, 2007.

[16] B. Gallego, S. R. Walter, R. O. Day, A. G. Dunn, V. Sivaraman, N. Shah, C. A. Longhurst, and E. Coiera. Bringing cohort studies to the bedside: framework for a 'green button' to support clinical decision-making. *Journal of Comparative Effectiveness Research*, pages 1–7, 2015.

[17] J. Ghosh and V. Karamcheti. Sequence learning with recurrent networks: analysis of internal representations. In *Aerospace Sensing*, pages 449–460, 1992.

[18] C. L. Goldzweig, A. Towfigh, M. Maglione, and P. G. Shekelle. Costs and benefits of health information technology: new trends from the literature. *Health affairs*, 28(2):w282–w293, 2009.

[19] K. Gregor, I. Danihelka, A. Graves, and D. Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[20] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching machines to read and comprehend. In *NIPS*, pages 1684–1692, 2015.

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

[22] A. K. Jha, C. M. DesRoches, E. G. Campbell, K. Donelan, S. R. Rao, T. G. Ferris, A. Shields, S. Rosenbaum, and D. Blumenthal. Use of electronic health records in us hospitals. *N Engl J Med*, 2009.

[23] A. Karpathy, J. Johnson, and F.-F. Li. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

[24] A. N. Kho et al. Use of diverse electronic medical record systems to identify genetic risk for type 2 diabetes within a genome-wide association study. *JAMIA*, 19(2):212–218, 2012.

[25] Q. V. Le. Building high-level features using large scale unsupervised learning. In *ICASSP*, 2013.

[26] Q. V. Le, N. Jaitly, and G. E. Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

[27] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzell. Learning to Diagnose with LSTM Recurrent Neural Networks. In *ICLR*, 2016.

[28] A. F. Martins and R. F. Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *ICML*, 2016.

[29] V. Mnih, N. Heess, A. Graves, et al. Recurrent models of visual attention. In *NIPS*, 2014.

[30] A. M. Rush, S. Chopra, and J. Weston. A neural attention model for abstractive sentence summarization. In *EMNLP*, 2015.

[31] S. Saria, D. Koller, and A. Penn. Learning individual and population level traits from clinical temporal data. In *NIPS, Predictive Models in Personalized Medicine workshop*, 2010.

[32] P. Schulam and S. Saria. A probabilistic graphical model for individualizing prognosis in chronic, complex diseases. In *AMIA*, volume 2015, page 143, 2015.

[33] J. Sun, F. Wang, J. Hu, and S. Edabollahi. Supervised patient similarity measure of heterogeneous patient records. *ACM SIGKDD Explorations Newsletter*, 14(1):16–24, 2012.

[34] K. Xu, J. Ba, R. Kiros, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

[35] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

A A method to use the timestamps

As before, we use $t_i^{(n)}$ to represent the timestamp of the i-th visit of the n-th patient. In the following, we suppress the superscript (n) to avoid cluttered notation. Note that the timestamp t_i can be anything that provides the temporal information of the i-th visit: the number of days from the first visit, the number of days between two consecutive visits, or the number of days until the index date of some event such as heart failure diagnosis.

In order to use the timestamps, we modify Step 2 and Step 3 in Section 2.2 as follows:

$$\mathbf{g}_i, \mathbf{g}_{i-1}, \ldots, \mathbf{g}_1 = \text{RNN}_\alpha(\mathbf{v}_i', \mathbf{v}_{i-1}', \ldots, \mathbf{v}_1'),$$
$$e_j = \mathbf{w}_\alpha^\top \mathbf{g}_j + b_\alpha, \quad \text{for} \quad j = 1, \ldots, i$$
$$\alpha_1, \alpha_2, \ldots, \alpha_i = \text{Softmax}(e_1, e_2, \ldots, e_i)$$
$$\mathbf{h}_i, \mathbf{h}_{i-1}, \ldots, \mathbf{h}_1 = \text{RNN}_{\boldsymbol{\beta}}(\mathbf{v}_i', \mathbf{v}_{i-1}', \ldots, \mathbf{v}_1')$$
$$\boldsymbol{\beta}_j = \tanh\left(\mathbf{W}_{\boldsymbol{\beta}} \mathbf{h}_j + \mathbf{b}_{\boldsymbol{\beta}}\right) \quad \text{for} \quad j = 1, \ldots, i,$$
$$\text{where } \mathbf{v}_i' = [\mathbf{v}_i, t_i]$$

where we use $\mathbf{v}_i'$, the concatenation of the visit embedding $\mathbf{v}_i$ and the timestamp t_i, to generate the attentions α and $\boldsymbol{\beta}$. However, when obtaining the context vector $\mathbf{c}_i$ as per Step 4, we use $\mathbf{v}_i$, not $\mathbf{v}_i'$ to match the dimensionality. The entire process could be understood such that we use the temporal information not to embed each visit, but to calculate the attentions for the entire visit sequence. This is consistent with our modeling approach where we lose the sequential information in embedding the visit with MLP, then recover the sequential information by generating the attentions using the RNN. By using the temporal information, specifically the log of the number of days from the first visit, we were able to improve the heart failure prediction AUC by 0.003 without any hyper-parameter tuning.

B Details of the experiment settings

B.1 Hyper-parameter Tuning

We used the validation set to tune the hyper-parameters: visit embedding size m, RNN_α's hidden layer size p, $\text{RNN}_{\boldsymbol{\beta}}$'s hidden layer size q, L_2 regularization coefficient, and drop-out rates.

L_2 regularization was applied to all weights except the ones in RNN_α and $\text{RNN}_{\boldsymbol{\beta}}$. Two separate drop-outs were used on the visit embedding $\mathbf{v}_i$ and the context vector $\mathbf{c}_i$. We performed the random search with predefined ranges $m, p, q \in \{32, 64, 128, 200, 256\}$, $L_2 \in \{0.1, 0.01, 0.001, 0.0001\}$, $dropout_{\mathbf{v}_i}, dropout_{\mathbf{c}_i} \in \{0.0, 0.2, 0.4, 0.6, 0.8\}$. We also performed the random search with m, p and q fixed to 256.

The final value we used to train RETAIN for heart failure prediction is $m, p, q = 128$, $dropout_{\mathbf{v}_i} = 0.6$, $dropout_{\mathbf{c}_i} = 0.6$ and 0.0001 for the L_2 regularization coefficient.

B.2 Code Grouper

Diagnosis codes, medication codes and procedure codes in the dataset are respectively using International Classification of Diseases (ICD-9), Generic Product Identifier (GPI) and Current Procedural Terminology (CPT).

Diagnosis codes are grouped by Clinical Classifications Software for ICD-9[3] which reduces the number of diagnosis code from approximately 14,000 to 283. Medication codes are grouped by Generic Product Identifier Drug Group[4] which reduces the dimension to from approximately 151,000 to 96. Procedure codes are grouped by Clinical Classifications Software for CPT[5], which reduces the number of CPT codes from approximately 9,000 to 238.

---

[3]https://www.hcup-us.ahrq.gov/toolssoftware/ccs/ccs.jsp
[4]http://www.wolterskluwercdi.com/drug-data/medi-span-electronic-drug-file/
[5]https://www.hcup-us.ahrq.gov/toolssoftware/ccs_svcsproc/ccssvcproc.jsp

Table 3: Qualifying ICD-9 codes for heart failure

ICD-9 Code	Description
398.91	Rheumatic heart failure (congestive)
402.01	Malignant hypertensive heart disease with heart failure
402.11	Benign hypertensive heart disease with heart failure
402.91	Unspecified hypertensive heart disease with heart failure
404.01	Hypertensive heart and chronic kidney disease, malignant, with heart failure and with chronic kidney disease stage I through stage IV, or unspecified
404.03	Hypertensive heart and chronic kidney disease, malignant, with heart failure and with chronic kidney disease stage V or end stage renal disease
404.11	Hypertensive heart and chronic kidney disease, benign, with heart failure and with chronic kidney disease stage I through stage IV, or unspecified
404.13	Hypertensive heart and chronic kidney disease, benign, with heart failure and chronic kidney disease stage V or end stage renal disease
404.91	Hypertensive heart and chronic kidney disease, unspecified, with heart failure and with chronic kidney disease stage I through stage IV, or unspecified
404.93	Hypertensive heart and chronic kidney disease, unspecified, with heart failure and chronic kidney disease stage V or end stage renal disease
428.0	Congestive heart failure, unspecified
428.1	Left heart failure
428.20	Systolic heart failure, unspecified
428.21	Acute systolic heart failure
428.22	Chronic systolic heart failure
428.23	Acute on chronic systolic heart failure
428.30	Diastolic heart failure, unspecified
428.31	Acute diastolic heart failure
428.32	Chronic diastolic heart failure
428.33	Acute on chronic diastolic heart failure
428.40	Combined systolic and diastolic heart failure, unspecified
428.41	Acute combined systolic and diastolic heart failure
428.42	Chronic combined systolic and diastolic heart failure
428.43	Acute on chronic combined systolic and diastolic heart failure
428.9	Heart failure, unspecified

B.3 Training Specifics of the Basline Models

- **LR**: We use 0.01 L_2 regularization coefficient for the logistic regression weight.

- **MLP**: We use drop-out rate 0.6 on the output of the hidden layer. We use 0.0001 L_2 regularization coefficient for the hidden layer weight and the logistic regression weight.

- **RNN**: We use drop-out rate 0.6 on the outputs of both hidden layers. We use 0.0001 L_2 regularization coefficient for the logistic regression weight. The dimension size of both hidden layers is 256.

- **RNN+α_M**: We use drop-out rate 0.4 on the output of the hidden layer and 0.6 on the output of the context vector $\sum_i \alpha_i \mathbf{v}_i$. We use 0.0001 L_2 regularization coefficient for the hidden layer weight of the MLP that generates α's and the logistic regression weight. The dimension size of the hidden layers in both RNN and MLP is 256.

- **RNN+α_R**: We use drop-out rate 0.4 on the output of the hidden layer and 0.6 on the output of the context vector $\sum_i \alpha_i \mathbf{v}_i$. We use 0.0001 L_2 regularization coefficient for the hidden layer weight of the RNN that generates α's and the logistic regression weight. The dimension size of the hidden layers in both RNNs is 256.

B.4 Heart Failure Case/Control Selection Criteria

Case patients were 40 to 85 years of age at the time of HF diagnosis. HF diagnosis (HFDx) is defined as: 1) Qualifying ICD-9 codes for HF appeared in the encounter records or medication orders. Qualifying ICD-9 codes are displayed in Table 3. 2) a minimum of three clinical encounters with qualifying ICD-9 codes had to occur within 12 months of each other, where the date of diagnosis was assigned to the earliest of the three dates. If the time span between the first and second appearances of the HF diagnostic code was greater than 12 months, the date of the second encounter was used as the

first qualifying encounter. The date at which HF diagnosis was given to the case is denoted as HFDx. Up to ten eligible controls (in terms of sex, age, location) were selected for each case, yielding an overall ratio of 9 controls per case. Each control was also assigned an index date, which is the HFDx of the matched case. Controls are selected such that they did not meet the operational criteria for HF diagnosis prior to the HFDx plus 182 days of their corresponding case. Control subjects were required to have their first office encounter within one year of the matching HF case patient's first office visit, and have at least one office encounter 30 days before or any time after the case's HF diagnosis date to ensure similar duration of observations among cases and controls.

C Results on encounter sequence modeling

Objective: Given a sequence of visits $\mathbf{x}_1, \ldots, \mathbf{x}_T$, the goal of encounter sequence modeling is, for each time step i, to predict the codes occurring at the next visit $\mathbf{x}_2, \ldots, \mathbf{x}_{T+1}$. In this experiment, we focus on predicting the diagnosis codes in the encounter sequence, so we create a separate set of labels $\mathbf{y}_1, \ldots, \mathbf{y}_T$ that do not contain non-diagnosis codes such as medication codes or procedure codes. Therefore $\mathbf{y}_i$ will contain diagnosis codes from the next visit $\mathbf{x}_{i+1}$.

Dataset: We divide the entire dataset described in Table 1 into 0.75:0.10:0.15 ratio, respectively for training set, validation set, and test set.

Baseline: We use the same baseline models we used for HF prediction. However, since we are predicting 283 binary labels now, we replace the logistic regression function with the Softmax function. The drop-out and L_2 regularization policies remain the same.

For LR and MLP, at each step i, we aggregate maximum ten past input vectors[6] $\mathbf{x}_{i-9}, \ldots, \mathbf{x}_i$ to create a pseudo-context vector $\widehat{\mathbf{c}}_i$. LR applies the Softmax function on top of $\widehat{\mathbf{c}}_i$. MLP places a hidden layer on top of $\widehat{\mathbf{c}}_i$ then applies the Softmax function.

Evaluation metric: We use the negative log likelihood Eq (1) on the test set to evaluate the model performance. We also use Recall@k as an additional metric to measure the prediction accuracy.

- **Recall@k:** Given a sequence of visits $\mathbf{x}_1, \ldots, \mathbf{x}_T$, we evaluate the model performance based on how accurately it can predict the diagnosis codes $\mathbf{y}_1, \ldots, \mathbf{y}_T$. We use the average Recall@k, which is expressed as below,

$$\frac{1}{N} \sum_{n=1}^{N} \frac{1}{T^{(n)}} \sum_{i=1}^{T^{(n)}} \text{Recall@}k(\widehat{\mathbf{y}}_i), \quad where \quad \text{Recall@}k(\widehat{\mathbf{y}}_i) = \frac{|\text{argsort}(\widehat{\mathbf{y}}_i)[: k] \cap nonzero(\mathbf{y}_i)|}{|nonzero(\mathbf{y}_i)|}$$

  where $argsort$ returns a list of indices that will decrementally sort a given vector and $nonzero$ returns a list of indices of the coordinates with non-zero values. We use Recall@k because of its similar nature to the way a human physician performs the differential diagnostic procedure, which is to generate a list of most likely diseases for an undiagnosed patient, then perform medical practice until the true disease, or diseases are determined.

Prediction accuracy: Table 4 displays the prediction performance of RETAIN and the baselines. We use $k = 5, 10$ for Recall@k to allow a reasonable number of prediction trials, as well as cover complex patients who often receive multiple diagnosis codes at a single visit.

RNN shows the best prediction accuracy for encounter diagnosis prediction. However, considering the purpose of encounter diagnosis prediction, which is to assist doctors to provide quality care for the patient, black-box behavior of RNN makes it unattractive as a clinical tool. On the other hand, RETAIN performs as well as other attention models, only slightly inferior to RNN, provides full interpretation of its prediction behavior, making it a feasible solution for clinical applications.

The interesting finding in Table 4 is that MLP is able to perform as accurately as RNN+α_M in terms of Recall@10. Considering the fact that MLP uses aggregated information of past ten visits, we can assume that encounter diagnosis prediction depends more on the frequency of disease occurrences rather than the order in which they occurred. This is quite different from the HF prediction task, where stationary models (LF, MLP) performed significantly worse than sequential models.

---

[6]We also tried aggregating all past input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_i$, but the performance was slightly worse than using just ten.

Table 4: Encounter diagnosis prediction performance of RETAIN and the baselines

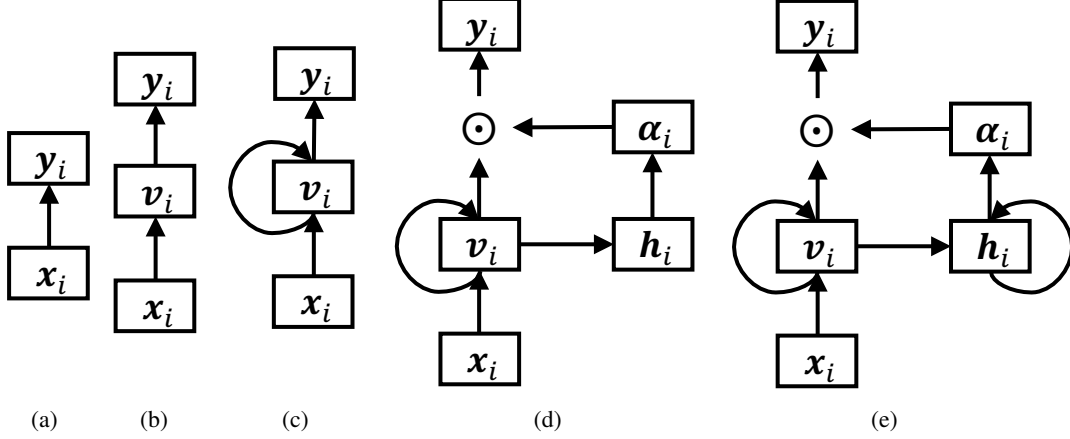| Model	Negative Likelihood	Recall@5	Recall@10
LR	0.0288	43.15	55.84
MLP	0.0267	50.72	65.02
RNN	**0.0258**	**55.18**	**69.09**
RNN+α_M	0.0262	52.15	65.81
RNN+α_R	0.0259	53.89	67.45
RETAIN	0.0259	54.25	67.74

Figure 4: Graphical illustration of the baselines: (a) Logistic regression (LR), (b) Multilayer Perceptron (MLP), (c) Recurrent neural network (RNN), (d) RNN with attention vectors generated via an MLP (RNN+α_M), (e) RNN with attention vectors generated via an RNN (RNN+α_R). RETAIN is given in Figure 1b.

D   Illustration and comparison of the baselines

Figure 4 illustrates the baselines used in the experiments and shows the relationship among them.