Concept/ Patient Representation

- Following Edward Choi's Ideas -

Introduction

Concept Representation

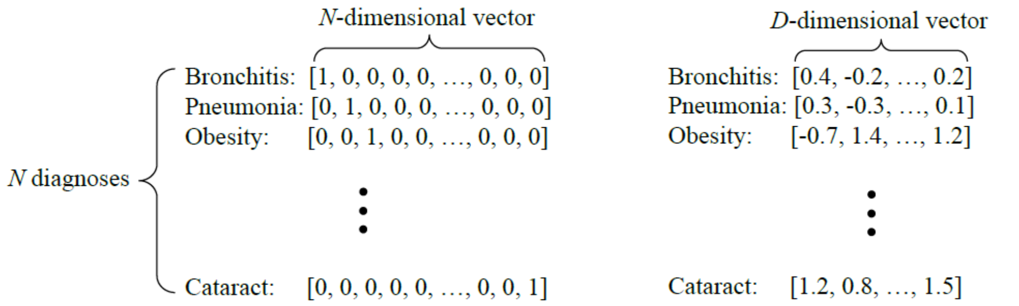
- Diagnosis, treatments, and medication codes create thousands of dummy variables (one-hot encoding) -> Sparse matrix
- Statistical models usually re-group(coarse classing) dummy variables.
- NLP techniques(word embedding) for medical concepts.
- 'Word2Vec' provides a few interesting features such as vector operation.

Patient Representation

- Prediction models in general require patient level data (i.e. disease prediction)
- Concept representation can be transformed to patient presentation.
  - ✓ However, summation/average of concept vectors loses temporal information as well as interpretability.
- E. Choi tries to incorporate sequential information whilst making the models interpretable at the same time.

Concept Presentation

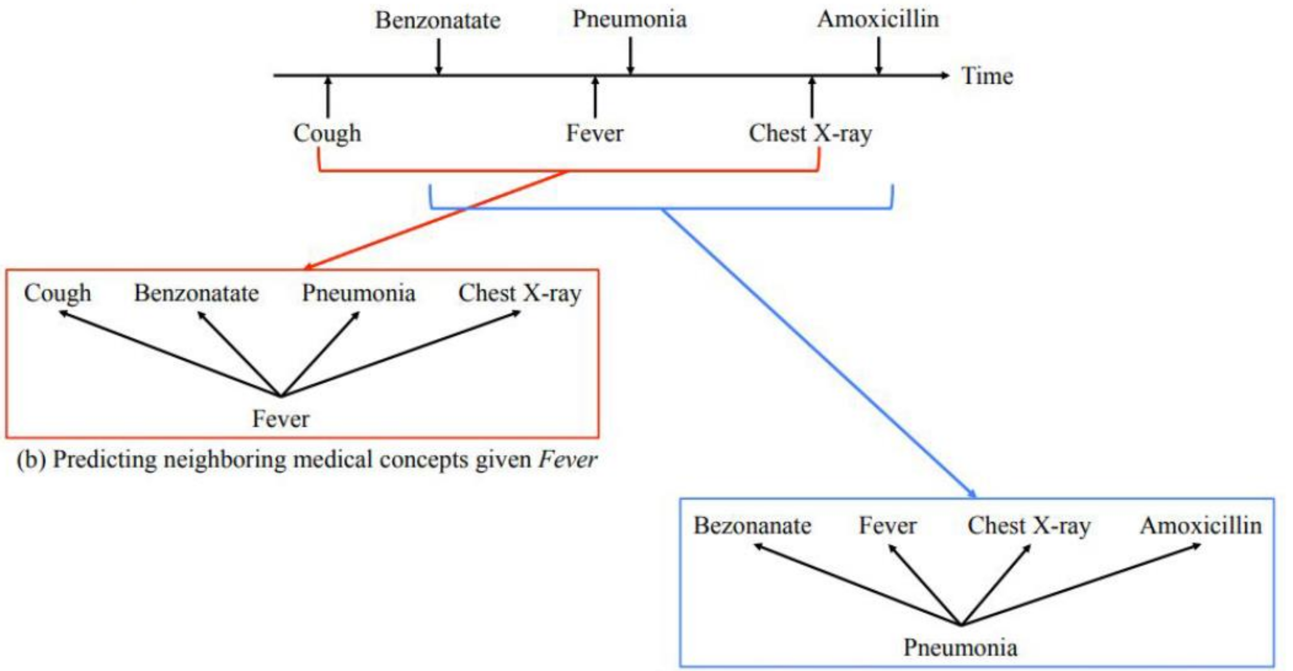
Medical ‘Word2Vec’



(a) One-hot encoding for diagnoses (b) A better representation of diagnoses

- Skip-gram training examples

(a) Patient medical records on a timeline

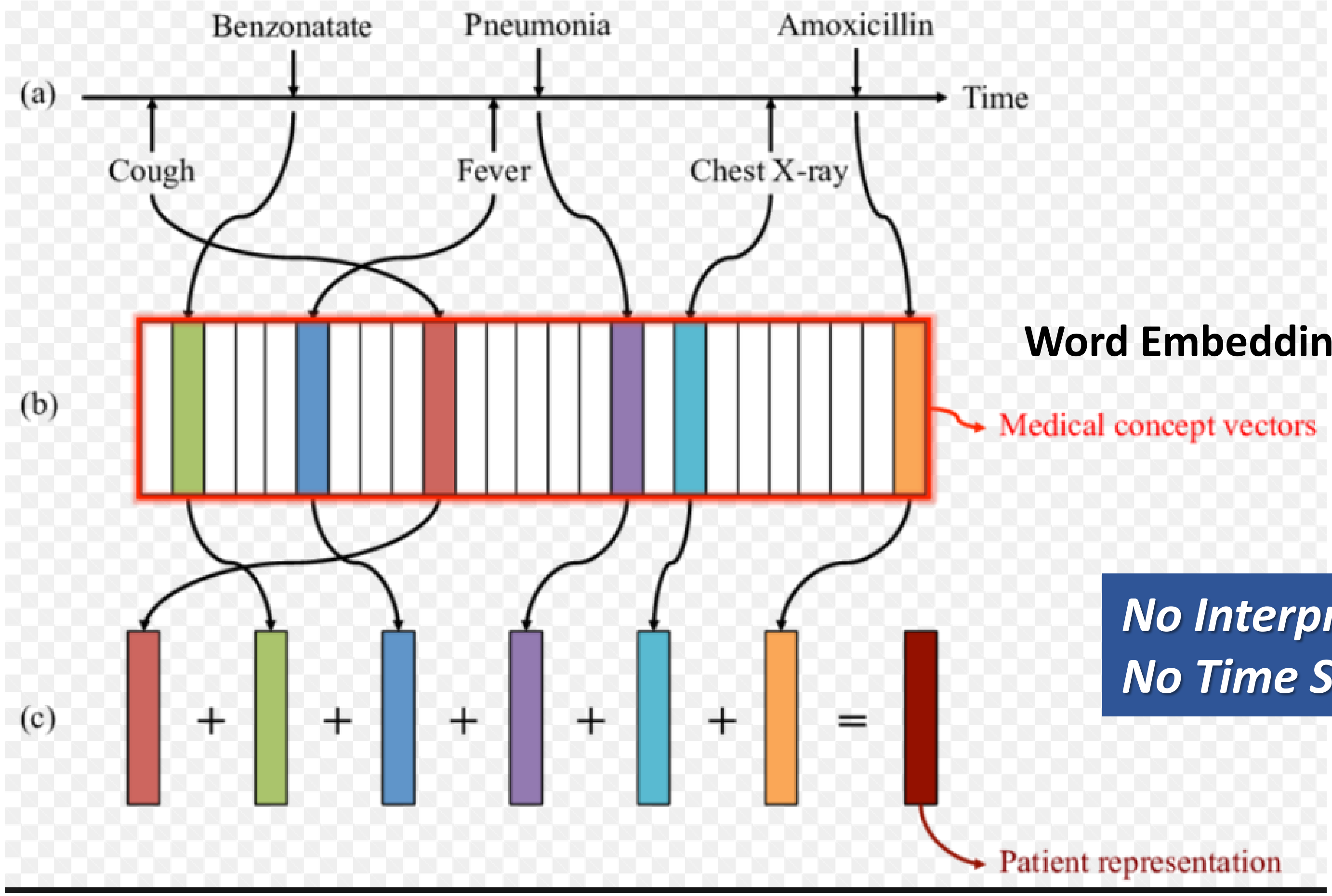


(b) Predicting neighboring medical concepts given *Fever*

(c) Predicting neighboring medical concepts given *Pneumonia*

https://arxiv.org/pdf/1602.03686.pdf

Concept Representation to Patient Presentation

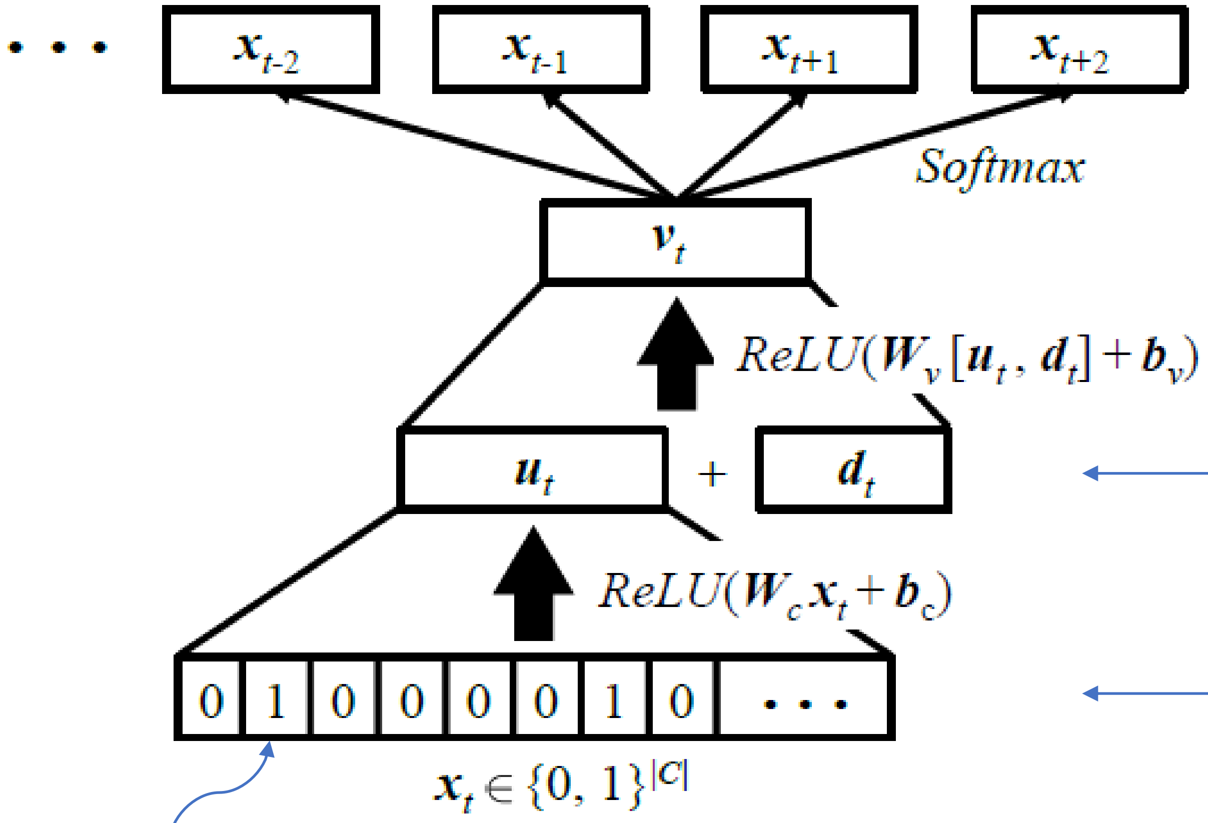


Word Embedding (NLP)

No Interpretation!
No Time Sequence!

Multi-Layer Representation Learning for Medical Concept

Mr.Choi names this architecture as Med2Vec!
Probably it is difficult to build a sequential model using medical concepts only. (lost of dups concepts)
Let's bring a 'visit' layer to the concept representation learning.



Predict pre and post visit data

Visit representation + Demographic data

Visit data containing medical concept

medical concept (diagnosed as gastritis)

AutoEncoder Patient Presentation – Deep Patient

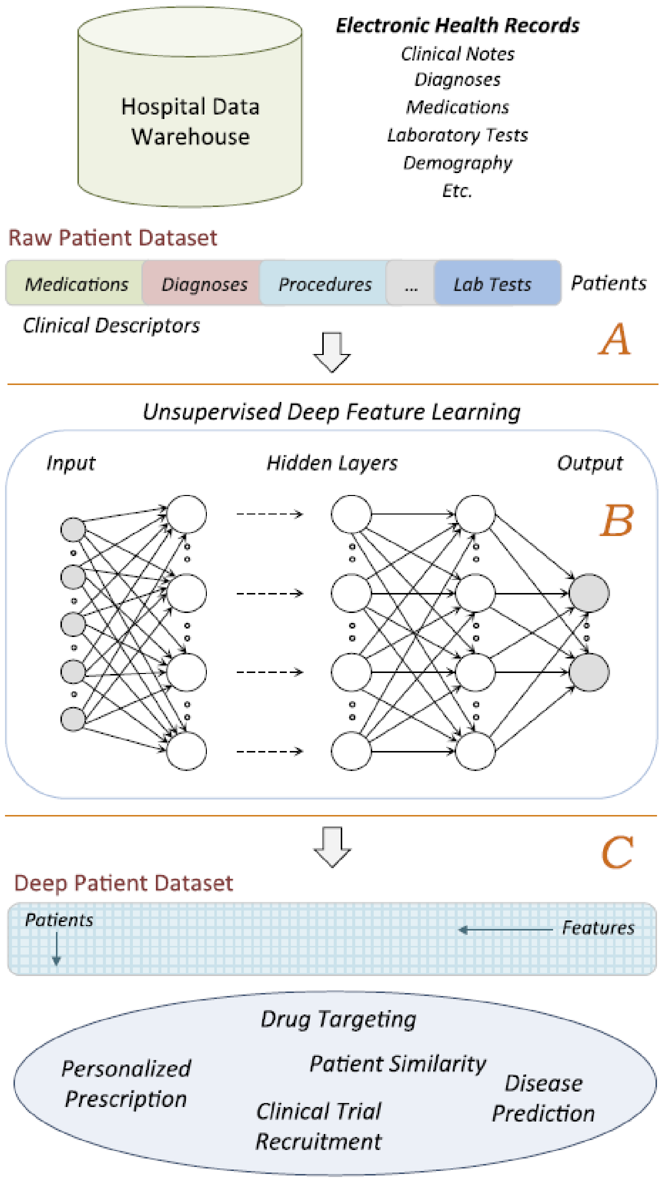
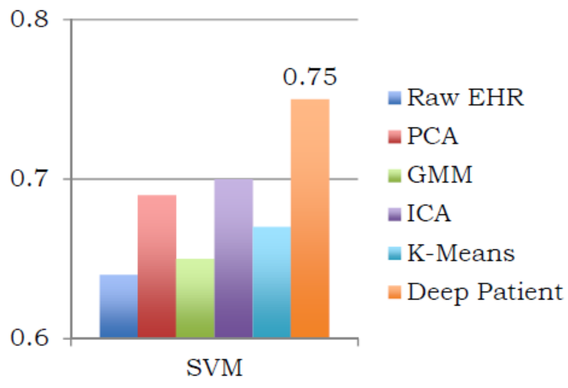


Figure 1. **Conceptual framework used to derive the deep patient representation through unsupervised deep learning of a large EHR data warehouse.** (A) Pre-processing stage to obtain raw patient representations from the EHRs. (B) The raw representations are modeled by the unsupervised deep architecture leading to a set of general and robust features. (C) The deep features are applied to the entire hospital database to derive patient representations that can be applied to a number of clinical tasks.

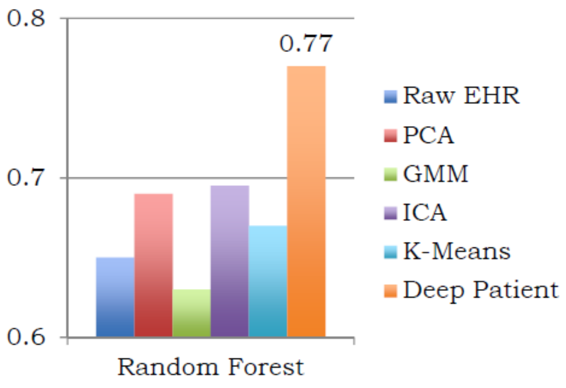
Stacked Denoising AutoEncoder
-> Good idea, but no interpretability and no temporal info !



Results in terms of the *Area Under the Receiver Operating Characteristic Curve (AUC-ROC)*

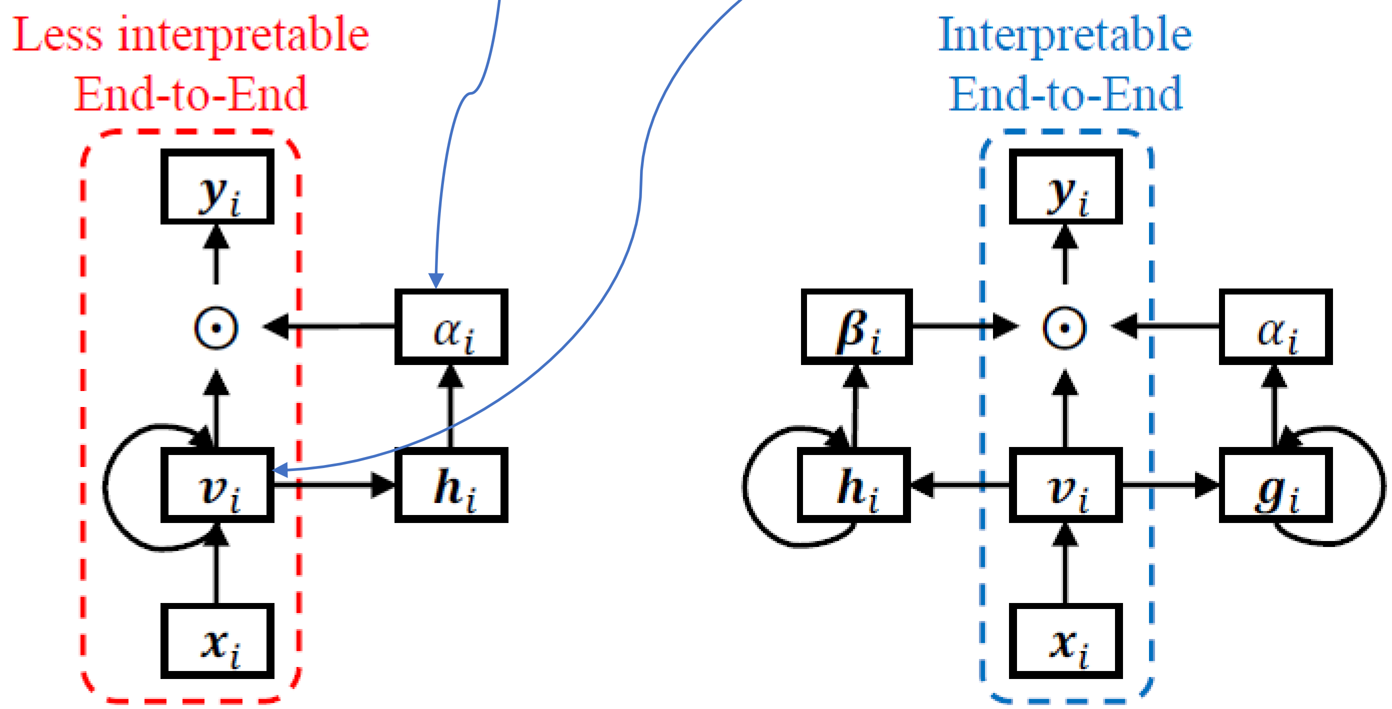
Deep Logistic Regression Network

AUC-ROC = 0.79



Retain – Interpretable and Predictive Model

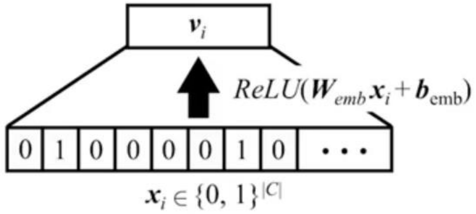
Let's have interpretability(Attention) and sequential information(RNN).

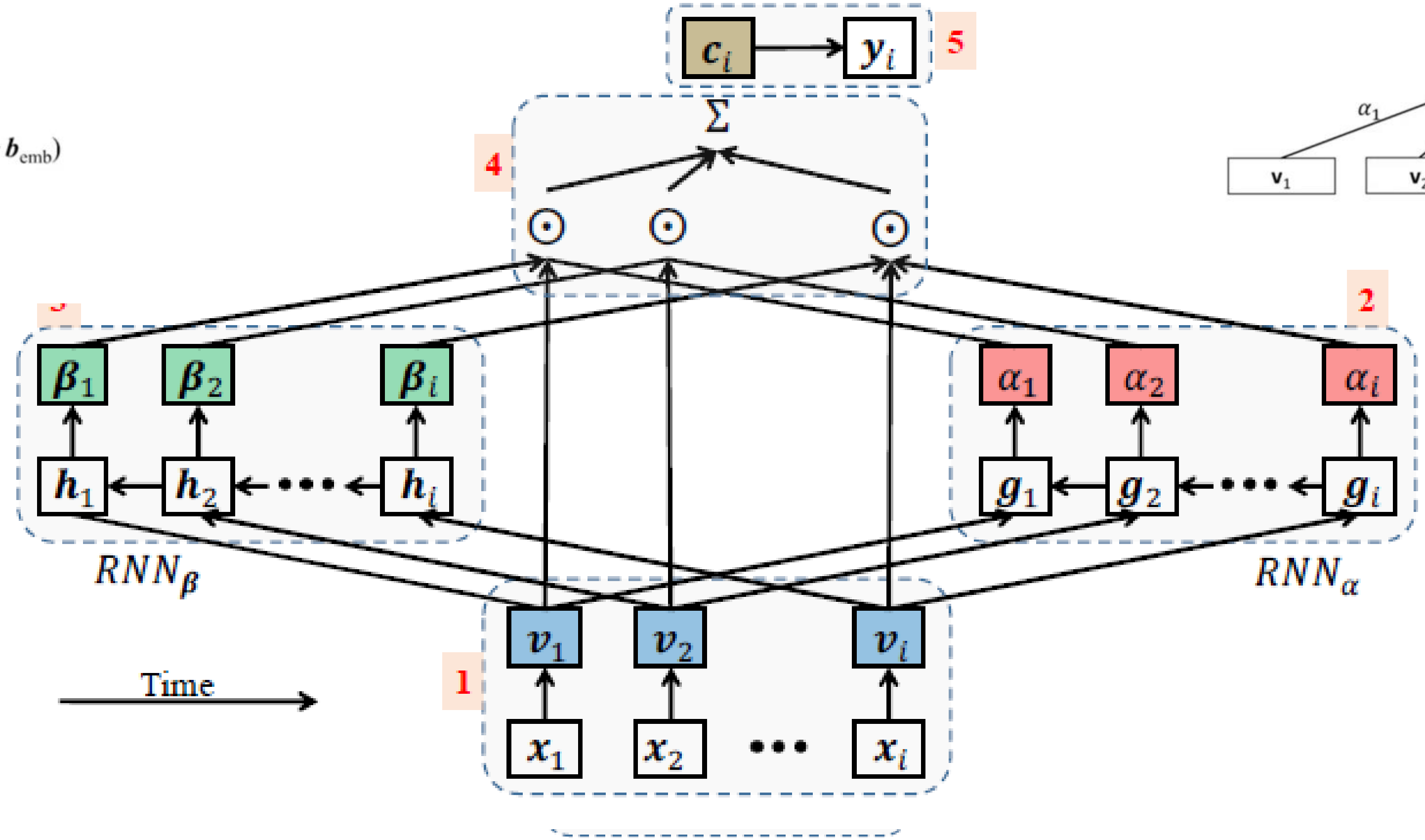


(a) Standard attention model (b) RETAIN model

Retain – Interpretable and Predictive Model

Let’s unfold the RNN model.





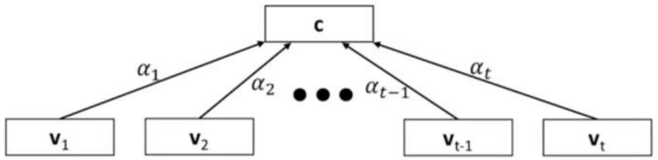


Figure 2: Unfolded view of RETAIN’s architecture: Given input sequence $\mathbf{x}_1, \ldots, \mathbf{x}_i$, we predict the label $\mathbf{y}_i$. **Step 1**: Embedding, **Step 2**: generating α values using RNN_α, **Step 3**: generating β values using RNN_β, **Step 4**: Generating the context vector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

Retain – Interpretable and Predictive Model

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

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,

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

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,

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

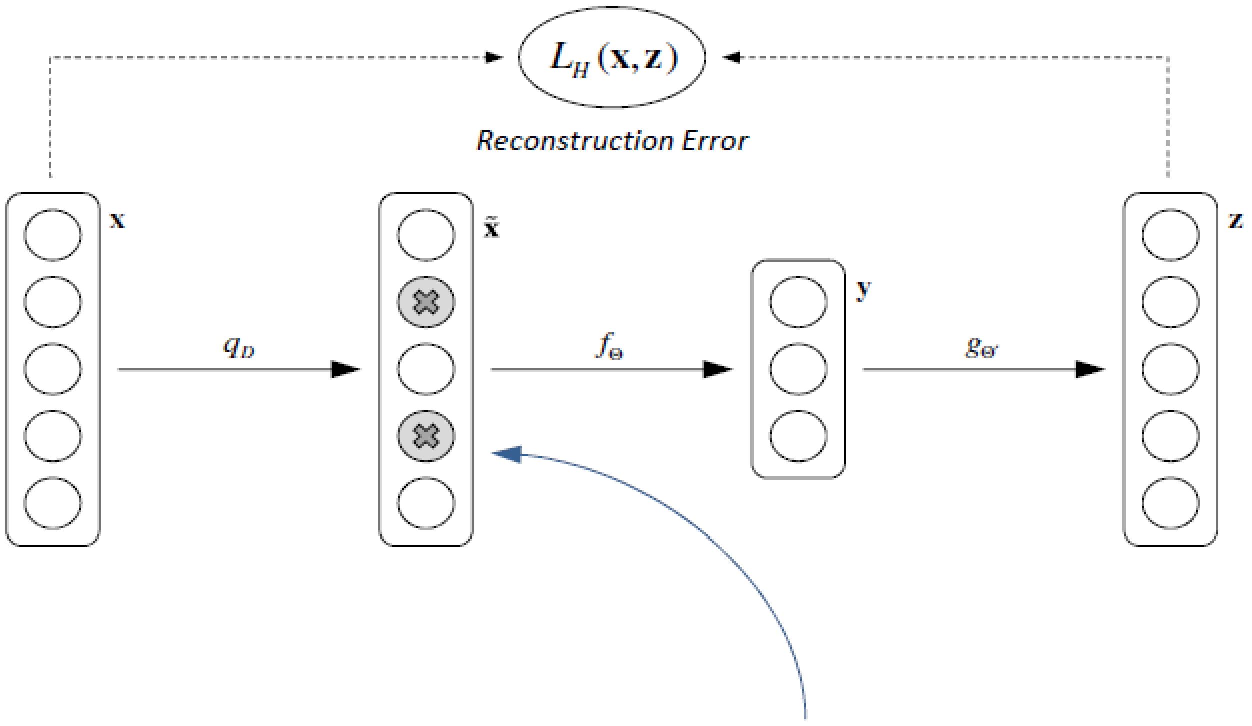
where $x_{j,k}$ is the k-th element of the input vector $\mathbf{x}_j$. Eq (4) can be completely deconstructed to the variables at each input $\mathbf{x}_1,\ldots,\mathbf{x}_i$, which allows for calculating the contribution ω of the k-th variable of the input $\mathbf{x}_j$ at time step $j \le i$, for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

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

APPENDIX

Stacked Denoising AutoEncoder

Denoising
Autoencoder



masking noise
corruption strategy

1. Corruption strategy
2. Encoder
3. Decoder
4. Minimize the difference between the original input and the reconstruction