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

Choi et al., arXiv, Feb (2017)

Group 5: Emma Marriott, Ankit Jain

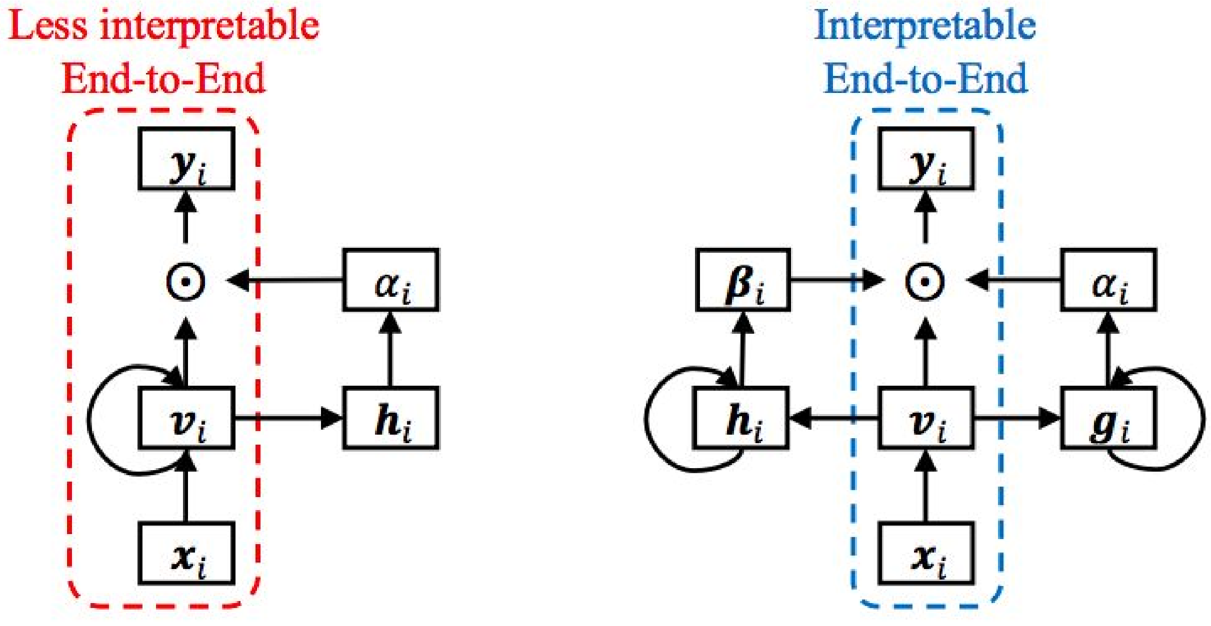
Outline

- Motivation
- Dataset
- Network Architecture
- Interpretability
- Implementation Details
- Experiments
- **Critical Review**

Motivation

- Most models have tradeoff between accuracy and interpretability, eg. RNN vs decision trees or clustering
- How to retain sequence level information and accuracy of RNNs but have parameters that can be understood?
- ***Attention-based*** models operate similar to a doctor - focus on important events in a sequence with a lot of data
- Have model behave the same way as doctor, working from most recent to oldest encounter

Solution



(a) Standard attention model

(b) RETAIN model

Overall Dataset

- EHRs from Sutter Health
- 14 million visits 263,000 patients over an 8 year period
- Data includes encounter records, medication orders, and procedure orders
- From encounter records and medication/procedure orders, extracted codes for diagnosis (ICD-9), medication (GPI) and procedures (CPT)
- Dimensionality reduction via code grouping
    - ICD-9 using Clinical Classifications Software 14k → 283
    - GPI using Generic Product Identifier Drug Group 91k → 96
    - CPT using Clinical Classifications Software 9k → 238

Training Data Set

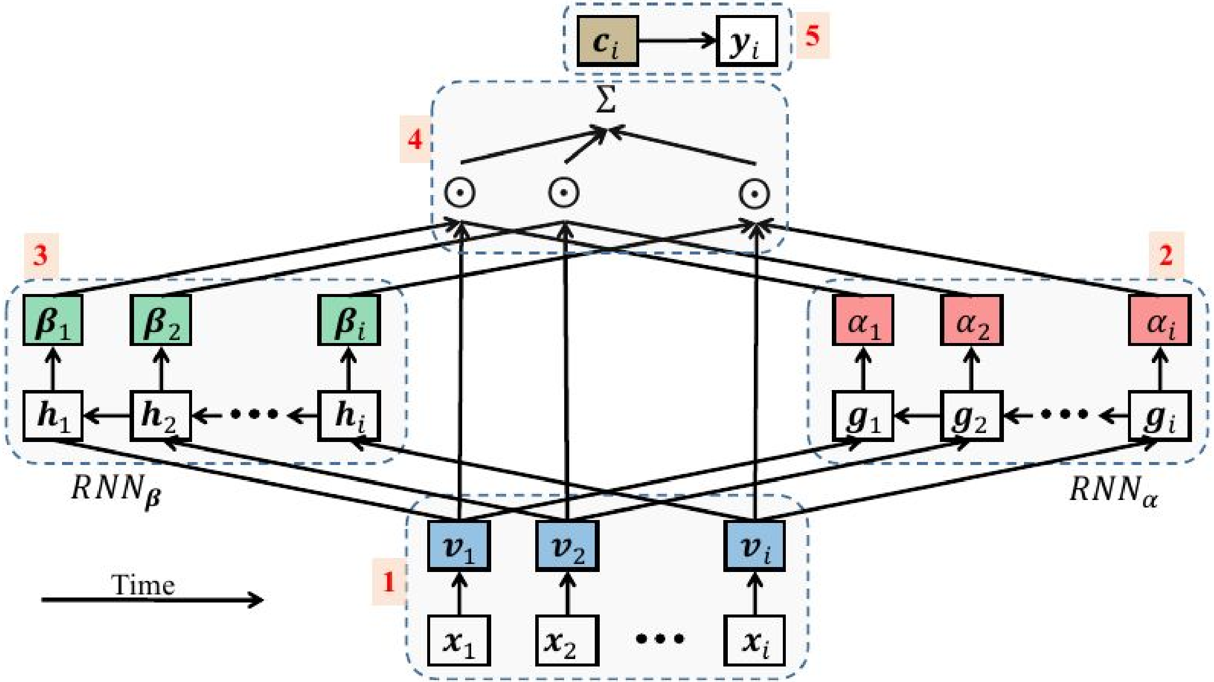
Positive Example Criteria (3,884 cases)

- 40 to 85 years of age at the time of HF diagnosis
- HF (heart failure) ICD-9 codes appeared in encounter diagnosis/medications for at least 3 distinct visits no further than a year apart

Negative Example Criteria (28,903 cases)

- 9 controls for every HF case
- Same sex, age, location as HF case
- 1st encounter within a year of HF patient's 1st encounter
- Did not meet heart failure diagnosis condition starting from time of diagnosis + 182 days after

Network Architecture



Steps:

1: Embedding

2: Visit-level attention

3: Variable-level attentions

4: Context vector

5: Making Prediction

Network Architecture: Details

$$\mathbf{v}_i = \mathbf{W}_{emb}\mathbf{x}_i$$

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

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

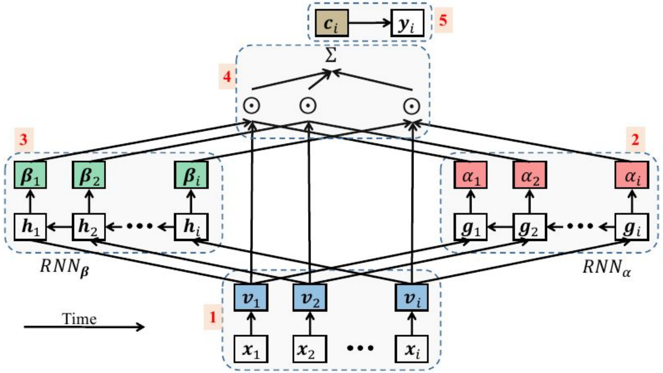
$$\alpha_1, \alpha_2, \ldots, \alpha_i = \mathrm{Softmax}(e_1, e_2, \ldots, e_i)$$

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

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

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

$$\widehat{\mathbf{y}}_i = \mathrm{Softmax}(\mathbf{W}\mathbf{c}_i + \mathbf{b})$$



Interpretability

- Contributing visits: largest α_i
- Contributing variables:

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

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

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

Implementation and Hyperparameters Tuning

Implementation: Adadelta with mini-batch of 100 patients. Intel Xeon E5, 256 GB RAM, two Nvidia Tesla K80.

$m, p, q = |\boldsymbol{v}|, |\boldsymbol{g}|, |\boldsymbol{h}|$

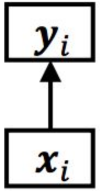
- m, p, q sampled randomly from (32, 64, 128, 200,256)
- L_2 regularization coefficient on embedding and context-vector weights sampled from (0.1, 0.01, 0.001, 0.0001)
- Dropouts on embedding weights and context-vector weights sampled from (0.0, 0.2, 0.4, 0.6, 0.8)

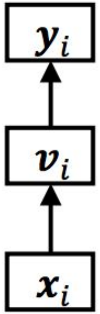
Tuned hyperparameters:

m, p, q = 128, L_2 coefficient = 0.0001, dropouts = 0.6

Experiments: Results

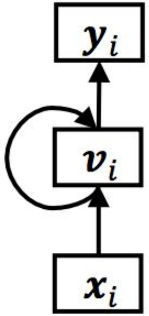
Logistic Regression

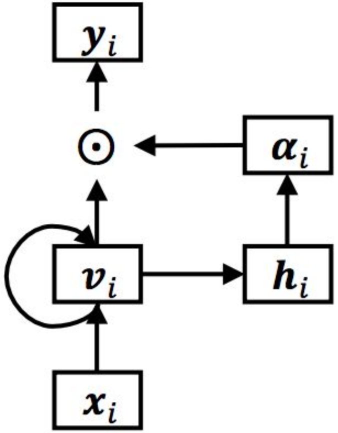




MLP

RNN





RNN + αM

RNN + αR

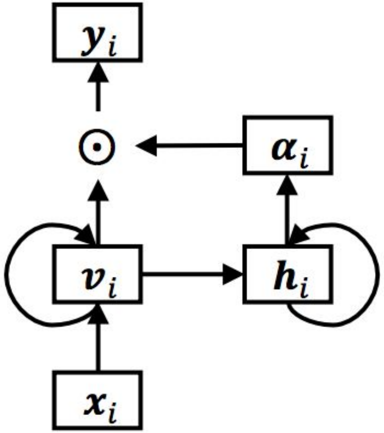
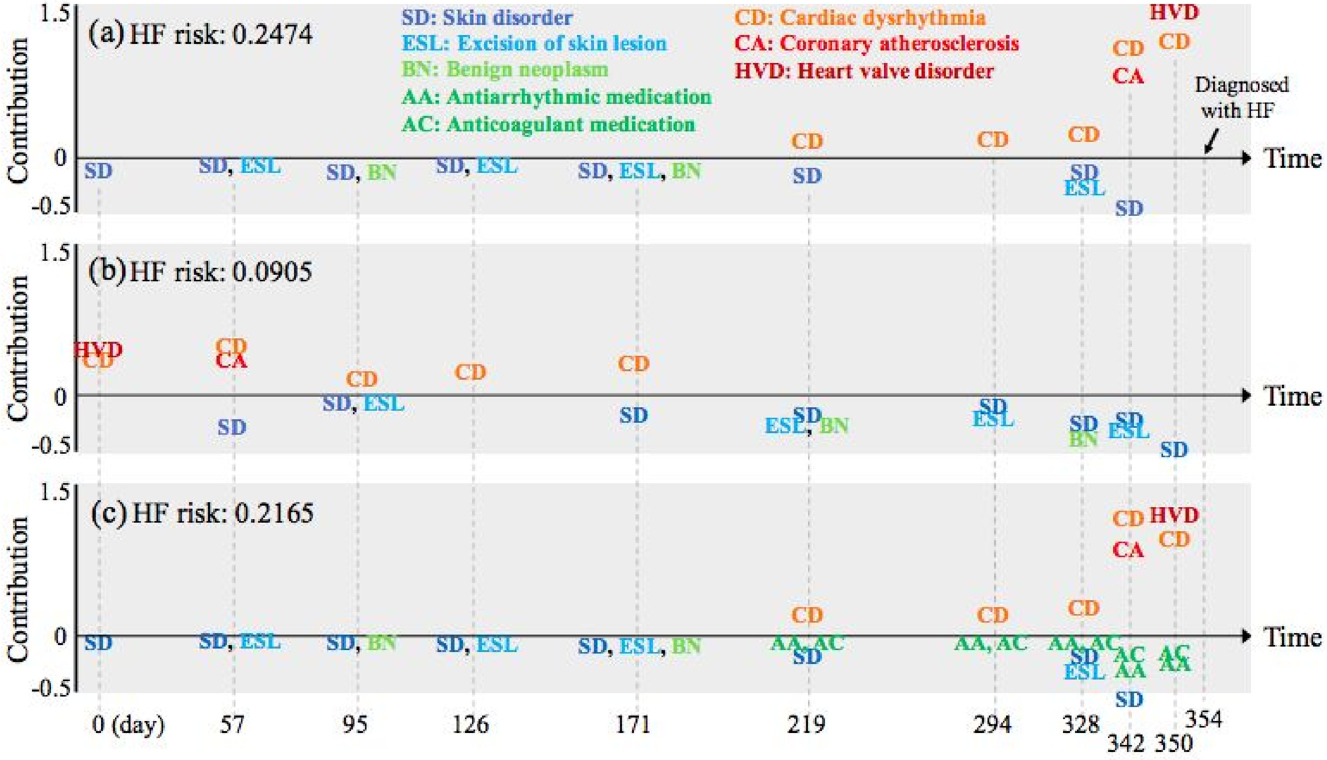


Table 2: Heart failure prediction performance of RETAIN and the baselines

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

Visualization



Critical Review

- Neat approach with RNN only for attention weights generation
- Used AUC to take care of class imbalance
- Extended the implementation for Encoder Sequence Modeling (ESM) and for including timestamps
- Logistic regression or MLP with aggregate features: not a fair comparison
- Selectively presented calculation time only for learning to diagnose case where prediction are made only at the end of time sequence
- 1:9 ratio for positive:negative is not realistic
- How about (MLP + RNNα) or (MLP + RNNβ)?
- Bi-directional RNNs
- Negative examples and model limitations?

Thanks You
Questions?