Proceedings of Machine Learning Research 116:93–111, 2020 Machine Learning for Health (ML4H) at NeurIPS 2019

Predicting utilization of healthcare services from individual disease trajectories using RNNs with multi-headed attention

Yogesh Kumar[1, 2] YOGESH.KUMAR@AALTO.FI
Henri Salo[2] HENRI.SALO@THL.FI
Tuomo Nieminen[2] TUOMO.NIEMINEN@THL.FI
Kristian Vepsäläinen[2] KRISTIAN.VEPSALAINEN@THL.FI
Sangita Kulathinal[2, 3] SANGITA.KULATHINAL@HELSINKI.FI
Pekka Marttinen[1, 2] PEKKA.MARTTINEN@AALTO.FI
[1] *Department of Computer Science, Aalto University, Finland*
[2] *Finnish Institute for Health and Welfare (THL), Finland*
[3] *Department of Statistics, University of Helsinki, Finland*

Editors: Adrian V. Dalca, Matthew B.A. McDermott, Emily Alsentzer, Samuel G. Finlayson, Michael Oberst, Fabian Falck, and Brett Beaulieu-Jones

Abstract

Healthcare resource allocation is an application that has been largely neglected by the machine learning community. We utilize the Electronic Health Records (EHR) of 1.4 million Finnish citizens, aged 65 and above, to develop a sequential deep learning model to predict utilization of healthcare services in the following year on individual level. Historical longitudinal EHR records from previous years, consisting of diagnosis codes, procedures, and patient demographics, are used sequentially as an input to a Recurrent Neural Networks (RNN). We improve the standard RNN regression pipeline for EHR code sequences by adding a Convolutional Embedding layer to address multiple codes recorded simultaneously, and Multi-headed attention. This reduces the number of epochs to converge by approximately 38% while improving the accuracy. We achieve approximately 10% improvement in R^2 score compared with state-of-the-art count-based baselines. Finally, we demonstrate the model's robustness to changes in healthcare practices over time, by showing that it retains it's ability to predict well into future years without any data available at the time of prediction, which is needed in practice to aid the allocation of healthcare resources.

1. Introduction

Electronic Health Records (EHR) have been used to solve a myriad of predictive problems using deep learning (LeCun et al., 2015; Goodfellow et al., 2016; Shickel et al., 2017; Rajkomar et al.), e.g., predicting the next diagnosis (Lipton et al., 2015; Choi et al., 2016b) and time duration to the next visit (Choi et al., 2016a; Harutyunyan et al., 2019) and learning interpretable patient representation for downstream tasks (Miotto et al., 2016; Zhang et al., 2018a; Choi et al., 2016c), etc. These models can help identify high-risk patients or recommend personalized treatments. An application domain that has received little attention in the machine learning literature is resource allocation, whereby a sponsor (e.g. a government) pays healthcare providers (or insurance plans as in the US) money based on individuals registered as customers with the provider. For fair and efficient resource allocation the sponsor needs a

risk adjustment model, which can predict the next year's healthcare cost for each individual, using data from previous years. In this work, we develop such a model, using the number of physical visits to a General Practitioner (GP) as a proxy for the cost.

Risk adjustment models are used in many countries, e.g., the USA, the Netherlands and Germany, to allocate healthcare resources (McGuire and van Kleef, 2018). Models used in practice are based on simple regression, with inputs such as demographic variables (age, gender), socio-economic variables and counts of previous year's diagnoses. State-of-the-art accuracy has been reached by random forests and ensembles (Rose, 2016; Shrestha et al., 2018; Breiman, 2001). Here, we develop models that use individual trajectories of disease diagnoses and treatments as input. Risk adjustment poses several constraints compared with typical machine learning applications (Ellis et al., 2018): 1) *Restrictions on features*: personal identifiers must be removed to protect privacy. Sensitive features e.g. race or income, may also be necessary to remove for fairness. Variables such as current year's spending, even if highly predictive for next year's spending, can't be used, to maintain incentives to control cost (otherwise providers could increase their revenue next year by treating more this year); 2) *Transformations of the dependent variable*: risk adjustment models often use 'top-coding' i.e. truncating the spending at some value. The motivation is that costs of heavy users of healthcare services may be compensated from a separate budget; 3) *Accommodating time lag between estimation and use of the model*. We formulate our model to incorporate restrictions 1) to 2), and demonstrate the model's ability to generalize across years without any data from future target years, as demanded by 3).

In addition to the constraints arising from the application, the sequential time series healthcare data poses some challenges that have not been fully addressed by previous works, and we suggest some enhancements to the architecture of existing RNN models. First, the observations come with irregular time intervals, and multiple diagnoses may be recorded at a single time point. To aid the vanilla Long Short Term Memory (LSTM) model in handling this, we added an embedding convolution layer which improves both the accuracy and training time compared to simply aggregating all the embeddings within a time step. Second, when using an RNN for regression, the output obtained from the RNN cell after the last time step is passed onto a stack of Fully Connected (FC) layers to make the prediction (Zhang et al., 2018b). With longer sequences, the problem of vanishing gradients (Hochreiter, 1998) could zero out the signals from input elements that show up early in the sequence. To tackle this, we added a multi-headed attention layer (Vaswani et al., 2017) that attends to the outputs from all the time steps of the RNN layer. These modifications help reduce the training time and improve the model accuracy and stability.

Our contributions can be summarized as follows:

- Introducing the problem of risk adjustment to the machine learning community, and presenting the first model for this application using state-of-the-art deep learning techniques and data on individual trajectories.
- Architectural enhancements, the embedding to handle multiple codes per visit and the multi-headed attention, to improve the accuracy and stability while reducing the training time.
- Empirical experiments with nationwide data on the elderly, which demonstrate that sequential deep learning models outperform all the count-based baselines, even with a

training set of 100,000 patients (roughly 10% of our dataset). This goes against the common misconception that deep learning models require very large datasets to show appreciable results.

2. Related Work

Recent works involving EHR data can be broadly classified into those that view the data sequentially and those that aggregate the counts of different medical codes. Even though deep learning methods have not been applied to the problem of predicting patient visits, they have been applied to other problems using the same data format, i.e., treating EHR data as sequences to make predictions. Lipton et al. (2015) used sequential real-valued measurements of 13 different vital measurements to predict one of 128 diagnoses using LSTM, while Choi et al. (2016a) applied a 2-layered Gated Recurrent Unit (GRU) model on diagnosis, procedure and medication sequence to predict the diagnosis for the next visit.

Attention based models have improved the state-of-the-art in tasks from various domains (Cui et al., 2016; Ba et al., 2014; Bahdanau et al., 2014; Chorowski et al., 2015; Hermann et al., 2015; Xu et al., 2015; Paulus et al., 2017). This is owed to the fact that attention mechanisms provide the neural network the ability to focus on a small subset of the input to make its prediction. Naturally, attention mechanisms have been utilized in EHR as well, to achieve impressive results; Choi et al. (2016b) added attention mechanism to the RNN model in Choi et al. (2016a) to improve interpretability, Ma et al. (2017) use a Bi-directional RNN model Schuster and Paliwal (1997) with attention mechanism, Song et al. (2018) uses the encoder from an RNN-less, Transformer model (Vaswani et al., 2017) evaluated on a multi-task learning benchmark established on the MIMIC-III (Johnson et al., 2016) by Harutyunyan et al. (2019).

3. Cohort

The data for this study was sourced from the Register of Primary Health Care Visits (AvoHilmo, avo) of Finnish Institute for Health and Welfare, THL and consists of out-patient visit information for every Finnish citizen aged 65 or above. This pseudonymized dataset consists of demographic information such as age and gender, as well as the diagnosis and procedure codes assigned for each patient visit collected between the years $2012 - 2018$. Unlike the in-patient hospitalization records, the term "patient visit" applies more broadly here and could also include other (sometimes more frequent) forms of contact with the health professionals, e.g., periodic health checkups, over-the-phone consultations, personal home care visits by a nurse. As a result, the distribution for the input number of patient visits is highly skewed with a very long tail.

The raw tabular data consisting of diagnosis and procedure codes for each patient visit was transformed to be sequential by grouping the rows based on both PatientID and VisitID (which are unique numeric codes assigned to each pseudonymized patient and hospital visit respectively). The intra-visit diagnosis codes appear in the same order with which the GP recorded them into the register. Though not strictly enforced, GPs are advised to enter the primary diagnosis as the first entry. In order to transform the processed data into a form that would be similar to natural language, the groupings based on VisitID are space-separated

Table 1: Basic statistics of EHR dataset across all years (2012-2018)

# of unique patients	1,396,766	Max # of codes per visit	47
# of visits	271,103,617	# of unique ICD-10 codes	12,082
Avg. # of visits per patient	182.733	# of unique ICPC-2 codes	1,360
Avg. # of codes per visit	2.783	# of unique procedure codes	495

and those based on PatientID are semicolon-separated. The resulting dataset consisted of 1,396,766 patient sequences across the years $2012 - 2018$ with an average of 182.733 visits per patient. Table 1 lists some basic statistics from the data.

Further, our dataset consists of two types of diagnosis codes - International Classification of Diseases (ICD-10) and International Classification for Primary Care (ICPC-2). ICPC-2 codes are more common in AvoHilmo and are usually assigned by a non-doctor. We designed all our models to be agnostic to the codes, so in all our experiments, we do not specifically distinguish between the diagnosis and procedure codes. The code syntax for ICD-10 and ICPC-2 overlap considerably, so in order to distinguish between them, we prefixed the ICPC-2 codes with "IP-".

4. Methods

In this section we take a closer look at the data representation and then discuss the details of the analyses and model architecture used for prediction.

4.1. Baseline models

This dataset is being modelled with sequential models for the first time and hence, we devised strong benchmarks to evaluate the sequential models against two non-sequential, count-based baseline models - Lasso Regression and Gradient Boosted Trees (GBDT) (Chen and Guestrin, 2016). Lasso Regression serves as a highly interpretable linear model benchmark and GBDT is our strong boosted ensemble benchmark. Tree based ensembles, specifically Random Forests (Breiman, 2001), have been the state-of-the-art in problems with similar objectives (Rose, 2016). GBDT trains an ensemble of weak decision trees in a forward fashion such that each tree is trained on the residual error of the preceding tree. In our implementation, we have used the LightGBM variant of GBDT which grows each tree leaf-wise and is thus faster and more accurate (Ke et al., 2017).

For these models, we use the TF-IDF encoding to convert the categorical tokens to numerical form. TF-IDF is a form of bag-of-words (BoW) representation where a sequence from a dataset containing $|V|$ unique tokens is transformed into an array $x \in \Re^{|V|}$. Each index i in x corresponds to a token in V and each element x_i is the count of the i^{th} token in the sequence. One disadvantage of using BoW directly is that the tokens that occur frequently in the data will tend to dominate the sequence vectors, so the TF-IDF encoding divides the frequency of a token in the sequence by the inverse document frequency, which is computed as $idf(t, D) = log\frac{N}{|d \in D : t \in d|}$, where D is the set of all the patient sequences, $N = |D|$ and $|d \in D : t \in d|$ is the number of patient sequences where the token t occurs.

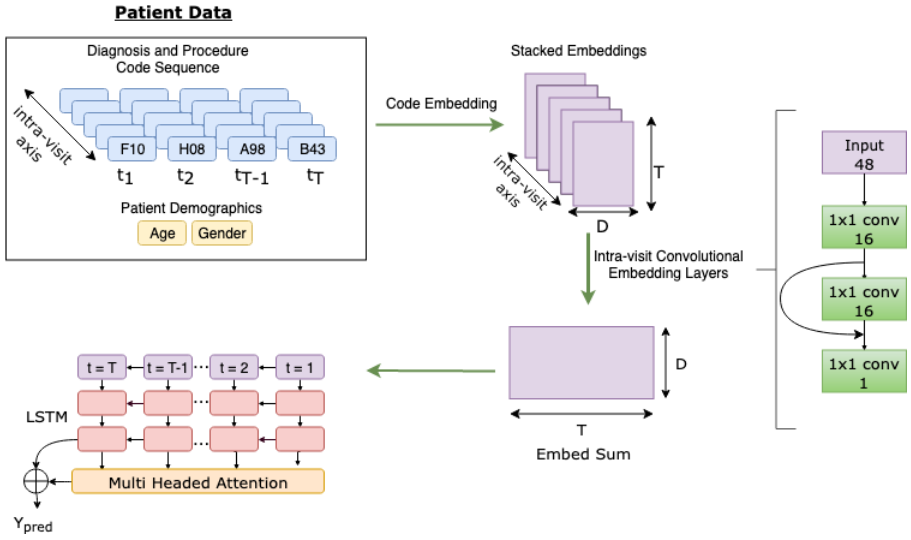

Figure 1: Deep sequential model formulation of individual health records. *Left* Overall architecture of the Long Short Term Memory (LSTM) model (Fully Connected layers have been omitted for clarity). D is the code embedding dimension and T is the maximum number of visits; *Right* Convolutional embedding. The intra-visit dimension is 48, which corresponds to the maximum number of codes in a single visit, and each green square denotes a convolutional layer.

Besides comparing the predictive performance based on R^2 score (Eq. 4) and Mean Absolute Error (MAE), we also test their ability to retain the predictive power for predicting healthcare services usage for one and two years. This study on model generalization to future years is further discussed in Section 5.5.

4.2. Sequential deep learning model

Even though the backbone of our sequential deep learning model is a stack of RNN layers, there were other modules that served as the scaffolding for transforming the raw sequences into the form suitable for RNNs.

4.2.1. EHR data representation

The EHR data when viewed sequentially, can be conceptualized as a multi-variate time series. However, the time and frequency of visits varies vastly among patients, so fixing a single time frame of reference for all the patients would result in highly sparse sequences. We instead use the tuple (t, X_t) as a data-point, where t is the timestamp and X_t contains the observations (diagnosis and procedure codes) for the t^{th} visit. Further, we also encode the time difference (in days) between each visit by concatenating the Δt with X_t. With this representation, the length of the input sequence will vary for different patients, to handle this we pad short sequences with zeros at the end.

The input code sequence X, consisting of the diagnosis and procedure codes $c \in V$, is first one-hot-encoded (OHE) to $\widehat{c} \in \Re^{|V|}$, where $\widehat{c}_p = 1$ for the p^{th} code in the vocabulary V

and is zero elsewhere. Further, multiple codes associated with each time step, t, were stacked along the intra-visit axis, j. The resulting vector X_t for each time step is of size $V \times J$, where J is the length of the intra-visit axis. Other numerical features such as the patient age, gender and basic statistics such as "average # of days between visits" were concatenated to the visit code vector at each time step, X_t, before being passed onto the machine learning models.

4.2.2. Code Embedding layer

We borrow the idea of embedding layer from NLP literature to encode categorical tokens into dense vectors. The OHE representation suffers from two problems - 1) the input dimension can become very large if the token vocabulary is high, 2) the dimensions of the OHE vectors are orthogonal to each other and thus, we are not providing the model with any inter-dimensional dependency signal.

In the embedding layer, the one-hot encoded vector for each code is multiplied with an embedding matrix of dimension $|V| \times D$, thus resulting in a D-dimensional dense vector. Usually, we set $D \ll |V|$, forcing the model to learn inter-dimensional dependencies and encode them into the dense vectors. The weights of the embedding matrix are learned by the neural network during training.

4.2.3. Intra-visit embedding aggregator

Each visit could be accompanied by more than one diagnosis or procedure code. Hence each time-step t, can have several codes c_j associated with it. The size of this dimension could go as high as 47 in our dataset (Table 1). Thus, the embeddings obtained from the code embedding layer need to be further aggregated along the intra-visit axis, j. Previous work using sequential EHR data handled this problem by aggregating along this dimension by simply summing across j (Choi et al., 2016a). The convolutional embedding block (Figure 1) consists of a sequence of 1×1 convolution operations. The purpose of 1×1 convolutions is to modify the size of the input tensor in the intra-visit axis while preserving the dimensions along the other directions. The convolutional block consists of three 1×1 convolutions, we apply batch normalization and PReLU activation to the tensor between each convolution layer and add skip-connections (He et al., 2016) between the first and third layers. Treating the intra-visit codes as the channels of the convolutional layers helps leverage the information in the ordering of these codes (Section 3). By stacking three convolutional layers separated by non-linear activation, we have also increased the representative capacity of the model, thus resulting in improved accuracy and faster convergence (Table 3).

4.2.4. LSTM layer

In recent years, RNNs with Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) cells have demonstrated state-of-the-art performance in tasks with sequential data such as machine translation, speech recognition, time series prediction etc. RNNs are designed to handle sequential data by sharing the parameter of the RNN cell with all the time-steps. For completeness, we include the equations defining the various gates used in LSTM below,

$$
\begin{aligned}
i_t &= \sigma\big(X_t U^i + h_{t-1} W^i\big) & C_t &= \sigma\big(f_t * C_{t-1} + i_t * \tilde{C}_t\big) & o_t &= \sigma\big(X_t U^o + h_{t-1} W^o\big) \\
f_t &= \sigma\big(X_t U^f + h_{t-1} W^f\big) & \tilde{C}_t &= \tanh\big(X_t U^g + h_{t-1} W^g\big) & h_t &= \tanh(C_t) * o_t
\end{aligned} \tag{1}
$$

In order to capture the long-term dependencies, the LSTM cell adds internal gating mechanism in the form of input, forget and output gates (i, f & o respectively in Eq. 1). X_t and h_t are the input and the RNN hidden value at time step t, respectively. As shown in Figure 1, the intra-visit aggregated embeddings from the convolutional blocks are fed sequentially to the two LSTM layers.

4.2.5. Multi-headed attention layer

Self-attention, also known as intra-attention, is designed to capture dependencies between the tokens belonging to the same sequence (Song et al., 2018). Self attention has been used in many NLP tasks such as machine translation (Vaswani et al., 2017), machine comprehension (Cui et al., 2016) and language models (Devlin et al., 2018; Radford et al., 2019). Multi-headed attention consists of multiple copies of these self-attentive layers, all receiving the same input.

The self-attention module takes as input a key-value pair (k_t, v_t) and a query q_t to compute the output o_t for each time step t. The self-attention coefficient is computed by first stacking together the q_t, k_t and v_t values along the time-axis to obtain Q, K and V, respectively and using the following equation,

$$
Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{T}}) \times V. \tag{2}
$$

Here, T is the length of the sequence. Since we care about learning dependencies between each time step, in our case, the key, value and query are all copies of the input sequence, X. For multi-headed self attention, these attention weights are calculated multiple times in parallel and the resulting vectors are concatenated together. Finally these concatenated vectors are projected onto the output space using a fully-connected layer. In the following equation, W^o, W_i^Q, W_i^K & W_i^V are the parameters learned by the model (Vaswani et al., 2017).

$$
\begin{aligned}
MultiHead(Q, K, V) = Concat(head_0, head_1, \ldots, head_h) W^o, \text{where} \\
head_i = Attention(QW_i^Q, K_i^K, V_i^V)
\end{aligned} \tag{3}
$$

In our model, this sequence of attention coefficients is aggregated across the time-steps using average or max pooling before being combined with the output from the last hidden state of the LSTM, using element-wise sum, and finally passed onto two fully-connected layers and the output layer.

5. Results

5.1. Experiment Setup

Given the code sequences $x_1, x_2, \ldots, x_T$ and patient features x_{num} for years $r_{t-i}, \ldots, r_{t-1}, r_t$, our goal is to predict the number of physical health center visits for the year r_{t+1}. A

Table 2: Test R^2 score for year 2016 (N = # of training examples)

Training Year(s)		Lasso	LightGBM	LSTM$_{mh_attn}$
2015	$N = 10K$	0.0381	**0.1918**	0.1551
	$N = 100K$	0.0372	0.2604	**0.2668**
	$N = 500K$	0.0367	0.2854	**0.2974**
2014, 2015	$N = 10K$	0.0744	**0.2167**	0.1954
	$N = 100K$	0.0750	0.2839	**0.2886**
	$N = 500K$	0.0752	0.3094	**0.3259**
2013, 2014, 2015	$N = 10K$	0.0925	**0.2416**	0.2080
	$N = 100K$	0.0918	0.2974	**0.3129**
	$N = 500K$	0.0913	0.3248	**0.3436**

validated and reliable model for this objective can help the government and healthcare providers allocate funds for the future health services usage. Among all the different ways a patient can contact the healthcare provider recorded in our data, we only predict the number of physical visits to the healthcare center. Specifically, we counted the visits from the register that were either classified as "outpatient", "visiting the hospital" or "customer visit to the health center reception", as physical visits. We chose this objective because physical visits demand higher amount of resources compared to, say, phone consultations. However, as input to the model, we still include all kinds of contact with the healthcare provider. As discussed in Section 3, the visits count distribution is highly skewed with a long tail. Since we are optimizing MSE which penalizes outliers heavily, it is common practice to transform the counts to log-space (Choi et al., 2016a) or add a "top-code" to the visits counts (Ellis et al., 2018). We chose to do the latter transformation, truncating the visits at 25, meaning all patients who are likely to visit the hospital more than twice a month are grouped together.

The RNN model was trained for 30 epochs with early-stopping after 8 epochs if there is no improvement on validation MSE loss. We also used Dropout (Srivastava et al., 2014) with $p = 0.3$ and weight-decay with coefficient 0.0001 for regularization. We used batch normalization (Ioffe and Szegedy, 2015) between the RNN and FC layers and layer normalization (Ba et al., 2016) between all other layers. The hidden size of the LSTM cell was set at 800, increasing the size further improved the performance, but drastically increased the training time as well. The multi-headed attention layer used 8 attention heads. We did not use pretrained embeddings for the embedding layer, even though it has been shown to improve the predictive performance (Choi et al., 2016a), since we wanted to keep the computation cost as low as possible. As a result, the entire training process can be completed in under 10 hours using a single GPU. We used the Adam optimizer (Kingma and Ba, 2014) for performing gradient descent with a batch size of 32 and learning rate $3e-4$, decayed step-wise every 5 epochs by a factor of 0.5. All RNN models were built using PyTorch (Paszke et al., 2017) and trained on a machine with a single NVIDIA Tesla V100 GPU. The codebase for the paper is available at https://github.com/aalto-ml4h/pummel-regression.

Table 3: Comparing various RNN architectures (two years history on 200K patients)

Model	R^2	Mean Absolute Error (MAE)	Runtime (mins/epoch)	Epochs to converge
$LSTM_{simple}$	0.3093	2.7333	8.8[1]	29
$LSTM_{conv\_emb}$	0.3115	2.7211	11.5	21
$LSTM_{simple\_attn}$	0.3033	2.7159	11.6	20
$LSTM_{mh\_attn}$	0.3122	2.7307	12.5	18

We evaluated all the models using MAE and Coefficient of Determination (R^2) metric. R^2 compares the MSE of the predictive model with the MSE obtained by predicting each test example, y_i, with the mean, $\bar{y}$, and is defined as

$$R^2 = 1 - \frac{\sum_i (y_i - \widehat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}, \tag{4}$$

where $\hat{y}_i$ is the predicted value.

5.2. Effect of the amount of training data

Collecting and processing medical data is expensive, their availability is scarce, and it requires thorough privacy protection through de-identification before analysis. Thus, we were also interested in quantifying the amount of data required to achieve satisfactory predictive performance. To achieve this, we conducted experiments varying the number of patients available for training and the number of past years given as input. We first randomly sampled 103,548 patients for test and 83,070 patients for validation set, while the training set was varied between $10K$, $100K$ and $500K$ patients. Each model was trained to predict the number of visits for the year 2016 and was optimized to minimize the Mean Squared Error (MSE). The validation set was used to pick the best hyperparameters for the model before the final models were evaluated on the test set.

The results on R^2 metric are presented in Table 2 and the MAE results are in the Supplement. The statistical significance of the difference between LSTM and LightGBM predictions was assessed by computing the p-value using Wilcoxon signed-rank test. The results were consistently $\ll 1e{-}10$, indicating the differences are highly statistically significant (due to the very large test set). To assess if there is variability between different training runs, we retrained our model a second time. The results in the supplement show that the prediction accuracy is almost exactly the same in the two runs. Finally, we note that the R^2 values in Table 2 are in the range expected based on results published for risk adjustment models. For example, Rose (2016) obtained $R^2 = 0.26$ for random forests and $R^2 = 0.27$ for an ensemble. However, we emphasize that these results are not directly comparable: we predicted the number of visits whereas Rose (2016) predicted the actual spending, different data sets with different covariates were used, and Rose (2016) did not treat the data as a time series.

1. The run time for tensor summation across the intra-visit axis dropped from 28 min to 9 min per epoch after a PyTorch version upgrade https://github.com/pytorch/pytorch/issues/18807. Because absolute run time in minutes strongly depends on the code implementation, we instead focus our discussion on the rate of convergence.

Table 4: Results from training across all years (2012 – 2017) to predict for 2018

Model	R^2	Mean Absolute Error (MAE)	Spearman Corr.
Lasso	0.0996	2.9548	0.3007
LightGBM	0.3307	2.3657	0.5902
LSTM_{BIG}	0.3656	2.2500	0.6056

5.3. Comparing RNN architectures

We evaluated the two modifications to the simple RNN architecture as discussed in Section 1. For these experiments, we set the number of training examples at 200,000 and the input years as 2014 – 2015 to predict for the year 2016. Other than that, the experiment setup and the test set was fixed to be the same as that in Section 5.1.

The results from these experiments are presented in Table 3. The convolutional embedding block to aggregate over the intra-visit axis improved the model's accuracy and decreased the convergence time by approximately 38% compared to $LSTM_{simple}$, where we just sum across the intra-visit axis. Further, the attention layer $LSTM_{mh\_attn}$ slightly increases the accuracy and also led to faster convergence than $LSTM_{conv\_emb}$. We also benchmark the multi-headed attention network with a simpler attention model, structured self attention (Lin et al., 2017), used with RNNs for text classification.

The idea to use an attention layer that has access to all the hidden states of the LSTM was motivated from the promising results we obtained by using Bidirectional RNNs (Schuster and Paliwal, 1997) in our initial experiments. This highlighted the importance of the outputs from the early hidden states for the model's predictive accuracy. However, since Bidirectional RNNs require considerably more computational resources and time to achieve acceptable performance, we decided on using the attention layers instead.

5.4. Training across all the years

We also ran experiments with input data spanning across all the years between 2012 – 2017 to predict patient visits for the year 2018. These experiments were run on all patient data (as reported in Table 1), with 118,969 and 94,854 patients heldout for test and validation sets respectively. The results (shown in Table 4) clearly demonstrate the superior results obtained with the sequential neural networks when we train using very long patient trajectories.

5.5. Model generalization to future years

In practice there is a time lag between training and usage of a risk adjustment model (Ellis et al., 2018), and it is necessary that the model can generalize to future years without training data. To explore our model's robustness in this respect, we first trained a model using years 2014 – 2015 as covariates to predict 2016 visits and then used this model to predict the visits in years 2017 and 2018 (using data from 2015 – 2016 and 2016 – 2017 as covariates, respectively). Intuitively, we expect the sequential models to better capture the features that get carried over time than the count based models. We see that this is the case in

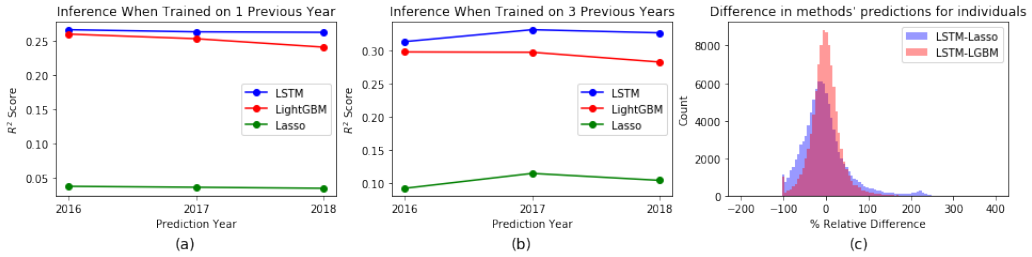

Figure 2: *Left, Center*: R^2 score when inferred on future years (all models were trained with $100K$ examples to predict 2016, no data from 2017 or 2018 was used in training); *Right*: Histogram showing the % relative change in predicted visits if we switch from Lasso or LightGBM to an LSTM model

Figure 2, which shows the R^2 score for all models from years 2016, 2017 and 2018. Somewhat surprisingly, for Lasso and LSTM the R^2 is even slightly higher in 2017 and 2018 (but not for LGBM), which likely reflects random variation in the data sets between different years.

6. Discussion

We applied state-of-the-art deep learning models to the important problem of predicting patient healthcare service usage, which has received little attention in the machine learning community. Except for the smallest data set size, the sequential deep learning model systematically outperformed the linear baseline (Lasso) and a strong ensemble baseline (LightGBM) with varying training set sizes and lengths of medical histories. Introducing the stack of 1×1 convolutional layers to summarize multiple codes in a single visit brought about significant improvements in both accuracy and training time, when compared to a simple aggregation across codes. Furthermore, a slight further improvement was obtained by using the multi-headed attention. Nevertheless, alternative ways to focus on different parts of very long sequences of medical events is a topic that still warrants further investigation in the future.

We conclude by discussing possible practical implications of using our model for resource allocation. Recall that in risk adjustment, payments to healthcare providers are based on the predicted costs for individuals registered as customers with the provider. To study the effects of using LSTM vs. the baselines, we compared the relative change in the predicted value (and hence hypothetical resource allocation) for each individual in our data set (Figure 2, *right*). We see that even if the increase in R^2 score is approximately 10%, on individual level the relative changes in predictions may be much greater. Studying such effects on different population subgroups, as well as validating the models according to various other criteria besides prediction accuracy, published in health economics literature (Layton et al., 2018), are important future topics.

Acknowledgement

This work was funded by the Academy of Finland (grants no. 286607 and 294015 to PM). We acknowledge the computational resources provided by Aalto Science-IT project.

References

Register of primary health care visits - THL. URL http://thl.fi/en/web/thlfi-en/statistics/information-on-statistics/register-descriptions/register-of-primary-health-care-visits.

Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F Stewart, and Jimeng Sun. Doctor ai: Predicting clinical events via recurrent neural networks. In *Machine Learning for Healthcare Conference*, pages 301–318, 2016a.

Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. RETAIN: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems*, pages 3504–3512, 2016b.

Edward Choi, Andy Schuetz, Walter F Stewart, and Jimeng Sun. Medical concept representation learning from electronic health records and its application on heart failure prediction. *arXiv preprint arXiv:1602.03686*, 2016c.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems*, pages 577–585, 2015.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423*, 2016.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Randall P Ellis, Bruno Martins, and Sherri Rose. Risk adjustment for health plan payment. In *Risk Adjustment, Risk Sharing and Premium Regulation in Health Insurance Markets*, pages 55–104. Elsevier, 2018.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015.

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

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

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, 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*, 3:160035, 2016.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Timothy J Layton, Randall P Ellis, Thomas G McGuire, and Richard C van Kleef. Evaluating the performance of health plan payment systems. In *Risk Adjustment, Risk Sharing and Premium Regulation in Health Insurance Markets*, pages 133–167. Elsevier, 2018.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzel. Learning to diagnose with lstm recurrent neural networks. *arXiv preprint arXiv:1511.03677*, 2015.

Fenglong Ma, Radha Chitta, Jing Zhou, Quanzeng You, Tong Sun, and Jing Gao. Dipole: Diagnosis prediction in healthcare via attention-based bidirectional recurrent neural networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1903–1911. ACM, 2017.

Thomas G McGuire and Richard C van Kleef. Regulated competition in health insurance markets: Paradigms and ongoing issues. In *Risk Adjustment, Risk Sharing and Premium Regulation in Health Insurance Markets*, pages 3–20. Elsevier, 2018.

Riccardo Miotto, Li Li, Brian A Kidd, and Joel T Dudley. Deep patient: an unsupervised representation to predict the future of patients from the electronic health records. *Scientific Reports*, 6:26094, 2016.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. 2017.

Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019.

A Rajkomar, E Oren, K Chen, Hajaj N Dai AM, PJ Liu, et al. Scalable and accurate deep learning for electronic health records. npj Digit Med. 2018.

Sherri Rose. A machine learning framework for plan payment risk adjustment. *Health Services Research*, 51(6):2358–2374, 2016.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

Benjamin Shickel, Patrick James Tighe, Azra Bihorac, and Parisa Rashidi. Deep EHR: a survey of recent advances in deep learning techniques for electronic health record (EHR) analysis. *IEEE Journal of Biomedical and Health Informatics*, 22(5):1589–1604, 2017.

Akritee Shrestha, Savannah Bergquist, Ellen Montz, and Sherri Rose. Mental health risk adjustment with clinical categories and machine learning. *Health Services Research*, 53: 3189–3206, 2018.

Huan Song, Deepta Rajan, Jayaraman J Thiagarajan, and Andreas Spanias. Attend and diagnose: Clinical time series analysis using attention models. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

Jinghe Zhang, Kamran Kowsari, James H Harrison, Jennifer M Lobo, and Laura E Barnes. Patient2vec: A personalized interpretable deep representation of the longitudinal electronic health record. *IEEE Access*, 6:65333–65346, 2018a.

Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253, 2018b.

Appendix A. Results for other metrics

We include the results for different metrics for the same experiments presented in Section 5

Table 5: Test MAE score for year 2016 (N = # of training examples)

Training Year(s)		Lasso	LightGBM	LSTM$_{attn}$
2015	$N = 10K$	3.5507	3.1662	3.1826
	$N = 100K$	3.5493	3.0300	3.0050
	$N = 500K$	3.5500	2.975	**2.9415**
2014, 2015	$N = 10K$	3.3014	2.9055	2.9634
	$N = 100K$	3.3073	2.7964	2.7971
	$N = 500K$	3.3050	2.7417	**2.6854**
2013, 2014, 2015	$N = 10K$	3.2975	2.7860	2.8898
	$N = 100K$	3.2220	2.6958	2.6508
	$N = 500K$	3.2197	2.6398	**2.5661**

Table 6: Test Spearman Correlation for year 2016 (N = # of training examples)

Training Year(s)		Lasso	LightGBM	LSTM$_{attn}$
2015	$N = 10K$	0.1389	0.4107	0.4132
	$N = 100K$	0.1363	0.4672	0.4724
	$N = 500K$	0.1383	0.4883	**0.4942**
2014, 2015	$N = 10K$	0.2542	0.4680	0.4781
	$N = 100K$	0.2464	0.5206	0.5161
	$N = 500K$	0.2450	0.5410	**0.5462**
2013, 2014, 2015	$N = 10K$	0.2880	0.5004	0.4941
	$N = 100K$	0.2824	0.5489	0.5502
	$N = 500K$	0.2993	0.5700	**0.5876**

Appendix B. Results when trained with previous visit count feature

We include results collected from preliminary stages of our study, where we used the # of previous visits as a feature for both the baselines and the $LSTM\_simple$ model.

Table 7: Mean Absolute Error on predicted number of patient visits for the year 2016

Training Year(s)	% of Training Data	Lasso	LightGBM	LSTM_{simple}
2015	1	3.6341	3.4643	3.2223
	10	3.6380	3.3189	3.0462
	100	3.6411	3.2637	**2.9523**
2014, 2015	1	3.2971	3.1231	2.8638
	10	3.2932	2.9845	2.7328
	100	3.2905	2.9430	**2.6470**
2013, 2014, 2015	1	3.1219	2.9402	2.7262
	10	3.1204	2.8293	2.5699
	100	3.1154	2.7762	**2.4836**

Table 8: R^2 on predicted number of patient visits for the year 2016

Training Year(s)	% of Training Data	Lasso	LightGBM	LSTM_{simple}
2015	1	0.0725	0.1323	0.0998
	10	0.0704	0.1976	0.1870
	100	0.0698	0.2214	**0.2334**
2014, 2015	1	0.1121	0.1608	0.1590
	10	0.1102	0.2200	0.2132
	100	0.1112	0.2384	**0.2486**
2013, 2014, 2015	1	0.1171	0.1563	0.1288
	10	0.1171	0.2150	0.2251
	100	0.1170	0.2372	**0.2531**

Appendix C. Validating the LSTM model across multiple runs

Since the LSTM based deep sequential model architecture has several inherent uncertainties due to random initialization of weights, dropout probability, etc., we trained the entire model one other time in order to verify if the results from the first and the second runs are comparable. These results are presented in table 9.

Table 9: Test R^2 score for year 2016 ($N = 500K$) using LSTM model for multiple runs

Training Year(s)	Run 1	Run 2
2015	0.2974	0.2971
2014, 2015	0.3259	0.3242
2013, 2014, 2015	0.3436	0.3484

Appendix D. Parameters for the LightGBM model

In this section we list the parameters used in the LightGBM model.

Table 10: LGBM Parameters

Parameter	**Value**
Boosting type	GBDT
Objective	Poisson
Metric	MSE
# of leaves	128
Max depth	-1
Learning rate	0.01
Feature fraction	0.8
Early stopping round	100
# of estimators	10,000