Multi-layer Representation Learning for Medical Concepts

Edward Choi[1], Mohammad Taha Bahadori[1], Elizabeth Searles[2], Catherine Coffey[2],
Michael Thompson[2], James Bost[2], Javier Tejedor-Sojo[2], Jimeng Sun[1]
[1]Georgia Institute of Technology [2]Children's Healthcare of Atlanta
mp2893@gatech.edu, mohammad.bahadori@cc.gatech.edu, {elizabeth.searles
catherine.coffey, michael.thompson, james.bost, javier.tejedor-sojo}choa.org,
jsun@cc.gatech.edu

ABSTRACT

Proper representations of medical concepts such as diagnosis, medication, procedure codes and visits from Electronic Health Records (EHR) has broad applications in healthcare analytics. Patient EHR data consists of a sequence of visits over time, where each visit includes multiple medical concepts, e.g., diagnosis, procedure, and medication codes. This hierarchical structure provides two types of relational information, namely sequential order of visits and co-occurrence of the codes within a visit. In this work, we propose `Med2Vec`, which not only learns the representations for both medical codes and visits from large EHR datasets with over million visits, but also allows us to interpret the learned representations confirmed positively by clinical experts. In the experiments, `Med2Vec` shows significant improvement in prediction accuracy in clinical applications compared to baselines such as Skip-gram, GloVe, and stacked autoencoder, while providing clinically meaningful interpretation.

Keywords

Representation Learning; Medical Concepts; Healthcare Analytics; Neural Networks

1. INTRODUCTION

Discovering efficient representations of high dimensional concepts has been a key challenge in a variety of applications recently [2]. Using various types of neural networks, high-dimensional data can be transformed to continuous real-valued concept vectors that efficiently capture their latent relationship from data. Such succinct representations have been shown to improve the performance of various complex tasks across domains spanning from image processing [22, 17, 36], language modeling [3, 25], word embedding [26, 30], music information retrieval [31], sentiment analysis [32], and multi-modal learning of images and text [18].

Efficient representations for medical concepts is an important, if not essential, element in healthcare applications as well. Medical concepts contain rich latent relationships that cannot be represented by simple one-hot coding [29, Chapter 2.3.2]. For example, pneumonia and bronchitis are clearly more related than pneumonia and obesity. In one-hot coding, such relationship between different codes are not represented. Despite its limitation, many healthcare applications [7, 33] still use the simple sum over one-hot vectors to derive patient feature vectors. To overcome this limitation, it is common in healthcare applications, to rely on carefully designed feature representations [34, 16, 38]. However, this process often involves ad-hoc feature engineering that requires considerable expert medical knowledge and is not scalable nor comprehensive in general.

Recently, studies have shown that it is possible to learn efficient representations of healthcare concepts without medical expertise while significantly improving the performance of various healthcare predictive models. [9, 10, 8, 23, 6] Despite this progress, learning efficient representations of healthcare concepts, however, is still an open challenge. The difficulty stems from several aspects:

1. Electronic Health Record (EHR) data have a unique structure where the visits are temporally ordered but the medical codes within a visit form an unordered set. A sequence of visits possesses sequential relationship among them which cannot be captured by simply aggregating code-level representations. Moreover, given the demographic information for patients, the structure of EHR becomes more complex.

2. Learned representations should be interpretable. While the interpretability of the representation in healthcare applications is essential, many of the state-of-the-art representation learning methods such as recurrent neural networks (RNN) are difficult to interpret.

3. The algorithm should be scalable enough to handle large EHR datasets with hundreds of thousands of patients and millions of visits.

To address such challenges in healthcare concept representation learning, we propose `Med2Vec` and make the following contributions.

- We propose `Med2Vec`, a simple and robust algorithm to efficiently learn succinct code-, and visit-level representations by using real-world EHR datasets, without depending on expert medical knowledge.

- `Med2Vec` learns interpretable representations and enables clinical applications to offer more than just improved performances. We conducted a detailed user

KDD '16, August 13-17, 2016, San Francisco, CA, USA
© 2016 ACM. ISBN 978-1-4503-4232-2/16/08...$15.00
DOI: http://dx.doi.org/10.1145/2939672.2939823

study with clinical experts to validate the interpretability of the resulting representation.

- We conduct experiments to demonstrate the scalability of Med2Vec, and show that our model can be readily applied to near 30K medical codes over two large datasets with 3 million and 5.5 million visits, respectively.
- We apply the learned representations to multiple real-world healthcare prediction problems and demonstrate the improved performance enabled by Med2Vec compared to several baselines.

In the following section, we discuss related works, then describe our method in section 3. In section 4, we explain experiment design and interpretation method in detail. We present the results and discussion in section 5. Then we conclude this paper with future work in section 6.

2. PRELIMINARIES AND RELATED WORK

In this section, we first describe the preliminary ideas used in learning representation for words. Then, we review the algorithms developed for representing healthcare data.

2.1 Learning representation for words

Representation learning of words using neural network based methods have been studied since the early 2000's [3, 12, 28, 35]. Among these techniques, Skip-gram [26] is the basis of many concept representation learning methods, including our own. Skip-gram is able to capture the subtle relationships between words, thus outperforming the previous works in a word analogy task[24].

Given a sequence of words $w_1, w_2, \dots, w_T$, Skip-gram learns the word representations based on the co-occurrence information of words inside a context window of a predefined size. The key principle of Skip-gram is that a word's representation should be able to predict the neighboring words. The objective of Skip-gram is to maximize the following average log probability.

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

where c is the size of the context window. The conditional probability is defined by the softmax function:

$$p(w_O|w_I) = \frac{\exp\left(v'^{\top}_{w_O} v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left(v'^{\top}_{w} v_{w_I}\right)}$$

where v_w and v'_w are the *input* and *output* vector representations of word w. W is the number of words in the vocabulary. Basically, Skip-gram tries to maximize the softmax probability of the inner product of the center word's vector and its context word's vectors.[1]

Pennington et al. proposed GloVe, [30] which learns another word representations by using a similar principle as Skip-gram. GloVe uses the global word co-occurrence matrix to learn the word representations. Since the global co-occurrence matrix is often sparse, GloVe can be computationally less demanding than Skip-gram, which is a neural network model using the sliding context window. On the other hand, GloVe employs a weighting function that could require a considerable amount tuning effort.

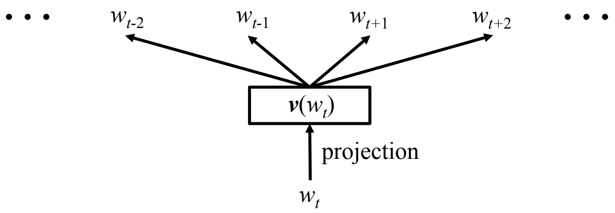


Figure 1: Skip-gram model architecture: $\boldsymbol{v}(w_t)$ is a vector representation for the word w_t. The goal of Skip-gram is to learn vector representations of words that are good at predicting neighboring words.

Beyond one level representation like Skip-gram and GloVe, researchers also proposed hierarchical learning representations for the text corpus, which has some analogy to our healthcare setting with two level concepts namely: codes and visits. Le and Mikolov [20] proposed to learn representations for paragraphs and words simultaneously by treating paragraphs indicators as words. However, their algorithm assigns a fixed set of vectors for both words and paragraphs in the training data. Moreover, their approach does not capture the sequential order among paragraphs. Skip-thought [19] proposed an encoder-decoder structure: an encoder (Gated Recurrent Units (GRU) in their case) learns a representation for a sentence that is able to regenerate its surrounding sentences (via GRU again). Skip-thought cannot be applied directly to EHR data because unlike words in sentences, the codes in a visit are unordered. Also, the interpretation of Skip-thought model is difficult, as they rely on complex RNNs.

2.2 Representation learning in healthcare

Recently researchers start to explore the possibility of efficient representation learning in the medical domain.

Medical text analysis.

Minarro et al. [27] learns the representations of medical terms by applying Skip-gram to various medical text collected from PubMed, Merck Manuals, Medscape and Wikipedia. De Vine et al. [13] learns the representations of UMLS concepts from free-text patient records and medical journal abstracts. They first replaced the words in documents to UMLS concepts, then applied Skip-gram to learn the distributed representations of the concepts. However, none of them studied longitudinal EHR data with a large number of medical codes.

Structured visit records analysis.

Skip-gram was directly applied to structured longitudinal visit records to learn the representation of medical codes (*e.g.* diagnosis, medication, procedure codes) [9, 10]. In [9], the authors demonstrated that simply aggregating the learned representation of medical codes to create a visit representation leads to improved predictive performance. However, simply aggregating the code representations is not the optimal method to generate a visit representation as it completely ignores the temporal relations across adjacent visits. We believe that taking advantage of the two-level in-

[1] Mikolov et al. [26] also use hierarchical softmax and negative sampling to speed up the learning process. We focus on the original simple formulation.

formation (the co-occurrence of codes within a visit and the sequential nature of visits) and the demographic information of patients will give us better representation for both medical codes and patient visits. RNNs have been applied to analysis of longitudinal patient records [8, 23] and can generate both code and patient representations. However, despite their outstanding predictive performance, RNNs are difficult to interpret [6] which limits their applications in healthcare.

3. METHOD

In this section, we describe the proposed algorithm Med2Vec. We start by mathematically formulating the EHR data structure and our goal. Then we describe our approach in a top-down fashion. We also explain how to interpret the learned representations. We conclude this section with complexity analysis.

EHR structure and our notation.

We denote the set of all medical codes $c_1, c_2, \ldots, c_{|\mathcal{C}|}$ in our EHR dataset by $\mathcal{C}$ with size $|\mathcal{C}|$. EHR data for each patient is in the form of a sequence of visits $V_1, \ldots, V_T$ where each visit contains a subset of medical codes $V_t \subseteq \mathcal{C}$. Without loss of generality, all algorithms will be presented for a single patient to avoid cluttered notations. The goal of Med2Vec is to learn two types of representations:

Code representations We aim to learn an embedding function $f_C : \mathcal{C} \mapsto \mathbb{R}^m_+$ that maps every code in the set of all medical codes $\mathcal{C}$ to non-negative real-valued vectors of dimension m. The non-negativity constraint is introduced to improve interpretability, as discussed in details in Section 3.5.

Visit representations Our second task is to learn another embedding function $f_V : \mathcal{V} \mapsto \mathbb{R}^n$ that maps every visit (a set of medical codes) to a real-valued vector of dimension n. The set $\mathcal{V}$ is the power set of the set of codes $\mathcal{C}$.

3.1 Med2Vec architecture

Figure 2 depicts the architecture of Med2Vec. Given a visit V_t, we use a multi-layer perceptron (MLP) to generate the corresponding visit representation $\boldsymbol{v}_t$. First, visit V_t is represented by a binary vector $\boldsymbol{x}_t \in \{0,1\}^{|\mathcal{C}|}$ where the i-th entry is 1 only if $c_i \in V_t$. Then $\boldsymbol{x}_t$ is converted to an intermediate visit representation $\boldsymbol{u}_t \in \mathbb{R}^m$ as follows,

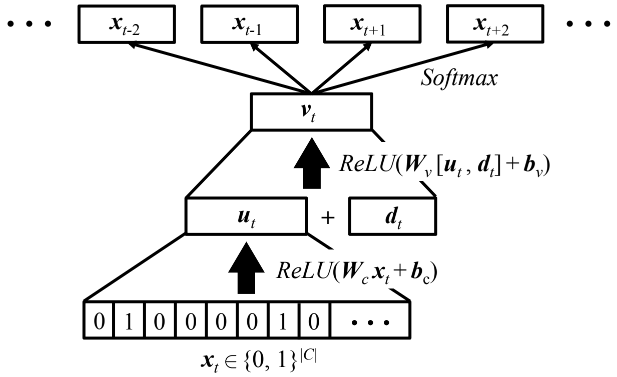
$$\boldsymbol{u}_t = ReLU(\boldsymbol{W}_c \boldsymbol{x}_t + \boldsymbol{b}_c) \tag{1}$$

using the code weight matrix $\boldsymbol{W}_c \in \mathbb{R}^{m \times |\mathcal{C}|}$ and the bias vector $\boldsymbol{b}_c \in \mathbb{R}^m$. The rectified linear unit is defined as $ReLU(\boldsymbol{v}) = \max(\boldsymbol{v}, \boldsymbol{0})$. Note that max() applies element-wise to vectors. We use the rectified linear unit (ReLU) as the activation function to enable interpretability, which will be discussed in section 3.3.

We concatenate the demographic information $\boldsymbol{d}_t \in \mathbb{R}^d$, where d is the size of the demographic information vector, to the intermediate visit representation $\boldsymbol{u}_t$ and create the final visit representation $\boldsymbol{v}_t \in \mathbb{R}^n$ as follows,

$$\boldsymbol{v}_t = ReLU(\boldsymbol{W}_v[\boldsymbol{u}_t, \boldsymbol{d}_t] + \boldsymbol{b}_v)$$

using the visit weight matrix $\boldsymbol{W}_v \in \mathbb{R}^{n \times (m+d)}$ and the bias vector $\boldsymbol{b}_v \in \mathbb{R}^n$, where n is the predefined size of the visit representation. We use ReLU once again as the activation function. We discuss our efficient training procedure of the parameters $\boldsymbol{W}_c, \boldsymbol{b}_c, \boldsymbol{W}_v$ and $\boldsymbol{b}_v$ in the next subsection.



• • •
$\boldsymbol{x}_{t\text{-}2}$
$\boldsymbol{x}_{t\text{-}1}$
$\boldsymbol{x}_{t+1}$
$\boldsymbol{x}_{t+2}$
• • •
Softmax
$\boldsymbol{v}_t$
$ReLU(\boldsymbol{W}_v[\boldsymbol{u}_t, \boldsymbol{d}_t]+\boldsymbol{b}_v)$
$\boldsymbol{u}_t$
+
$\boldsymbol{d}_t$
$ReLU(\boldsymbol{W}_c\boldsymbol{x}_t+\boldsymbol{b}_c)$
0 1 0 0 0 0 1 0 • • •
$\boldsymbol{x}_t \in \{0, 1\}^{|C|}$

Figure 2: Structure of Med2Vec: A visit comprised of several medical codes is converted to a binary vector $\boldsymbol{x}_t \in \{0,1\}^{|\mathcal{C}|}$. The binary vector is then converted to an intermediate visit representation $\boldsymbol{u}_t$. $\boldsymbol{u}_t$ is concatenated with a vector of demographic information $\boldsymbol{d}_t$, and converted to the final visit representation $\boldsymbol{v}_t$, which is trained to predict its neighboring visits $\ldots, \boldsymbol{x}_{t-2}, \boldsymbol{x}_{t-1}, \boldsymbol{x}_{t+1}, \boldsymbol{x}_{t+2}, \ldots$

3.2 Learning from the visit-level information

As mentioned in the introduction, the sequential information of visits can be exploited for learning efficient representations of visits and potentially codes. We train the MLP using a very straightforward intuition as follows: a visit describes a state in a continuous process that is a patient's clinical experience. Therefore, given a visit representation, we should be able to predict what has happened in the past, and what will happen in the future. Specifically, given a visit representation $\boldsymbol{v}_t$, we train a softmax classifier that predicts the medical codes of the visits within a context window[2]. We minimize the cross entropy error as follows,

$$\min_{\boldsymbol{W}_s, \boldsymbol{b}_s} \frac{1}{T} \sum_{t=1}^{T} \sum_{-w \le i \le w, i \ne 0} -\boldsymbol{x}_{t+i}{}^{\top} \log \hat{\boldsymbol{y}}_t - (\boldsymbol{1} - \boldsymbol{x}_{t+i})^{\top} \log(\boldsymbol{1} - \hat{\boldsymbol{y}}_t), \tag{2}$$

$$\text{where} \qquad \hat{\boldsymbol{y}}_t = \frac{\exp(\boldsymbol{W}_s \boldsymbol{v}_t + \boldsymbol{b}_s)}{\sum_{j=1}^{|\mathcal{C}|} \exp(\boldsymbol{W}_s[j,:]\boldsymbol{v}_t + \boldsymbol{b}_s[j])}$$

where $\boldsymbol{W}_s \in \mathbb{R}^{|\mathcal{C}| \times n}$ and $\boldsymbol{b}_s \in \mathbb{R}^{|\mathcal{C}|}$ are the weight matrix and bias vector for the softmax classifier, w the predefined context window size, exp the element-wise exponential function, and $\boldsymbol{1}$ denotes an all one vector. We have used MATLAB's notation for selecting a row in $\boldsymbol{W}_s$ and a coordinate of $\boldsymbol{b}_s$.

3.3 Learning from the code-level information

As we described in the introduction, healthcare datasets contain two-level information: visit-level sequence information and code-level co-occurrence information. Since the loss function in Eq. (2) can efficiently capture the sequence level information, now we need to find a way to use the second source of information, i.e., the intra-visit co-occurrence of the codes.

A natural choice to capture the code co-occurrence information is to use Skip-gram. The main idea would be that

[2] We also tried predicting the visit representations $\ldots, \boldsymbol{v}_{t-1}, \boldsymbol{v}_{t+1}, \ldots$ instead of the medical codes, but obtained poor results.

the representations for the codes that occur in the same visit should predict each other. To embed Skip-gram in `Med2Vec`, we can train $\boldsymbol{W}_c \in \mathbb{R}^{m\times|\mathcal{C}|}$ (which also produces intermediate visit level representations) so that the i-th column of $\boldsymbol{W}_c$ will be the representation for the i-th medical code among total $|\mathcal{C}|$ codes. Note that given the unordered nature of the codes inside a visit, unlike the original Skip-gram, we do not distinguish between the "input" medical code and the "output" medical code. In text, it is sensible to assume that a word can serve a different role as a center word and a context word, whereas in EHR datasets, we cannot classify codes as center or context codes. It is also desirable to learn the representations of different types of codes (*e.g.* diagnosis, medication, procedure code) in the same latent space so that we can capture the hidden relationships between them.

However, coordinate-wise interpretation of Skip-gram codes is not straightforward because the positive and negative values of $\boldsymbol{W}_c$ make it hard for each coordinate to focus on a single coherent medical concept. For intuitive interpretation, we should learn code representations with non-negative values. Note that in Eq.(1), if the binary vector $\boldsymbol{x}_t$ is a one-hot vector, then the intermediate visit representation $\boldsymbol{u}_t$ becomes a code representation. Therefore, using the Skip-gram algorithm, we train the non-negative weight $ReLU(\boldsymbol{W}_c)$ instead of $\boldsymbol{W}_c$. This will not only use the intra-visit co-occurrence information, but also guarantee non-negative code representations. Moreover, ReLU produces sparse code representations, which further facilitates easier interpretation of the codes.

The code representations to be learned is denoted as a matrix $\boldsymbol{W}'_c = ReLU(\boldsymbol{W}_c) \in \mathbb{R}^{m\times|\mathcal{C}|}$. From a sequence of visits $V_1, V_2, \ldots, V_T$, the code-level representations can be learned by maximizing the following log-likelihood,

$$\min_{\boldsymbol{W}'_c} \quad \frac{1}{T}\sum_{t=1}^{T}\sum_{i:c_i\in V_t}\sum_{j:c_j\in V_t, j\neq i} \log p(c_j|c_i), \qquad (3)$$

$$\text{where} \quad p(c_j|c_i) = \frac{\exp\left(\boldsymbol{W}'_c[:,j]^\top \boldsymbol{W}'_c[:,i]\right)}{\sum_{k=1}^{|\mathcal{C}|}\exp\left(\boldsymbol{W}'_c[:,k]^\top \boldsymbol{W}'_c[:,i]\right)}. \qquad (4)$$

3.4 Unified training

The single unified framework can be obtained by adding the two objective functions (3) and (2) as follows,

$$\underset{\boldsymbol{W}_{c,v,s},\boldsymbol{b}_{c,v,s}}{\operatorname{argmin}} \frac{1}{T}\sum_{t=1}^{T}\Big\{ -\sum_{i:c_i\in V_t}\sum_{j:c_j\in V_t,j\neq i}\log p(c_j|c_i)$$
$$+ \sum_{-w\leq k\leq w, k\neq 0} -\boldsymbol{x}_{t+k}^\top \log\hat{\boldsymbol{y}}_t - (\boldsymbol{1}-\boldsymbol{x}_{t+k})^\top \log(\boldsymbol{1}-\hat{\boldsymbol{y}}_t)\Big\}$$

By combining the two objective functions we learn both code representations and visit representations from the same source of patient visit records, exploiting both intra-visit co-occurrence information as well as inter-visit sequential information at the same time.

3.5 Interpretation of learned representations

While the original Skip-gram learns code representations that have interesting properties such as additivity, in healthcare we need stronger interpretability. We need to be able to associate clinical meaning to each dimension of both code and visit representations. Interpreting the learned representations is based on analyzing each coordinate in both code and visit embedding spaces.

Interpreting code representations.

If information is properly embedded into a lower dimensional non-negative space, each coordinate of the lower dimension can be readily interpreted. Non-negative matrix factorization (NMF) is a good example. Since we trained $ReLU(\boldsymbol{W}_c) \in \mathbb{R}^{m\times|\mathcal{C}|}$, a non-negative matrix, to represent the medical codes, we can employ a simple method to interpret the meaning of each coordinate of the m-dimensional code embedding space. We can find the top k codes that have the largest values for the i-th coordinate of the code embedding space as follows,

$$\operatorname{argsort}(\boldsymbol{W}_c[i,:])[1:k]$$

where argsort returns the indices of a vector that index its values in a descending order. By studying the returned medical codes, we can view each coordinate as a disease group. Detailed examples are given in section 5.1

Interpreting visit representations.

To interpret the learned visit vectors, we can use the same principle we used for interpreting the code representation. For the i-th coordinate of the n-dimensional visit embedding space, we can find the top k coordinates of the code embedding space that have the strongest values as follows,

$$\operatorname{argsort}(\boldsymbol{W}_v[i,:])[1:k]$$

where we use the same argsort as before. Once we obtain a set of code coordinates, we can use the knowledge learned from interpreting the code representations to understand how each visit coordinate is associated with a group of diseases. This simple interpretation is possible because the intermediate visit representation $\boldsymbol{u}_t$ is a non-negative vector, due to the $ReLU$ activation function.

In the experiments, we also tried to find the input vector $\boldsymbol{x}_t$ that most activates the target visit coordinate [14, 21]. However, the results were very sensitive to the initial value of $\boldsymbol{x}_t$, and even averaging over multiple samples were producing unreliable results.

3.6 Complexity analysis

We first analyze the computational complexity of the code-level objective function Eq. (3). Without loss of generality, we assume the visit records of all patients are concatenated into a single sequence of visits. Then the complexity for Eq. (3) is as follows,

$$\mathcal{O}(T\overline{M}^2|\mathcal{C}|m)$$

where T is the number of visits, $\overline{M}^2$ is the average of squared number of medical codes within a visit, $|\mathcal{C}|$ the number of unique medical codes, m the size of the code representation. The M^2 factor comes from iterating over all possible pairs of codes within a visit. The complexity of the visit-level objective function Eq.(2) is as follows,

$$\mathcal{O}(Tw(|\mathcal{C}|(m+n)+mn))$$

where w is the size of the context window, n the size of the visit representation. The added terms come from generating a visit representation via MLP. Since size of code representation m and size of visit representation n generally have the same order of magnitude, we can replace n with m. Furthermore, m is generally smaller than $|\mathcal{C}|$ by at least two orders of magnitude. Therefore the overall complexity of Med2Vec can be simplified as follows.

$$\mathcal{O}(T|\mathcal{C}|m(\overline{M}^2 + w))$$

Here we notice that $\overline{M}^2$ is generally larger than w. In our work, the average number of codes $\overline{M}$ per visit for two datasets are 7.88 and 3.19 according to Tables 1, respectively, whereas we select the window size w to be at most 5 in our experiments. Therefore the complexity of Med2Vec is dominated by the code representation learning process, for which we use the Skip-gram algorithm. This means that exploiting visit-level information to learn efficient representations for both visits and codes does not incur much additional cost.

4. EXPERIMENTS

In this section, we evaluate the performance of Med2Vec in both public and proprietary datasets. First we describe the datasets. Then we describe evaluation strategies for code and visit representations, along with implementation details. Then we present the experiment results of code and visit representations with discussion. We conclude with convergence and scalability study. We make the source code of Med2Vec publicly available at https://github.com/mp2893/med2vec.

4.1 Dataset description

We evaluate performance of Med2Vec on a dataset provided by Children's Healthcare of Atlanta (CHOA)[3]. We extract visit records from the dataset, where each visit contains several medical codes (*e.g.* diagnosis, medication, procedure codes). The diagnosis codes follow ICD-9 codes, the medication codes are denoted by National Drug Codes (NDC), and the procedure codes follow Category I of Current Procedural Terminology (CPT). We exclude patients who had less that two visits to showcase Med2Vec's ability to use sequential information of visits. The basic statistics of the dataset are summarized in Table 1. The data are fully de-identified and do not include any personal health information (PHI).

We divide the dataset into two groups in a 4:1 ratio. The former is used to train Med2Vec. The latter is held out for evaluating the visit-level representations, where we train models to predict visit-related labels.

We also use CMS dataset, a publicly available [4] synthetic medical claims dataset. The basic information of CMS is also given in Table 1. Compared to CHOA dataset, the CMS dataset has more patients but fewer unique medical codes. The average number of codes per visit is also smaller than that of CHOA dataset. Since CMS dataset is synthetic, we use it only for testing the scalability in section 4.7.

4.2 Evaluation Strategy of code representations

Qualitative evaluation by medical experts.

For a comprehensive qualitative evaluation, we perform a *relatedness* test by selecting 100 most frequent diagnosis codes and their 5 closest diagnoses, medications and procedures in terms of cosine similarity. This will allow us to know if the learned representations effectively capture the latent relationships among them. Two medical experts from CHOA check each item and assign *related*, *possible* and *unrelated* labels.

Table 1: Basic statistics of CHOA and CMS dataset.

Dataset	**CHOA**	**CMS**
# of patients	550,339	831,210
# of visits	3,359,240	5,464,950
Avg. # of visits per patient	6.1	6.57
# of unique medical codes	28,840	21,033
- # of unique diagnosis codes	10,414	14,111
- # of unique medication codes	12,892	N/A
- # of unique procedure codes	5,534	6,922
Avg. # of codes per visit	7.88	3.19
Max # of codes per visit	440	44
(95%, 99%) percentile # of codes per visit	(22, 53)	(9, 13)

Quantitative evaluation with baselines.

We use medical code groupers to quantitatively evaluate the code representations. Code groupers are used to collapse individual medical codes into clinically meaningful categories. For example, Clinical Classifications Software (CCS) groups ICD9 diagnosis codes into 283 categories such as tuberculosis, bacterial infection, and viral infection.

We apply K-means clustering to the learned code representations and calculate the normalized mutual information (NMI) based on the group label of each code. We use the CCS as the ground truth for evaluating the code representation for diagnosis. For medication code evaluation, we use American Hospital Formulary Service (AHFS) pharmacologic-therapeutic classification, which groups NDC codes into 165 categories. For procedure code evaluation, we use the second-level grouping of CPT category I, which groups CPT codes into 115 categories.Thus, we set the number of clusters k to 283, 165, 115 respectively for the diagnosis, medication, procedure code evaluation, which matches the numbers of groups from individual groupers.

For baselines, we use popular methods that efficiently exploit co-occurrence information. Skip-gram (which is used in learning representations of medical concepts by [10, 9]) is trained using Eq. (3). GloVe will be trained on the co-occurrence matrix of medical codes, for which we counted the codes co-occurring within a visit. Additionally, we also report well-known baselines such as singular value decomposition on the co-occurrence matrix.

4.3 Evaluation strategy of visit representation

We evaluate the quality of the visit representations by performing two visit-level prediction tasks: predicting the future visit and predicting the present status. The former will evaluate a visit representation's potential effectiveness in predictive healthcare while the latter will evaluate the how well it captures the information in the given visit. The details of the two tasks are given below.

Predicting future medical codes: We predict the medical codes that will occur in the next visit using the visit representations. Specifically, given two consecutive visits V_i and V_j, the medical codes $c \in V_j$ will be the target $\boldsymbol{y}$, the

[3] http://www.choa.org/

[4] https://www.cms.gov/Medicare/Quality-Initiatives-Patient-Assessment-Instruments/OASIS/DataSet.html

medical codes $c \in V_i$ will be the input $\boldsymbol{x}$, and we use softmax to predict $\boldsymbol{y}$ given $\boldsymbol{x}$. The predictive performance will be measured by Recall@k due to its similarity to the differential diagnosis. Doctors iteratively perform differential diagnosis by generating a list of most likely diseases for an undiagnosed patient based on the available information. We set $k = 30$ to cover the complex cases of CHOA dataset, as over 167,000 visits are assigned with more than 20 medical codes according to Table 1. We predict the grouped medical codes, obtained by the medical groupers used in Section 4.2.
Predicting Clinical Risk Groups (CRG) level: A patient's CRG level indicates his severity level. It ranges from 1 to 9, including 5a and 5b. The CRG levels can be divided into two groups: non-severe (CRG 1-5a) and severe (CRG 5b-9). Given a visit, we use logistic regression to predict the binary CRG class associated with the visit. We use Area Under The Curve (AUC) to measure the classification accuracy, as it is more robust to class imbalance in data.

Baselines.

For baselines, we use the following methods.
Binary vector model (One-hot+): In order to compare with the raw input data, we use the binary vector $\boldsymbol{x}_t$ as the visit representation.
Stacked autoencoder (SA): Stacked autoencoder is one of the most popular unsupervised representation learning algorithms [37]. Using the binary vector $\boldsymbol{x}_t$ concatenated with patient demographic information as the input, we train a 3-layer stacked autoencoder (SA) [4] to minimize the reconstruction error.The trained SA will then be used to generate visit representations.
Sum of Skip-gram vectors (Skip-gram+): We first learn the code-level representations with Skip-gram only (Eq. (3)). Then for the visit-level representation, we simply add the representations of the codes within the visit. This approach was proven very effective for heart failure prediction in [9]. We append patient demographic information at the end.
Sum of GloVe vectors (GloVe+): We perform the same process as Skip-gram+, but use GloVe vectors instead of Skip-gram vectors. We use the recommended hyperparameter setting from [30].

Evaluation details.

We use the held-off dataset, which was *not* used to learn the code and visit representations, to perform the two prediction tasks. The held-off dataset contains 672,110 visits assigned with CRG levels. In order to train the predictors, we divide the held-out data to training and testing folds with ration 4:1. Both softmax and logistic regression are trained for 10 epochs on the training fold. We perform 5-fold cross validation for each task and report the average performance. For all baseline models and `Med2Vec`, we use age, sex and ethnicity as the demographic information in the input data.

4.4 Implementation and training details

For learning code and visit representations using `Med2Vec` and all baselines, we use Adadelta [39] in a mini-batch fashion. For Skip-gram, SA and `Med2Vec`, we use 1,000 visits[5] per batch. For GloVe, we use 1,000 non-zero entries of the co-occurrence matrix per batch. The optimization terminates after a fixed number of epochs. In section 4.6, we show the relationship between training epochs and the performance. We also show the convergence behavior of `Med2Vec` and the baselines in section 4.7.

Table 2: Average score of the medical codes from the relatedness test. 2 was assigned for *related*, 1 for *possible* and 0 for *unrelated*

Average	Diagnosis	Medication	Procedure
1.34	1.59	0.95	1.47

Table 3: Clustering NMI of the diagnosis, medication and procedure code representations of various models. All models learned 200 dimensional code vectors. All models except SVD were trained for 10 epochs.

Model	Diagnosis	Medication	Procedure
SVD ($\sigma \boldsymbol{V}^\top$)	0.1824	0.0843	0.1781
Skip-gram	0.2251	0.1216	0.2432
GloVe	0.4205	0.2163	0.3499
Med2Vec	0.2328	0.1089	0.21

`Med2Vec`, Skip-gram, GloVe and SA are implemented with Theano 0.7.0 [5]. K-means clustering for the code-level evaluation and SVD are performed using Scikit-learn 0.14.1. Softmax and logistic regression models for the visit-level evaluation are implemented with Keras 0.3.1, and trained for 10 epochs. All tasks are executed on a machine equipped with Intel Xeon E5-2697v3, 256GB memory and two Nvidia K80 Tesla cards.

We train multiple models using various hyperparameter settings. For all models we vary the size of the code representations m (or the size of the hidden layer for SA), and the number of training epochs. Additionally for `Med2Vec`, we vary the size of the visit representations n, and the size of the visit context window w.

To alleviate the curse of dimensionality when training the softmax classifier (Eq.(2)) of `Med2Vec`, we always use the medical code groupers of section 4.2 so that the softmax classifier is trained to predict the grouped medical codes instead of the exact medical codes. To confirm the impact of this strategy, we train an additional `Med2Vec` without using the medical code groupers.

4.5 Results of the code-level evaluation

Table 2 shows the average score of the medical codes from the qualitative code evaluation. On average, `Med2Vec` successfully captures the relationship between medical codes. However, `Med2Vec` seems to have a hard time capturing proper representation of medications. This is due to the precise nature the medication prescription. For example, `Med2Vec` calculated that *Ofloxacin*, an antibiotic sometimes used to treat middle-ear infection, was related to *sensorineural hearling loss* (SNHL), an inner-ear problem. On the surface level, this is a wrong relationship. But `Med2Vec` can be seen as capturing the deeper relationship between medical concepts that is not always clear on the surface level.

Table 3 shows the clustering NMI of diagnosis, medication and procedure codes, measured for various models. `Med2Vec`

[5] for efficient computation, we preprocessed the EHR dataset so that the visit records of all patients are concatenated into a single sequence of visits.

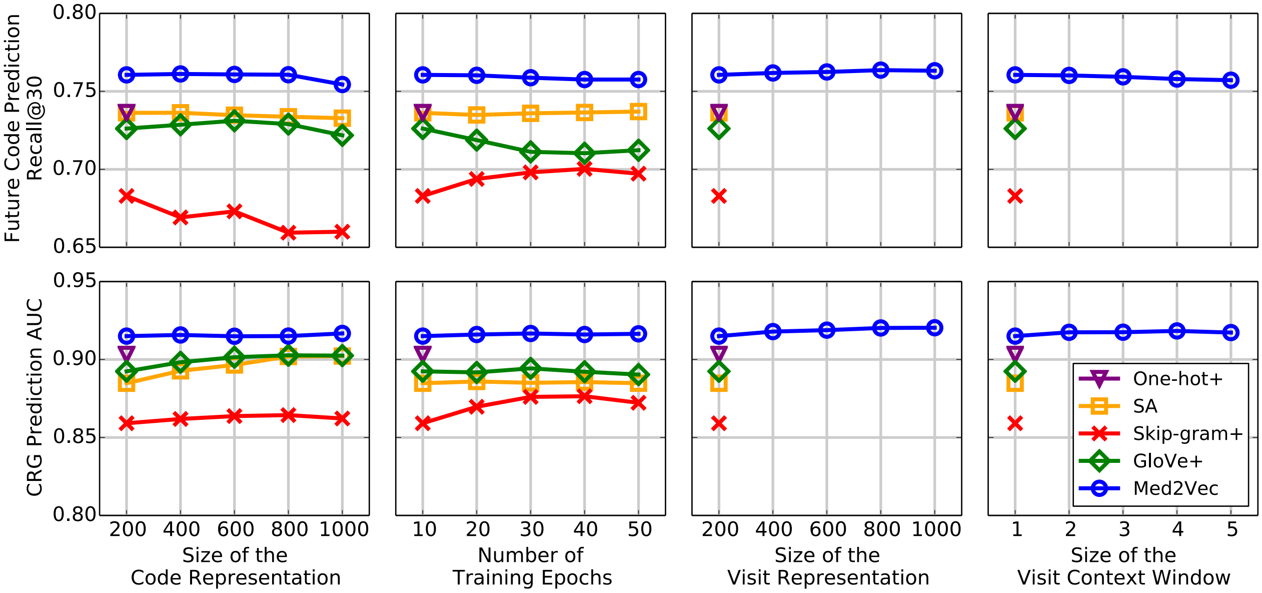


Figure 3: The top row and the bottom row respectively show the Recall@30 for predicting the future medical codes and the AUC for predicting the CRG class when changing different hyperparameters. The basic configuration for Med2Vec is $m, n = 200$, $w = 1$, and the training epoch set to 10. The basic configuration for all baseline models is 200 for code representation size (or hidden layer size) and training epoch also set to 10. In each column, we change one hyperparameter while fixing others to the basic configuration.

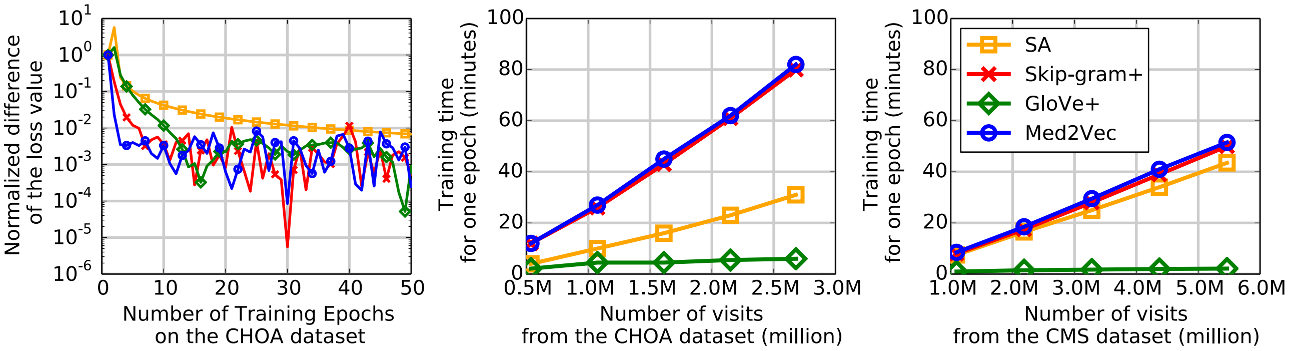


Figure 4: The first figure shows the convergence behavior of all models on the CHOA dataset. The second and third figures show the relationship between the training time and the dataset size for all models respectively using the CHOA dataset and the CMS dataset.

shows more or less similar conformity to the existing groupers as Skip-gram. SVD shows the weakest conformity among all models. GloVe exhibits significantly stronger conformity than any other models. Exploiting the global co-occurrence matrix seems to help learn code representations where similar codes are closer to each other in terms of Euclidean distance.

However, the degree of conformity of the code representations to the groupers does not necessarily indicate how well the code representations capture the hidden relationships. For example, CCS categorizes ICD9 224.4 *Benign neoplasm of cornea* as CCS 47 *Other and unspecified benign neoplasm*, and ICD9 370.00 *Unspecified corneal ulcer* as CCS 91 *Other eye disorders*. But the two diagnosis codes are both eye related problems, and they could be considered related in that sense. Therefore we recommend the readers use the evaluation results for comparing the performance between `Med2Vec` and other baselines, rather than for measuring the absolute performance.

In the following visit-level evaluation, we show the dominant predictive performance of `Med2Vec` indicates that code representations' strong conformity to the groupers does not necessarily imply good visit representations.

4.6 Results of the visit-level evaluation

The first row of Figure 3 shows the Recall@30 for predicting the future medical codes. First, in all of the experiments, `Med2Vec` achieves the highest performance, despite the fact that it is constrained to be positive and interpretable. The second observation is that `Med2Vec`'s performance is robust to choice of the hyperparameters in a wide range of values. Comparing to a more volatile performance of Skip-gram, we can see that including the visit information in training not only improves the performance, but also stabilizes it too.

Another fascinating aspect of the results is the overfitting

Table 4: Performance comparison of two Med2Vec models. The top row was trained with the grouped code as mentioned in section 4.4. The bottom row was trained without using the groupers. Both models were trained for 10 epochs with $m, n = 200$, $w = 1$.

Model	Future code prediction	CRG prediction
Grouped codes	0.7605	0.9150
Exact codes	0.7574	0.9155

pattern in different algorithms. Increasing the code representation size degrades the performance of all of the algorithms, as it leads to overfitting. Similar behavior can be seen as we train GloVe+ for more epochs which suggests early stopping technique should be used in representation learning [1]. For Med2Vec, increasing the visit representation size n seems to have the strongest influence to its predictive performance.

The bottom row of figures in Figure 3 shows the AUC for predicting the CRG class of the given visit. The overfitting patterns are not as prominent as the previous task. This is due to the different nature of the two prediction tasks. While the goal of CRG prediction is to predict a value related to the current visit, predicting the future codes is taking a step away from the current visit. This different nature of the two tasks also contributes to the better performance of One-hot+ on the CRG prediction. One-hot+ contains the entire information of the given visit, although in a very high-dimensional space. Therefore predicting the CRG level, which has a tight relationship with the medical codes within a visit, is an easier task for One-hot+ than predicting the future codes.

Table 4 shows the performance comparison between two different Med2Vec models. The top model is trained with the grouped codes as explained in section 4.4, while the bottom models is trained with the exact codes. Considering the marginal difference of the CRG prediction AUC, it is evident that our strategy to alleviate the curse of dimensionality was beneficial. Moreover, using the grouped codes will improve the training speed as the softmax function will require less computation.

4.7 Convergence behavior and scalability

We compare the convergence behavior of Med2Vec with Skip-gram (Eq. (3)), GloVe and SA. For SA, we measure the convergence behavior of a single-layer. We train the models for 50 epochs and plot the normalized difference of the loss value $\frac{\mathcal{L}_t - \mathcal{L}_{t-1}}{\mathcal{L}_t}$, where $\mathcal{L}_t$ denotes the loss value at time t. We also study the scalability of all models except One-hot+, as there is no representation learning in it. We vary the size of the training data and plot the time taken for each model to run one epoch.

The left figure of Fig 4 shows the convergence behavior of all models when trained on the CHOA dataset. SA shows the most stable convergence behavior, which is natural given that we used a single-layer SA, a much less complex model compared to GloVe, Skip-gram and Med2Vec. All models except SA seem to reach convergence after 10 epochs of training. Note that Med2Vec shows similar, if not better convergence behavior compared to Skip-gram even with added complexity.

The center figure of Fig 4 shows the minutes taken to train all models for one epoch using the CHOA dataset. As we have analzyed in section ssec:complexity, Med2Vec takes essentially the same time to train for one epoch. Both Skip-gram and Med2Vec, however, takes longer than SA and GloVe. This is mainly due to having the softmax function for training the code representations. GloVe, which is trained on the very sparse co-occurrence matrix naturally takes the least time to train.

The right figure of Fig 4 shows the training time when using the CMS dataset. Note that Med2Vec and Skip-gram takes similar time to train as SA. This is due to the smaller number of codes per visit, which is the computationally dominating factor of both Med2Vec and Skip-gram. GloVe takes less time as the number of unique codes are smaller in the CMS dataset. SA, on the other hand, takes more time because the number of visits have doubled while the the number of unique codes is about 73% of that of the CHOA dataset.

5. INTERPRETATION

Given the importance of interpretability in healthcare, we demonstrate three stages of interpretability for our model in collaboration with the medical experts from CHOA. First, to analyze the learned code representations we show top five medical codes for each of six coordinates of the code embedding space and explain the characteristic of each coordinate. This way, we show how we can annotate each dimension of the code embedding space with clinical concepts. The six coordinates are specifically chosen so that they can be used in the later stages. Second, we demonstrate the interpretability of Med2Vec's visit representations by analyzing the meaning of two coordinates in the visit embedding space.

Finally, we extend the interpretability of Med2Vec to a real-world task, the CRG prediction, and analyze the medical codes that have strong influence on the CRG level. Once we learn the logistic regression weight $\boldsymbol{w}_{LR}$ for the CRG prediction, we can extract knowledge from the learned weights by analyzing the visit coordinates to which the weights are strongly connected.

Instead of analyzing the visit coordinates, however, we propose an approximate way of directly finding out which code coordinate plays an important role in predicting the CRG class. Our goal is to find $\boldsymbol{u}_t$ such that maximizes the output activation as follows[6]

$$\boldsymbol{u}_t^\star = \underset{\boldsymbol{u}_t, \|\boldsymbol{u}_t\|_2=1, \boldsymbol{u}_t \succeq 0}{\operatorname{argmax}} \left[ReLU(\boldsymbol{W}_v \boldsymbol{u}_t + \boldsymbol{b}_v) \right]^\top \boldsymbol{w}_{LR} \qquad (5)$$

Given the fact that $ReLU(\cdot)$ is an increasing function (not-strictly though), we make an approximation and find the solution without the $ReLU(\cdot)$ term. The approximate solution can be found in closed form $\boldsymbol{u}_t^\star \propto (\boldsymbol{W}_v^\top \boldsymbol{w}_{LR})_+$. Finally, we calculate the element-wise product of $\boldsymbol{u}_t^\star$ and $\max(\boldsymbol{W}_c + \boldsymbol{b}_c)$. This is to take into account the fact that each code coordinate has different maximum value. Therefore, instead of simply selecting the code coordinate with the strongest connection to the CRG level, we consider each coordinate's maximum ability to activate the positive CRG prediction.

The resulting vector will show the maximum influence each code coordinate can have on the CRG prediction.

[6] As we are interested in influential codes, we assume the demographic information vector is zero vector and omit it for ease of notation.

Table 5: Medical codes with the strongest value in six different coordinates of the 200 dimensional code embedding space. We choose ten medical codes per coordinate. Shortened descriptions of diagnosis codes are compensated by their ICD9 codes. Medications and procedures are appended with (R) and (P) respectively.

Coordinate 112	Coordinate 152	Coordinate 141
Kidney replaced by transplant (V42.0)	X-ray, knee (P)	Cystic fibrosis (277.02)
Hb-SS disease without crisis (282.61)	X-ray, thoracolumbar (P)	Intracranial injury (854.00)
Heart replaced by transplant (V42.1)	Accidents in public building (E849.6)	Persistent mental disorders (294.9)
RBC antibody screening (P)	Activities involving gymnastics (E005.2)	Subdural hemorrhage (432.1)
Complications of transplanted bone marrow (996.85)	Struck by objects/persons in sports (E917.0)	Neurofibromatosis (237.71)
	Encounter for removal of sutures (V58.32)	Other conditions of brain (348.89)
Sickle-cell disease (282.60)	Struck by object in sports (E917.5)	Conductive hearing loss (389.05)
Liver replaced by transplant (V42.7)	Unspecified fracture of ankle (824.8)	Unspecified causes of encephalitis, myelitis, encephalomyelitis (323.9)
Hb-SS disease with crisis (282.62)	Accidents occurring in place for recreation and sport (E849.4)	
Prograf PO (R)		Sensorineural hearing loss (389.15)
Complications of transplanted heart (996.83)	Activities involving basketball (E007.6)	Intracerebral hemorrhage (431)
Coordinate 184	Coordinate 190	Coordinate 199
	Down's syndrome (758.0)	
Pain in joint, shoulder region (719.41)	Congenital anomalies (759.89)	Infantile cerebral palsy (343.9)
Pain in joint, lower leg (719.46)	Tuberous sclerosis (759.5)	Congenital quadriplegia (343.2)
Pain in joint, ankle and foot (719.47)	Anomalies of larynx, trachea, and bronchus (748.3)	Congenital diplegia (343.0)
Pain in joint, multiple sites (719.49)		Quadriplegia (344.00)
Generalized convulsive epilepsy (345.10)	Autosomal deletions (758.39)	Congenital hemiplegia (343.1)
Pain in joint, upper arm (719.42)	Conditions due to anomaly of unspecified chromosome (758.9)	Baclofen 10mg tablet (R)
Cerebral artery occlusion (434.91)		Wheelchair management (P)
MRI, brain (780.59)	Acquired hypothyroidism (244.9)	Tracheostomy status (V44.0)
Other joint derangement (718.81)	Conditions due to chromosome anomalies (758.89)	Paraplegia (344.1)
Fecal occult blood (790.6)	Anomalies of spleen (759.0)	Baclofen 5mg/ml liquid (R)
	Conditions due to autosomal anomalies (758.5)	

5.1 Results

Table 5 shows top ten codes with the largest value in each of the six coordinates of the code embedding space. The coordinate 112 is clearly related to sickle-cell disease and organ transplant. The two are closely related in that sickle cell disease can be treated with bone-marrow transplant. Prograf is a medication used for preventing organ rejection. Coordinate 152 groups medical codes related to sports-related injuries, specifically broken bones. Coordinate 141 is related to brain injuries and hearing loss due to the brain injuries. Neurofibromatosis(NF) is also related to this coordinate because it can cause tumors along the nerves in the brain. Cystic fibrosis(CF) seems to be a weak link in this group as it is only related to NF in the sense that both NF and CF are genetically inherited. Coordinate 184 clearly represents medical codes related to epilepsy. Epilepsy is often accompanied by convulsions, which can cause joint pain. Cerebral artery occlusion is related epilepsy in the sense that epileptic seizures can be a manifestation of cerebral arterial occlusive diseases[11]. Also, both blood in feces and the joint pain can be attributed to Henoch–Schönlein purpura, a disease primarily found in children. Coordinate 190 groups diseases that are caused by congenital chromosome anomalies, especially the autosome. Acquired hypothyroidism seems to be an outlier of this coordinate. Coordinate 199 is strongly related to congenital paralysis. Baclofen is a medication used as a muscle relaxer. Quadraplegia patients can have weakened respiratory function due to impaired abdominal muscles[15], in which case tracheostomy could be required.

We now analyze two visit coordinates: coordinate 50 and 41. Both visit coordinates have the strongest connection to the logistic regression learned for the CRG prediction. For visit coordinate 50, the two strongest code coordinates connected to it are code coordinates 112 and 152. Then naturally, from our analysis above, we can easily see that visit coordinate 50 is strongly activated by sickle-cell disease and sports-related injuries. For visit coordinate 41, code coordinates 141 and 184 have the strongest connection. Again from the analysis above, we can directly infer that visit coordinate 41 can be seen as a patient group consisting of brain damage & hearing loss patients and epilepsy patients. By repeating this process, we can find the code coordinates that are likely to strongly influence the CRG level.

However, finding the influential code coordinates for CRG level can be achieved without analyzing the visit representation if we use Eq.(5). Applying Eq.(5) to the logistic regression weight of the CRG prediction, we learned that code coordinates 190 and 199 are the two strongest influencer of the CRG level. Using the analysis from above, we can naturally conclude that patients suffering from congenital chromosome anomalies or congenital paralysis are most likely to be considered to be in severe states, which is obviously true in any clinical setting.

The results indicate that interpretable visit representations learned by Med2Vec not only improve the prediction accuracy, but also identify the influential clinical concepts.

6. CONCLUSION

In this paper, we proposed Med2Vec, a scalable two layer neural network for learning lower dimensional representations for medical concepts. Med2Vec incorporates both code co-occurence information and visit sequence information of the EHR data which improves the accuracy of both code and visit representations. Throughout several experiments, we successfully demonstrated the superior performance of Med2Vec in two predictive tasks and provided clinical interpretation of the learned representations.

Acknowledgments

This work was supported by the National Science Foundation, award IIS- #1418511 and CCF-#1533768, research partnership between Children's Healthcare of Atlanta and the Georgia Institute of Technology, CDC I-SMILE project, Google Faculty Award, Sutter health, UCB and Samsung Scholarship.

References

[1] Y. Bengio. Learning deep architectures for ai. *Foundations and Trends® in Machine Learning*, 2009.

[2] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *PAMI*, 2013.

[3] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *JMLR*, 2003.

[4] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, et al. Greedy layer-wise training of deep networks. *NIPS*, 2007.

[5] 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.

[6] 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.

[7] R. Chen, H. Su, Y. Zhen, M. Khalilia, D. Hirsch, M. Thompson, T. Davis, Y. Peng, S. Lin, J. Tejedor-Sojo, E. Searles, and J. Sun. Cloud-based predictive modeling system and its application to asthma readmission prediction. In *AMIA*. AMIA, 2015.

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

[9] E. Choi, A. Schuetz, W. F. Stewart, and J. Sun. Medical concept representation learning from electronic health records and its application on heart failure prediction. *arXiv preprint arXiv:1602.03686*, 2016.

[10] Y. Choi, C. Y.-I. Chiu, and D. Sontag. Learning low-dimensional representations of medical concepts. 2016. To be submitted to AMIA CRI.

[11] L. Cocito, E. Favale, and L. Reni. Epileptic seizures in cerebral arterial occlusive disease. *Stroke*, 1982.

[12] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, 2008.

[13] L. De Vine, G. Zuccon, B. Koopman, L. Sitbon, and P. Bruza. Medical semantic similarity with a neural language model. In *KDD*, 2014.

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

[15] J. Forner. Lung volumes and mechanics of breathing in tetraplegics. *Spinal Cord*, 18(4):258–266, 1980.

[16] M. Ghassemi, T. Naumann, F. Doshi-Velez, N. Brimmer, R. Joshi, A. Rumshisky, and P. Szolovits. Unfolding physiological state: Mortality modelling in intensive care units. In *KDD*, 2014.

[17] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 2006.

[18] R. Kiros, R. Zemel, and R. R. Salakhutdinov. A multiplicative model for learning distributed text-based attribute representations. In *NIPS*, 2014.

[19] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler. Skip-thought vectors. In *NIPS*, 2015.

[20] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *ICML*, 2014.

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

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[23] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzell. Learning to diagnose with lstm recurrent neural networks. In *ICLR*, 2016.

[24] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[25] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, 2010.

[26] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[27] J. A. Minarro-Giménez, O. Marín-Alonso, and M. Samwald. Exploring the application of deep learning techniques on medical text corpora. *Studies in health technology and informatics*, 2013.

[28] A. Mnih and G. E. Hinton. A scalable hierarchical distributed language model. In *NIPS*, 2009.

[29] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[30] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. *EMNLP*, 2014.

[31] J. Schluter and S. Bock. Improved musical onset detection with convolutional neural networks. In *ICASSP*, 2014.

[32] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, 2013.

[33] J. Sun, C. D. McNaughton, P. Zhang, A. Perer, A. Gkoulalas-Divanis, J. C. Denny, J. Kirby, T. Lasko, A. Saip, and B. A. Malin. Predicting changes in hypertension control using electronic health records from a chronic disease management program. *JAMIA*, 21, 2014.

[34] J. Sun, F. Wang, J. Hu, and S. Edabollahi. Supervised patient similarity measure of heterogeneous patient records. *KDD Explorations*, 2012.

[35] J. Turian, L. Ratinov, and Y. Bengio. Word representations: a simple and general method for semi-supervised learning. In *ACL*, 2010.

[36] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008.

[37] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR*, 2010.

[38] Y. Wang, K. Ng, R. J. Byrd, J. Hu, S. Ebadollahi, Z. Daar, C. deFilippi, S. R. Steinhubl, and W. F. Stewart. Early detection of heart failure with varying prediction windows by structured and unstructured data in electronic health records. In *EMBC*, 2015.

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