A Neural Architecture for Automated ICD Coding

Pengtao Xie[†*], Haoran Shi[§], Ming Zhang[§] and Eric P. Xing[†]
[†]Petuum Inc, USA
[*]Machine Learning Department, Carnegie Mellon University, USA
[§]School of Electronics Engineering and Computer Science, Peking University, China
{pengtao.xie,eric.xing}@petuum.com
shore@pku.edu.cn, mzhang@net.pku.edu.cn

Abstract

The International Classification of Diseases (ICD) provides a hierarchy of diagnostic codes for classifying diseases. Medical coding – which assigns a subset of ICD codes to a patient visit – is a mandatory process that is crucial for patient care and billing. Manual coding is time-consuming, expensive, and error-prone. In this paper, we build a neural architecture for automated coding. It takes the diagnosis descriptions (DDs) of a patient as inputs and selects the most relevant ICD codes. This architecture contains four major ingredients: (1) tree-of-sequences LSTM encoding of code descriptions (CDs), (2) adversarial learning for reconciling the different writing styles of DDs and CDs, (3) isotonic constraints for incorporating the importance order among the assigned codes, and (4) attentional matching for performing many-to-one and one-to-many mappings from DDs to CDs. We demonstrate the effectiveness of the proposed methods on a clinical datasets with 59K patient visits.

1 Introduction

The International Classification of Diseases (ICD) is a healthcare classification system maintained by the World Health Organization (Organization et al., 1978). It provides a hierarchy of diagnostic codes of diseases, disorders, injuries, signs, symptoms, etc. It is widely used for reporting diseases and health conditions, assisting in medical reimbursement decisions, collecting morbidity and mortality statistics, to name a few.

While ICD codes are important for making clinical and financial decisions, medical coding – which assigns proper ICD codes to a patient visit – is time-consuming, error-prone, and expensive. Medical coders review the diagnosis descriptions written by physicians in the form of textual phrases and sentences, and (if necessary) other information in the electronic health record of a clinical episode, then manually attribute the appropriate ICD codes by following the coding guidelines (O'malley et al., 2005). Several types of errors frequently occur. First, the ICD codes are organized in a hierarchical structure. For a node representing a disease C, the children of this node represent the subtypes of C. In many cases, the difference between disease subtypes is very subtle. It is common that human coders select incorrect subtypes. Second, when writing diagnosis descriptions, physicians often utilize abbreviations and synonyms, which causes ambiguity and imprecision when the coders are matching ICD codes to those descriptions (Sheppard et al., 2008). Third, in many cases, several diagnosis descriptions are closely related and should be mapped to a single ICD code. However, unexperienced coders may code each disease separately. Such errors are called *unbundling*. The cost incurred by coding errors and the financial investment spent on improving coding quality are estimated to be $25 billion per year in the US (Lang, 2007; Farkas and Szarvas, 2008).

To reduce coding errors and cost, we aim at building an ICD coding model which automatically and accurately translates the free-text diagnosis descriptions into ICD codes. To achieve this goal, several technical challenges need to be addressed. First, there exists a hierarchical structure among the ICD codes. This hierarchy can be leveraged to improve coding accuracy. On one hand, if code A and B are both children of C, then it is unlikely to simultaneously assign A and B to a patient. On the other hand, if the distance be-

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pages 1066–1076
Melbourne, Australia, July 15 - 20, 2018.

tween A and B in the code tree is smaller than that between A and C and we know A is the correct code, then B is more likely to be a correct code than C, since codes with smaller distance are more clinically relevant. How to explore this hierarchical structure for better coding is technically demanding. Second, the diagnosis descriptions and the textual descriptions of ICD codes are written in quite different styles even if they refer to the same disease. In particular, the textual description of an ICD code is formally and precisely worded, while diagnosis descriptions are usually written by physicians in an informal and ungrammatical way, with telegraphic phrases, abbreviations, and typos. Third, it is required that the assigned ICD codes are ranked according to their relevance to the patient. How to correctly determine this order is technically nontrivial. Fourth, as stated earlier, there does not necessarily exist an one-to-one mapping between diagnosis descriptions and ICD codes, and human coders should consider the overall health condition when assigning codes. In many cases, two closely related diagnosis descriptions need to be mapped onto a single combination ICD code. On the other hand, physicians may write two health conditions into one diagnosis description which should be mapped onto two ICD codes under such circumstances.

Contributions In this paper, we design a neural architecture to automatically perform ICD coding given the diagnosis descriptions. Specifically, we make the following contributions:

- We propose a tree-of-sequences LSTM architecture to simultaneously capture the hierarchical relationship among codes and the semantics of each code.
- We use an adversarial learning approach to reconcile the heterogeneous writing styles of diagnosis descriptions and ICD code descriptions.
- We use isotonic constraints to preserve the importance order among codes and develop an algorithm based on ADMM and isotonic projection to solve the constrained problem.
- We use an attentional matching mechanism to perform many-to-one and one-to-many mappings between diagnosis descriptions and codes.
- On a clinical datasets with 59K patient visits, we demonstrate the effectiveness of the proposed methods.

The rest of the paper is organized as follows. Section 2 introduces related works. Section 3 and 4 present the dataset and methods. Section 5 gives experimental results. Section 6 presents conclusions and discussions.

2 Related Works

Larkey and Croft (1996) studied the automatic assignment of ICD-9 codes to dictated inpatient discharge summaries, using a combination of three classifiers: k-nearest neighbors, relevance feedback, and Bayesian independence classifiers. This method assigns a single code to each patient visit. However, in clinical practice, each patient is usually assigned with multiple codes. Franz et al. (2000) investigated the automated coding of German-language free-text diagnosis phrases. This approach performs one-to-one mapping between diagnosis descriptions and ICD codes. This is not in accordance with the coding practice where one-to-many and many-to-one mappings widely exist (O'malley et al., 2005). Pestian et al. (2007) studied the assignment of ICD-9 codes to radiology reports. Kavuluru et al. (2013) proposed an unsupervised ensemble approach to automatically perform ICD-9 coding based on textual narratives in electronic health records (EHRs) Kavuluru et al. (2015) developed multi-label classification, feature selection, and learning to rank approaches for ICD-9 code assignment of in-patient visits based on EHRs. Koopman et al. (2015) explored the automatic ICD-10 classification of cancers from free-text death certificates. These methods did not consider the hierarchical relationship or importance order among codes.

The tree LSTM network was first proposed by (Tai et al., 2015) to model the constituent or dependency parse trees of sentences. Teng and Zhang (2016) extended the unidirectional tree LSTM to a bidirectional one. Xie and Xing (2017) proposed a sequence-of-trees LSTM network to model a passage. In this network, a sequential LSTM is used to compose a sequence of tree LSTMs. The tree LSTMs are built on the constituent parse trees of individual sentences and the sequential LSTM is built on the sequence of sentences. Our proposed tree-of-sequences LSTM network differs from the previous works in twofold. First, it is applied to a code tree to capture the hierarchical relationship among codes. Second, it uses a tree LSTM to compose a hierarchy

Diagnosis Descriptions
1. Prematurity at 35 4/7 weeks gestation
2. Twin number two of twin gestation
3. Respiratory distress secondary to transient tachypnea of the newborn
4. Suspicion for sepsis ruled out
Assigned ICD Codes
1. V31.00 (Twin birth, mate liveborn, born in hospital, delivered without mention of cesarean section)
2. 765.18 (Other preterm infants, 2,000-2,499 grams)
3. 775.6 (Neonatal hypoglycemia)
4. 770.6 (Transitory tachypnea of newborn)
5. V29.0 (Observation for suspected infectious condition)
6. V05.3 (Need for prophylactic vaccination and inoculation against viral hepatitis)

Table 1: The diagnosis descriptions of a patient visit and the assigned ICD codes. Inside the parentheses are the descriptions of the codes. The codes are ranked according to descending importance.

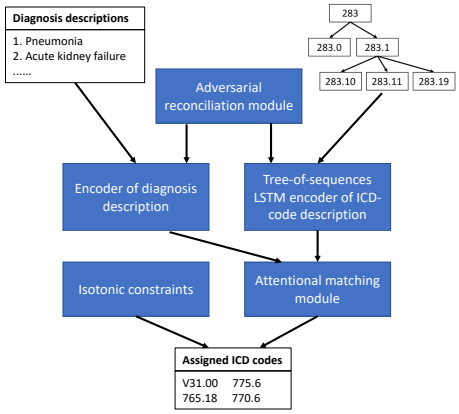

Figure 1: Architecture of the ICD Coding Model

of sequential LSTMs.

Adversarial learning (Goodfellow et al., 2014) has been widely applied to image generation (Goodfellow et al., 2014), domain adaptation (Ganin and Lempitsky, 2015), feature learning (Donahue et al., 2016), text generation (Yu et al., 2017), to name a few. In this paper, we use adversarial learning for mitigating the discrepancy among the writing styles of a pair of sentences.

The attention mechanism was widely used in machine translation (Bahdanau et al., 2014), image captioning (Xu et al., 2015), reading comprehension (Seo et al., 2016), text classification (Yang et al., 2016), etc. In this work, we compute attention between sentences to perform many-to-one and one-to-many mappings.

3 Dataset and Preprocessing

We performed the study on the publicly available MIMIC-III dataset (Johnson et al., 2016), which contains de-identified electronic health records (EHRs) of 58,976 patient visits in the Beth Israel Deaconess Medical Center from 2001 to 2012. Each EHR has a clinical note called discharge summary, which contains multiple sections of information, such as 'discharge diagnosis', 'past medical history', etc. From the 'discharge diagnosis' and 'final diagnosis' sections, we extracted the diagnosis descriptions (DDs) written by physicians. Each DD is a short phrase or a sentence, articulating a certain disease or condition. Medical coders perform ICD coding mainly based on DDs. Following such a practice, in this paper, we set the inputs of the automated coding model to be the DDs while acknowledging that other information in the EHRs is also valuable and is referred to by coders for code assignment. For simplicity, we leave the incorporation of non-DD information to future study.

Each patient visit is assigned with a list of ICD codes, ranked in descending order of importance and relevance. For each visit, the number of codes is usually not equal to the number of diagnosis descriptions. These ground-truth codes serve as the labels to train our coding model. The entire dataset contains 6,984 unique codes, each of which has a textual description, describing a disease, symptom, or condition. The codes are organized into a hierarchy where the top-level codes correspond to general diseases while the bottom-level ones represent specific diseases. In the code tree, children of a node represent subtypes of a disease. Table 1 shows the DDs and codes of an exemplar patient.

4 Methods

In this section, we present a neural architecture for ICD coding.

4.1 Overview

Figure 1 shows the overview of our approach. The proposed ICD coding model consists of five modules. The model takes the ICD-code tree and diagnosis descriptions (DDs) of a patient as inputs and assigns a set of ICD codes to the patient. The encoder of DDs generates a latent representation vector for a DD. The encoder of ICD codes is a tree-of-sequences long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) network. It takes the textual descriptions of the ICD codes and their hierarchical structure as in-

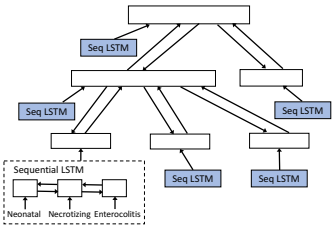

Figure 2: Tree-of-Sequences LSTM

puts and produces a latent representation for each code. The representation aims at simultaneously capturing the semantics of each code and the hierarchical relationship among codes. By incorporating the code hierarchy, the model can avoid selecting codes that are subtypes of the same disease and promote the selection of codes that are clinically correlated. The writing styles of DDs and code descriptions (CDs) are largely different, which makes the matching between a DD and a CD error-prone. To address this issue, we develop an adversarial learning approach to reconcile the writing styles. On top of the latent representation vectors of the descriptions, we build a discriminative network to distinguish which ones are DDs and which are CDs. The encoders of DDs and CDs try to make such a discrimination impossible. By doing this, the learned representations are independent of the writing styles and facilitate more accurate matching. The representations of DDs and CDs are fed into an attentional matching module to perform code assignment. This attentional mechanism allows multiple DDs to be matched to a single code and allows a single DD to be matched to multiple codes. During training, we incorporate the order of importance among codes as isotonic constraints. These constraints regulate the model's weight parameters so that codes with higher importance are given larger prediction scores.

4.2 Tree-of-Sequences LSTM Encoder

This section introduces the encoder of ICD codes. Each code has a description (a sequence of words) that tells the semantics of this code. We use a sequential LSTM (SLSTM) (Hochreiter and Schmidhuber, 1997) to encode this description. To capture the hierarchical relationship among codes, we build a tree LSTM (TLSTM) (Tai et al., 2015) along the code tree. At each TLSTM node, the input vector is the latent representation generated by the SLSTM. Combining these two types of LSTMs together, we obtain a tree-of-sequences LSTM network (Figure 2).

Sequential LSTM A sequential LSTM (SLSTM) (Hochreiter and Schmidhuber, 1997) network is a special type of recurrent neural network that (1) learns the latent representation (which usually reflects certain semantic information) of words, and (2) models the sequential structure among words. In the word sequence, each word t is allocated with an SLSTM unit, which consists of the following components: an input gate $\mathbf{i}_t$, a forget gate $\mathbf{f}_t$, an output gate $\mathbf{o}_t$, a memory cell $\mathbf{c}_t$, and a hidden state $\mathbf{s}_t$. These components (vectors) are computed as follows:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}^{(i)}\mathbf{s}_{t-1} + \mathbf{U}^{(i)}\mathbf{x}_t + \mathbf{b}^{(i)}) \\
\mathbf{f}_t &= \sigma(\mathbf{W}^{(f)}\mathbf{s}_{t-1} + \mathbf{U}^{(f)}\mathbf{x}_t + \mathbf{b}^{(f)}) \\
\mathbf{o}_t &= \sigma(\mathbf{W}^{(o)}\mathbf{s}_{t-1} + \mathbf{U}^{(o)}\mathbf{x}_t + \mathbf{b}^{(o)}) \\
\mathbf{c}_t &= \mathbf{i}_t \odot \tanh(\mathbf{W}^{(c)}\mathbf{s}_{t-1} + \mathbf{U}^{(c)}\mathbf{x}_t + \mathbf{b}^{(c)}) \\
&\quad + \mathbf{f}_t \odot \mathbf{c}_{t-1} \\
\mathbf{s}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned} \tag{1}$$

where $\mathbf{x}_t$ is the embedding vector of word t. $\mathbf{W}$, $\mathbf{U}$ are component-specific weight matrices and $\mathbf{b}$ are bias vectors.

Tree-of-sequences LSTM We use a bidirectional tree LSTM (TLSTM) (Tai et al., 2015; Xie and Xing, 2017) to capture the hierarchical relationships among codes. The inputs of this LSTM include the code hierarchy and hidden states of individual codes produced by the SLSTMs. It consists of a bottom-up TLSTM and a top-down TLSTM, which produce two hidden states $\mathbf{h}_\uparrow$ and $\mathbf{h}_\downarrow$ at each node in the tree.

In the bottom-up TLSTM, an internal node (representing a code C, having M children) is comprised of these components: an input gate $\mathbf{i}_\uparrow$, an output gate $\mathbf{o}_\uparrow$, a memory cell $\mathbf{c}_\uparrow$, a hidden state $\mathbf{h}_\uparrow$ and M child-specific forget gates $\{\mathbf{f}_\uparrow^{(m)}\}_{m=1}^M$ where $\mathbf{f}_\uparrow^{(m)}$ corresponds to the m-th child. The transition equations among components are:

$$\begin{aligned}
\mathbf{i}_\uparrow &= \sigma(\textstyle\sum_{m=1}^M \mathbf{W}_\uparrow^{(i,m)}\mathbf{h}_\uparrow^{(m)} + \mathbf{U}^{(i)}\mathbf{s} + \mathbf{b}_\uparrow^{(i)}) \\
\forall m, \mathbf{f}_\uparrow^{(m)} &= \sigma(\mathbf{W}_\uparrow^{(f,m)}\mathbf{h}_\uparrow^{(m)} + \mathbf{U}^{(f,m)}\mathbf{s} + \mathbf{b}_\uparrow^{(f,m)}) \\
\mathbf{o}_\uparrow &= \sigma(\textstyle\sum_{m=1}^M \mathbf{W}_\uparrow^{(o,m)}\mathbf{h}_\uparrow^{(m)} + \mathbf{U}^{(o)}\mathbf{s} + \mathbf{b}_\uparrow^{(o)}) \\
\mathbf{u}_\uparrow &= \tanh(\textstyle\sum_{m=1}^M \mathbf{W}_\uparrow^{(u,m)}\mathbf{h}_\uparrow^{(m)} + \mathbf{U}^{(u)}\mathbf{s} + \mathbf{b}_\uparrow^{(u)}) \\
\mathbf{c}_\uparrow &= \mathbf{i}_\uparrow \odot \mathbf{u}_\uparrow + \textstyle\sum_{m=1}^M \mathbf{f}_\uparrow^{(m)} \odot \mathbf{c}_\uparrow^{(m)} \\
\mathbf{h}_\uparrow &= \mathbf{o}_\uparrow \odot \tanh(\mathbf{c}_\uparrow)
\end{aligned} \tag{2}$$

where $\mathbf{s}$ is the SLSTM hidden state that encodes the description of code C; $\{\mathbf{h}_{\uparrow}^{(m)}\}_{m=1}^{M}$ and $\{\mathbf{c}_{\uparrow}^{(m)}\}_{m=1}^{M}$ are the bottom-up TLSTM hidden states and memory cells of the children. $\mathbf{W}$, $\mathbf{U}$, $\mathbf{b}$ are component-specific weight matrices and bias vectors. For a leaf node having no children, its only input is the SLSTM hidden state $\mathbf{s}$ and no forget gates are needed.

In the top-down TLSTM, for a non-root node, it has such components: an input gate $\mathbf{i}_{\downarrow}$, a forget gate $\mathbf{f}_{\downarrow}$, an output gate $\mathbf{o}_{\downarrow}$, a memory cell $\mathbf{c}_{\downarrow}$ and a hidden state $\mathbf{h}_{\downarrow}$. The transition equations are:

$$\begin{aligned}
\mathbf{i}_{\downarrow} &= \sigma(\mathbf{W}_{\downarrow}^{(i)}\mathbf{h}_{\downarrow}^{(p)} + \mathbf{b}_{\downarrow}^{(i)}) \\
\mathbf{f}_{\downarrow} &= \sigma(\mathbf{W}_{\downarrow}^{(f)}\mathbf{h}_{\downarrow}^{(p)} + \mathbf{b}_{\downarrow}^{(f)}) \\
\mathbf{o}_{\downarrow} &= \sigma(\mathbf{W}_{\downarrow}^{(o)}\mathbf{h}_{\downarrow}^{(p)} + \mathbf{b}_{\downarrow}^{(o)}) \\
\mathbf{u}_{\downarrow} &= \tanh(\mathbf{W}_{\downarrow}^{(u)}\mathbf{h}_{\downarrow}^{(p)} + \mathbf{b}_{\downarrow}^{(u)}) \\
\mathbf{c}_{\downarrow} &= \mathbf{i}_{\downarrow} \odot \mathbf{u}_{\downarrow} + \mathbf{f}_{\downarrow} \odot \mathbf{c}_{\downarrow}^{(p)} \\
\mathbf{h}_{\downarrow} &= \mathbf{o}_{\downarrow} \odot \tanh(\mathbf{c}_{\downarrow})
\end{aligned} \tag{3}$$

where $\mathbf{h}_{\downarrow}^{(p)}$ and $\mathbf{c}_{\downarrow}^{(p)}$ are the top-down TLSTM hidden state and memory cell of the parent of this node. For the root node which has no parent, $\mathbf{h}_{\downarrow}$ cannot be computed using the above equations. Instead, we set $\mathbf{h}_{\downarrow}$ to $\mathbf{h}_{\uparrow}$ (the bottom-up TLSTM hidden state generated at the root node). $\mathbf{h}_{\uparrow}$ captures the semantics of all codes in this hierarchy, which is then propagated downwards to each individual code via the top-down TLSTM dynamics.

We concatenate the hidden states of the two directions to obtain the bidirectional TLSTM encoding of each code $\mathbf{h} = [\mathbf{h}_{\uparrow}; \mathbf{h}_{\downarrow}]$. The bottom-up TLSTM composes the semantics of children (representing sub-diseases) and merge them into the current node, which hence captures child-to-parent relationship. The top-down TLSTM makes each node inherit the semantics of its parent, which captures parent-to-child relation. As a result, the hierarchical relationship among codes are encoded in the hidden states.

For the diagnosis descriptions of a patient, we use an SLSTM network to encode each description individually. The weight parameters of this SLSTM are tied with those of the SLSTM used for encoding code descriptions.

4.3 Attentional Matching

Next, we introduce how to map the DDs to codes. We denote the hidden representations of DDs and codes as $\{\mathbf{h}_m\}_{m=1}^{M}$ and $\{\mathbf{u}_n\}_{n=1}^{N}$ respectively, where M is the number of DDs of one patient and N is the total number of codes in the dataset. The mapping from DDs to codes is not one-to-one. In many cases, a code is assigned only when a certain combination of K ($1 < K \leq M$) diseases simultaneously appear within the M DDs and the value of K depends on this code. Among the K diseases, their importance of determining the assignment of this code is different. For the rest $M - K$ DDs, we can consider their importance score to be zero. We use a soft-attention mechanism (Bahdanau et al., 2014) to calculate these importance scores. For a code $\mathbf{u}_n$, the importance of a DD $\mathbf{h}_m$ to $\mathbf{u}_n$ is calculated as $a_{nm} = \mathbf{u}_n^{\top}\mathbf{h}_m$. We normalize the scores $\{a_{nm}\}_{m=1}^{M}$ of all DDs into a probabilistic simplex using the softmax operation: $\tilde{a}_{nm} = \exp(a_{nm}) / \sum_{l=1}^{M} \exp(a_{nl})$. Given these normalized importance scores $\{\tilde{a}_{nm}\}_{m=1}^{M}$, we use them to weight the representations of DDs and get a single attentional vector of the M DDs: $\widehat{\mathbf{h}}_n = \sum_{m=1}^{M} \tilde{a}_{nm}\mathbf{h}_m$. Then we concatenate $\widehat{\mathbf{h}}_n$ and $\mathbf{u}_n$, and use a linear classifier to predict the probability that code n should be assigned: $p_n = \text{sigmoid}(\mathbf{w}_n^{\top}[\widehat{\mathbf{h}}_n; \mathbf{u}_n] + b_n)$, where the coefficients $\mathbf{w}_n$ and bias b_n are specific to code n.

We train the weight parameters Θ of the proposed model using the data of L patient visits. Θ includes the sequential LSTM weights $\mathbf{W}_s$, tree LSTM weights $\mathbf{W}_t$ and weights $\mathbf{W}_p$ in the final prediction layer. Let $c^{(l)} \in \mathbb{R}^N$ be a binary vector where $c_n^{(l)} = 1$ if the n-th code is assigned to this patient and $c_n^{(l)} = 0$ if otherwise. Θ can be learned by minimizing the following prediction loss:

$$\min_{\Theta} \quad \mathcal{L}_{\text{pred}}(\Theta) = \sum_{l=1}^{L}\sum_{n=1}^{N} \text{CE}(p_n^{(l)}, c_n^{(l)}) \tag{4}$$

where $p_n^{(l)}$ is the predicted probability that code n is assigned to patient visit l and $p_n^{(l)}$ is a function of Θ. $\text{CE}(\cdot,\cdot)$ is the cross-entropy loss.

4.4 Adversarial Reconciliation of Writing Styles

We use an adversarial learning (Goodfellow et al., 2014) approach to reconcile the different writing styles of diagnosis descriptions (DDs) and code descriptions (CDs). The basic idea is: after encoded, if a description cannot be discerned to be a DD or a CD, then the difference in their writing styles is eliminated. We build a discriminative network which takes the encoding vector of a description as input and tries to identify it as a DD

or CD. The encoders of DDs and CDs adjust their weight parameters so that such a discrimination is difficult to be achieved by the discriminative network. Consider all the descriptions $\{t_r, y_r\}_{r=1}^R$ where t_r is a description and y_r is a binary label. $y_r = 1$ if t_r is a DD and $y_r = 0$ if otherwise. Let $f(t_r; \mathbf{W}_s)$ denote the sequential LSTM (SLSTM) encoder parameterized by $\mathbf{W}_s$. This encoder is shared by the DDs and CDs. Note that for CDs, a tree LSTM is further applied on top of the encodings produced by the SLSTM. We use the SLSTM encoding vectors of CDs as the input of the discriminative network rather than using the TLSTM encodings since the latter are irrelevant to writing styles. Let $g(f(t_r; \mathbf{W}_s); \mathbf{W}_d)$ denote the discriminative network parameterized by $\mathbf{W}_d$. It takes the encoding vector $f(t_r; \mathbf{W}_s)$ as input and produces the probability that t_r is a DD. Adversarial learning is performed by solving this problem:

$$\max_{\mathbf{W}_s} \min_{\mathbf{W}_d} \mathcal{L}_{\text{adv}} = \sum_{r=1}^{R} \text{CE}(g(f(t_r; \mathbf{W}_s); \mathbf{W}_d), y_r) \tag{5}$$

The discriminative network tries to differentiate DDs from CDs by minimizing this classification loss while the encoder maximizes this loss so that DDs and CDs are not distinguishable.

4.5 Isotonic Constraints

Next, we incorporate the importance order among ICD codes. For the $D^{(l)}$ codes assigned to patient l, without loss of generality, we assume the order is $1 \succ 2 \cdots \succ D^{(l)}$ (the order is given by human coders as ground-truth in the MIMIC-III dataset). We use the predicted probability p_i ($1 \leq i \leq D^{(l)}$) defined in Section 4.3 to characterize the importance of code i. To incorporate the order, we impose an isotonic constraint on the probabilities: $p_1^{(l)} \succ p_2^{(l)} \cdots \succ p_{D^{(l)}}^{(l)}$, and solve the following problem:

$$\begin{array}{ll} \min_{\Theta} & \mathcal{L}_{\text{pred}}(\Theta) + \max_{\mathbf{W}_d}(-\lambda \mathcal{L}_{\text{adv}}(\mathbf{W}_s, \mathbf{W}_d)) \\ s.t. & p_1^{(l)} \succ p_2^{(l)} \cdots \succ p_{D^{(l)}}^{(l)} \\ & \forall l = 1, \cdots, L \end{array} \tag{6}$$

where the probabilities $p_i^{(l)}$ are functions of Θ and λ is a tradeoff parameter.

We develop an algorithm based on the alternating direction method of multiplier (ADMM) (Boyd et al., 2011) to solve the problem defined in Eq.(6). Let $\mathbf{p}^{(l)}$ be a $|D^{(l)}|$-dimensional vector where the i-th element is $p_i^{(l)}$. We first write the problem into an equivalent form

$$\begin{array}{ll} \min_{\Theta} & \mathcal{L}_{\text{pred}}(\Theta) + \max_{\mathbf{W}_d}(-\lambda \mathcal{L}_{\text{adv}}(\mathbf{W}_s, \mathbf{W}_d)) \\ s.t. & \mathbf{p}^{(l)} = \mathbf{q}^{(l)} \\ & q_1^{(l)} \succ q_2^{(l)} \cdots \succ q_{|D^{(l)}|}^{(l)} \\ & \forall l = 1, \cdots, L \end{array} \tag{7}$$

Then we write down the augmented Lagrangian

$$\begin{array}{ll} \min\limits_{\Theta, \mathbf{q}, \mathbf{v}} & \mathcal{L}_{\text{pred}}(\Theta) + \max_{\mathbf{W}_d}(-\lambda \mathcal{L}_{\text{adv}}(\mathbf{W}_s, \mathbf{W}_d)) \\ & + \langle \mathbf{p}^{(l)} - \mathbf{q}^{(l)}, \mathbf{v}^{(l)} \rangle + \frac{\rho}{2} \|\mathbf{p}^{(l)} - \mathbf{q}^{(l)}\|_2^2 \\ s.t. & q_1^{(l)} \succ q_2^{(l)} \cdots \succ q_{|D^{(l)}|}^{(l)} \\ & \forall l = 1, \cdots, L \end{array} \tag{8}$$

We solve this problem by alternating between $\{\mathbf{p}^{(l)}\}_{l=1}^L$, $\{\mathbf{q}^{(l)}\}_{l=1}^L$ and $\{\mathbf{v}^{(l)}\}_{l=1}^L$ The sub-problem defined over $\mathbf{q}^{(l)}$ is

$$\begin{array}{ll} \min_{\mathbf{q}^{(l)}} & -\langle \mathbf{q}^{(l)}, \mathbf{v}^{(l)} \rangle + \frac{\rho}{2} \|\mathbf{p}^{(l)} - \mathbf{q}^{(l)}\|_2^2 \\ s.t. & q_1^{(l)} \succ q_2^{(l)} \cdots \succ q_{|D^{(l)}|}^{(l)} \end{array} \tag{9}$$

which is an isotonic projection problem and can be solved via the algorithm proposed in (Yu and Xing, 2016). With $\{\mathbf{q}^{(l)}\}_{l=1}^L$ and $\{\mathbf{v}^{(l)}\}_{l=1}^L$ fixed, the sub-problem is $\min_{\Theta} \mathcal{L}_{\text{pred}}(\Theta) + \max_{\mathbf{W}_d}(-\lambda \mathcal{L}_{\text{adv}}(\mathbf{W}_s, \mathbf{W}_d))$ which can be solved using stochastic gradient descent (SGD). The update of $\mathbf{v}^{(l)}$ is simple: $\mathbf{v}^{(l)} = \mathbf{v}^{(l)} + \rho(\mathbf{p}^{(l)} - \mathbf{q}^{(l)})$.

5 Experiments

In this section, we present experiment results.

5.1 Experimental Settings

Out of the 6,984 unique codes, we selected 2,833 codes that have the top frequencies to perform the study. We split the data into a train/validation/test dataset with 40k/7k/12k patient visits respectively. The hyperparameters were tuned on the validation set. The SLSTMs were bidirectional and dropout with 0.5 probability (Srivastava et al., 2014) was used. The size of hidden states in all LSTMs was set to 100. The word embeddings were trained on the fly and their dimension was set to 200. The tradeoff parameter λ was set to 0.1. The parameter ρ in the ADMM algorithm was set to 1. In the SGD algorithm for solving $\min_{\Theta} \mathcal{L}_{\text{pred}}(\Theta) + \max_{\mathbf{W}_d}(-\lambda \mathcal{L}_{\text{adv}}(\mathbf{W}_s, \mathbf{W}_d))$, we used the ADAM (Kingma and Ba, 2014) optimizer with an initial learning rate 0.001 and a mini-batch size 20. Sensitivity (true positive rate) and

specificity (true negative rate) were used to evaluate the code assignment performance. We calculated these two scores for each individual code on the test set, then took a weighted (proportional to codes' frequencies) average across all codes. To evaluate the ranking performance of codes, we used normalized discounted cumulative gain (NDCG) (Järvelin and Kekäläinen, 2002).

5.2 Ablation Study

We perform ablation study to verify the effectiveness of each module in our model. To evaluate module X, we remove it from the model without changing other modules and denote such a baseline by No-X. The comparisons of No-X with the full model are given in Table 2.

	Sensitivity	Specificity
(Larkey and Croft, 1996)	0.15	0.17
(Franz et al., 2000)	0.19	0.21
(Pestian et al., 2007)	0.12	0.21
(Kavuluru et al., 2013)	0.09	0.11
(Kavuluru et al., 2015)	0.21	0.25
(Koopman et al., 2015)	0.18	0.20
LET	0.23	0.29
HierNet	0.26	0.30
HybridNet	0.25	0.31
BranchNet	0.25	0.29
No-TLSTM	0.23	0.28
Bottom-up TLSTM	0.27	0.31
No-AL	0.26	0.31
No-IC	0.24	0.29
No-AM	0.27	0.29
Our full model	**0.29**	**0.33**

Table 2: Sensitivity and Specificity on the Test Set

Tree-of-sequences LSTM To evaluate this module, we compared with the two configurations: (1) *No-TLSTM*, which removes the tree LSTM and directly uses the hidden states produced by the sequential LSTM as final representations of codes; (2) *Bottom-up TLSTM*, which removes the hidden states generated by the top-down TLSTM. In addition, we compared with four hierarchical classification baselines including (1) hierarchical network (HierNet) (Yan et al., 2015), (2) HybridNet (Hou et al., 2017), (3) branch network (BranchNet) (Zhu and Bain, 2017), (4) label embedding tree (LET) (Bengio et al., 2010), by using them to replace the bidirectional tree LSTM while keeping other modules untouched. Table 2 shows the average sensitivity and specificity scores achieved by these methods on the test set. We make the following observations. First, removing tree LSTM largely degrades performance: the sensitivity and specificity of No-TLSTM is 0.23 and 0.28 respectively while our full model (which uses bidirectional TLSTM) achieves 0.29 and 0.33 respectively. The reason is No-TLSTM ignores the hierarchical relationship among codes. Second, bottom-up tree LSTM alone performs less well than bidirectional tree LSTM. This demonstrates the necessity of the top-down TLSTM, which ensures every two codes are connected by directed paths and can more expressively capture code-relations in the hierarchy. Third, our method outperforms the four baselines. The possible reason is our method directly builds codes' hierarchical relationship into their representations while the baselines perform representation-learning and relationship-capturing separately.

Next, we present some qualitative results. For a patient (admission ID 147798) having a DD 'E Coli urinary tract infection', without using tree LSTM, two sibling codes 585.2 (chronic kidney disease, stage II (mild)) – which is the ground-truth – and 585.4 (chronic kidney disease, stage IV (severe)) are simultaneously assigned possibly because their textual descriptions are very similar (only differ in the level of severity). This is incorrect because 585.2 and 585.4 are the children of 585 (chronic kidney disease) and the severity level of this disease cannot simultaneously be mild and severe. After tree LSTM is added, the false prediction of 585.4 is eliminated, which demonstrates the effectiveness of tree LSTM in incorporating one constraint induced by the code hierarchy: among the nodes sharing the same parent, only one should be selected.

For patient 197205, No-TLSTM assigns the following codes: 462 (subacute sclerosing panencephalitis), 790.29 (other abnormal glucose), 799.9 (unspecified viral infection), and 285.21 (anemia in chronic kidney disease). Among these codes, the first three are ground-truth and the fourth one is incorrect (the ground-truth is 401.9 (unspecified essential hypertension)). Adding tree LSTM fixes this error. The average distance between 401.9 and the rest of ground-truth codes is 6.2. For the incorrectly assigned code 285.21, such a distance is 7.9. This demonstrates that tree LSTM is able to capture another constraint imposed by the hierarchy: codes with smaller tree-distance are more likely to be assigned together.

Position	2	4	6	8
No-IC	0.27	0.26	0.23	0.20
IC	0.32	0.29	0.27	0.23

Table 3: Comparison of NDCG Scores in the Ablation Study of Isotonic Constraints.

Adversarial learning To evaluate the efficacy of adversarial learning (AL), we remove it from the full model and refer to this baseline as No-AL. Specifically, in Eq.(6), the loss term $\max_{\mathbf{W}_d}(-\mathcal{L}_{\text{adv}}(\mathbf{W}_s, \mathbf{W}_d))$ is taken away. Table 2 shows the results, from which we observe that after AL is removed, the sensitivity and specificity are dropped from 0.29 and 0.33 to 0.26 and 0.31 respectively. No-AL does not reconcile different writing styles of diagnosis descriptions (DDs) and code descriptions (CDs). As a result, a DD and a CD that have similar semantics may be mismatched because their writing styles are different. For example, a patient (admission ID 147583) has a DD 'h/o DVT on anticoagulation', which contains abbreviation DVT (deep vein thrombosis). Due to the presence of this abbreviation, it is difficult to assign a proper code to this DD since the textual descriptions of codes do not contain abbreviations. With adversarial learning, our model can correctly map this DD to a ground-truth code: 443.9 (peripheral vascular disease, unspecified). Without AL, this code is not selected. As another example, a DD 'coronary artery disease, STEMI, s/p 2 stents placed in RCA' was given to patient 148532. This DD is written informally and ungrammatically, and contains too much detailed information, e.g., 's/p 2 stents placed in RCA'. Such a writing style is quite different from that of CDs. With AL, our model successfully matches this DD to a ground-truth code: 414.01 (coronary atherosclerosis of native coronary artery). On the contrary, No-AL fails to achieve this.

Isotonic constraint (IC) To evaluate this ingredient, we remove the ICs from Eq.(6) during training and denote this baseline as No-IC. We use NDCG to measure the ranking performance, which is calculated in the following way. Consider a testing patient-visit l where the ground-truth ICD codes are $\mathcal{M}^{(l)}$. For any code c, we define the relevance score of c to l as 0 if $c \notin \mathcal{M}^{(l)}$ and as $|\mathcal{M}^{(l)}| - r(c)$ if otherwise, where $r(c)$ is the ground-truth rank of c in $\mathcal{M}^{(l)}$. We rank codes in descending order of their corresponding prediction probabilities and obtain the predicted rank for each code. We calculate the NDCG scores at position 2, 4, 6, 8 based on the relevance scores and predicted ranks, which are shown in Table 3. As can be seen, using IC achieves much higher NDCG than No-IC, which demonstrates the effectiveness of IC in capturing the importance order among codes.

We also evaluate how IC affects the sensitivity and specificity of code assignment. As can be seen from Table 2, No-IC degrades the two scores from 0.29 and 0.33 to 0.24 and 0.29 respectively, which indicates that IC is helpful in training a model that can more correctly assign codes. This is because IC encourages codes that are highly relevant to the patients to be ranked at top positions, which prevents the selection of irrelevant codes.

Attentional matching (AM) In the evaluation of this module, we compare with a baseline – No-AM, which performs an unweighted average of the M DDs: $\widehat{\mathbf{h}}_n = \frac{1}{M}\sum_{m=1}^{M}\mathbf{h}_m$, concatenates $\widehat{\mathbf{h}}_n$ with $\mathbf{u}_n$ and feeds the concatenated vector into the final prediction layer. From Table 2, we can see our full model (with AM) outperforms No-AM, which demonstrates the effectiveness of attentional matching. In determining whether a code should be assigned, different DDs have different importance weights. No-AM ignores such weights, therefore performing less well.

AM can correctly perform many-to-one mapping from multiple DDs to a CD. For example, patient 190236 was given two DDs: 'renal insufficiency' and 'acute renal failure'. AM maps them to a combined ICD code: 403.91 (hypertensive chronic kidney disease, unspecified, with chronic kidney disease stage V or end stage renal disease), which is in the ground-truth provided by medical coders. On the contrary, No-AM fails to assign this code. On the other hand, AM is able to correctly map a DD to multiple CDs. For example, a DD 'congestive heart failure, diastolic' was given to patient 140851. AM successfully maps this DD to two codes: (1) 428.0 (congestive heart failure, unspecified); (2) 428.30 (diastolic heart failure, unspecified). Without AM, this DD is mapped only to 428.0.

5.3 Holistic Comparison with Other Baselines

In addition to evaluating the four modules individually, we also compared our full model with four other baselines proposed by (Larkey and Croft,

1996; Franz et al., 2000; Pestian et al., 2007; Kavuluru et al., 2013, 2015; Koopman et al., 2015) for ICD coding. Table 2 shows the results. As can be seen, our approach achieves much better sensitivity and specificity scores. The reason that our model works better is two-fold. First, our model is based on deep neural network, which has arguably better modeling power than linear methods used in the baselines. Second, our model is able to capture the hierarchical relationship and importance order among codes, can alleviate the discrepancy in writing styles and allows flexible many-to-one and one-to-many mappings from DDs to CDs. These merits are not possessed by the baselines.

6 Conclusions and Discussions

In this paper, we build a neural network model for automated ICD coding. Evaluations on the MIMIC-III dataset demonstrate the following. First, the tree-of-sequences LSTM network effectively discourages the co-selection of sibling codes and promotes the co-assignment of clinically-relevant codes. Adversarial learning improves the matching accuracy by alleviating the discrepancy among the writing styles of DDs and CDs. Third, isotonic constraints promote the correct ranking of codes. Fourth, the attentional matching mechanism is able to perform many-to-one and one-to-many mappings.

In the coding practice of human coders, in addition to the diagnosis descriptions, other information contained in nursing notes, lab values, and medical procedures are also leveraged for code assignment. We have initiated preliminary investigation along this line and added two new input sources: (1) the rest of discharge summary and (2) lab values. The sensitivity is improved from 0.29 to 0.32 and the specificity is improved from 0.33 to 0.35. A full study is ongoing.

At present, the major limitations of this work include: (1) it does not perform well on infrequent codes; (2) it is less capable of dealing with abbreviations. We will address these two issues in future by investigating diversity-promoting regularization (Xie et al., 2017) and leveraging an external knowledge base that maps medical abbreviations into their full names.

The proposed methods can be applied to other tasks in NLP. The tree-of-sequences model can be applied for ontology annotation. It takes the textual descriptions of concepts in the ontology and their hierarchical structure as inputs and produces a latent representation for each concept. The representations can simultaneously capture the semantics of codes and their relationships. The proposed adversarial reconciliation of writing styles and attentional matching can be applied for knowledge mapping or entity linking. For example, in tweets, we can use the method to map an informally written mention 'nbcbightlynews' to a canonical entity 'NBC Nightly News' in the knowledge base.

Acknowledgements

We would like to thank the anonymous reviewers for their very constructive and helpful comments and suggestions. Pengtao Xie and Eric P. Xing are supported by National Institutes of Health P30DA035778, Pennsylvania Department of Health BD4BH4100070287, and National Science Foundation IIS1617583.

References

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

Samy Bengio, Jason Weston, and David Grangier. 2010. Label embedding trees for large multi-class tasks. In *NIPS*.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. 2016. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*.

Richárd Farkas and György Szarvas. 2008. Automatic construction of rule-based icd-9-cm coding systems. *BMC bioinformatics*, 9(3):S10.

Pius Franz, Albrecht Zaiss, Stefan Schulz, Udo Hahn, and Rüdiger Klar. 2000. Automated coding of diagnoses–three methods compared. In *Proceedings of the AMIA Symposium*, page 250. American Medical Informatics Association.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron

Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

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

Saihui Hou, Yushan Feng, and Zilei Wang. 2017. Vegfru: A domain-specific dataset for fine-grained visual categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 541–549.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3.

Ramakanth Kavuluru, Sifei Han, and Daniel Harris. 2013. Unsupervised extraction of diagnosis codes from emrs using knowledge-based and extractive text summarization techniques. In *Canadian conference on artificial intelligence*, pages 77–88. Springer.

Ramakanth Kavuluru, Anthony Rios, and Yuan Lu. 2015. An empirical evaluation of supervised learning approaches in assigning diagnosis codes to electronic medical records. *Artificial intelligence in medicine*, 65(2):155–166.

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

Bevan Koopman, Guido Zuccon, Anthony Nguyen, Anton Bergheim, and Narelle Grayson. 2015. Automatic icd-10 classification of cancers from free-text death certificates. *International journal of medical informatics*, 84(11):956–965.

Dee Lang. 2007. Consultant report-natural language processing in the health care industry. *Cincinnati Children's Hospital Medical Center, Winter*.

Leah S Larkey and W Bruce Croft. 1996. Combining classifiers in text categorization. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 289–297. ACM.

Kimberly J O'malley, Karon F Cook, Matt D Price, Kimberly Raiford Wildes, John F Hurdle, and Carol M Ashton. 2005. Measuring diagnoses: Icd code accuracy. *Health services research*, 40(5p2):1620–1639.

World Health Organization et al. 1978. International classification of diseases:[9th] ninth revision, basic tabulation list with alphabetic index. *World Health Organization*.

John P Pestian, Christopher Brew, Paweł Matykiewicz, Dj J Hovermale, Neil Johnson, K Bretonnel Cohen, and Włodzisław Duch. 2007. A shared task involving multi-label classification of clinical free text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, pages 97–104. Association for Computational Linguistics.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Joanna E Sheppard, Laura CE Weidner, Saher Zakai, Simon Fountain-Polley, and Judith Williams. 2008. Ambiguous abbreviations: an audit of abbreviations in paediatric note keeping. *Archives of disease in childhood*, 93(3):204–206.

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

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Zhiyang Teng and Yue Zhang. 2016. Bidirectional tree-structured lstm with head lexicalization. *arXiv preprint arXiv:1611.06788*.

Pengtao Xie, Aarti Singh, and Eric P. Xing. 2017. Uncorrelation and evenness: a new diversity-promoting regularizer. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3811–3820.

Pengtao Xie and Eric Xing. 2017. A constituent-centric neural architecture for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1405–1414.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention.

Zhicheng Yan, Hao Zhang, Robinson Piramuthu, Vignesh Jagadeesh, Dennis DeCoste, Wei Di, and Yizhou Yu. 2015. Hd-cnn: hierarchical deep convolutional neural networks for large scale visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2740–2748.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*.

Yao-Liang Yu and Eric P Xing. 2016. Exact algorithms for isotonic regression and related. In *Journal of Physics: Conference Series*, volume 699, page 012016. IOP Publishing.

Xinqi Zhu and Michael Bain. 2017. B-cnn: Branch convolutional neural network for hierarchical classification. *arXiv preprint arXiv:1709.09890*.