Adapting the Transformer Architecture for ICD-9 Code Assignment

Justin Lovelace, Ryan Wells
Texas A&M University
{justinlovelace, rawells14}@tamu.edu

Abstract

ICD-9 coding has been a standard for labeling patient electronic health records to create a concise and clear patient history. By automated assignment of ICD-9 coding, medical facilities can improve the efficiency and accuracy of representing a patient. We propose a novel adaptation of the popular transformer model to the task of multi-label document classification.

1. Introduction

Electronic Health Records (EHRs) contain a large quantity of unstructured clinical notes generated by physicians during a patient's hospital stay. These textual narratives are often labeled with codes standardized by the International Classification of Diseases (ICD). These codes provide a concise, structured summary of the patient's diagnoses and procedure history.

Manual assignment of these codes, however, is a slow and error-prone task even for trained human coders. This problem is due in part to the large amount of possible codes available. For example, the set of codes that we will be looking at (ICD-9-CM) defines nearly $18,000$ procedure and diagnosis codes. Among the items taken into account during assignment are primarily discharge summaries. These summaries represent a concise overview of a patient's hospital stay, making them well-suited for predicting these codes. Due to the inefficiency and challenge of assigning these codes, there is a demand for a machine learning approach to aid in the annotation process.

1.1. ICD-9-CM Codes

ICD-9-CM codes are the United States version of the Ninth Revision of the International Classification of Diseases. They define a hierarchy of alpha-numeric codes that map to a wide variety of diseases and diagnoses. These codes are organized in a tree-like structure where children represent subcategories of their parents. These codes serve a variety of administrative purposes in the clinical setting such as billing and have been found to be very useful for patient phenotyping and predictive modeling of patient states.

1.2. Applications

These codes were originally intended for use in the clinical setting and serve a variety of purposes there. Among these are, billing, epidemiological assessment, and quality control of healthcare providers. Beyond the clinical setting, they have also been found useful by the health informatics community. They aid in cohort identification and have been successfully used in a variety of predictive modeling tasks. By automating this task or aiding the assignment of these codes, trained medical professionals can use their time more efficiently.

1.3. MIMIC-III

The MIMIC-III database contains a centralized dataset of diverse patient metrics that were admitted to critical care units. This unique dataset is openly accessible and contains over $53,000$ hospital admissions and over $18,000$ distinct ICD-9-CM codes that can be used for academic research [2]. Some of the metrics associated with each admission record in this database include clinical measurements, patient demographics, vitals, medications, and importantly, the respective ICD-9-CM code. This database is hosted through *Physio-Net*, allowing for query building and easy access to the dataset [5]. For this paper, we downloaded the full database to store it locally while preprocessing and training.

1.4. Challenges

This problem can be approached as a multi-label document classification problem where we must develop a model to assign an arbitrary number of labels to a given discharge summary. There are a number of characteristics of the task that make automated ICD coding especially difficult. The first is the extremely large potential label set in which some labels are used very sparsely. Another challenge is the multi-label setting where some patient narratives can contain a large number of these labels. Another challenge arises from the nature of the text that we will be working with. Due to the length of the narratives that we will be working with, it is also important to highlight the relevant sections that were important for predicting each label. This will aid in the interpretability of our model which is an important aspect of any decision support system. An additional challenge is the length of the discharge summaries. Due to the complexity of these summaries, long-term dependencies within the text must be learned. Lastly, the error of disease sub-type classification occurs commonly, even in manual coding assignment. Since several codes can be grouped under a common disease, the differences between some of these diseases can be subtle. Small excerpts from these discharge summaries can be seen in Table 1 with their corresponding ICD-9 Code.

Table 1. A few examples of medical codes and their corresponding discharge summaries presented in Mullenbach et al. [7]

ICD-9 Code	**Discharge Summary**
934.1	*...line placed bronchoscopy performed showing large mucus plug on the left on transfer to...*
442.84	*...and gelfoam embolization of right hepatic artery branch pseudoaneurysmcoil embolization of the gastroduodena...*
428.20	*...no mitral valve prolapse moderateto severe mitral regurgitationis seen the tricuspid valve...*

2. Related Work

2.1. Automated ICD-9 Coding

As stated in Section 1.4, many of the challenges within the scope of automated ICD-9 coding can also be seen in the traditional area of multi-label document classification. One of the most noteworthy of these is the non-mutual exclusivity of ICD-9 codes. Solutions to this problem have been proposed by researchers though a variety of techniques. Some of the most noteworthy works proposed by such include Baumel et al. (HA-GRU) [1], Mullenbach et al. (CAML) [7], and Sadoughi et al. (MVC-LDA) [8].

2.1.1 HA-GRU

Baumel et al. explored a number of possible approaches for automated ICD-9 coding [1]. Among the models tested, they saw the best performance with max-pooled CNNs and the proposed Hierarchical Attention bidirectional Gated Recurrent Unit model (HA-GRU). The HA-GRU model is comprised of multiple Gated Recurrent Units (GRUs) with bi-directional encoding. The first GRU operates over individual tokens at the sentence level. The second GRU operates at the document level, and is applied on encoded sentences. This allows the model to operate on very large documents, unlike sole GRUs, where the sequence applied would be the number of tokens in the documents. For documents within the MIMIC-III database, this could potentially be up to $13,590$ tokens which is computationally more difficult [1, 2].

In addition to the raw prediction of codes, they were able to explore the explainability of their models by looking at the n-grams that triggered their max-pooling layers for their CNN model and by looking at the sentences that received the most attention for their HA-GRU model. This resulted in a micro-F1 score of 40.52% on all ICD-9 codes and 55.86% on rolled-up codes [1] on the MIMIC III dataset [1, 2].

2.1.2 CAML

Mullenbach et al. later introduce a method they term Convolutional Attention for Multi-Label classification (CAML) which is able to outperform the earlier work performed by Elhadad et al. [7]. This model utilizes an attention mechanism to select the most relevant n-grams from the text for each ICD-9 label. This work also viewed the task of ICD-9 code assignment as requiring explainability since physicians will frequently observe certain subsets of clinical notes as having more importance. This resulted in the work proposed by Mullenbach et al. leveraging the fact that different portions of the text may be more or less relevant for different labels.

Their proposed method consisted of utilizing the combination of convolutional filters and label-wise attention. They initially convolve a filter with pre-trained word embeddings of the summaries. This produces a Matrix, **H**, representing the document. As stated, different subsets of the text have different amounts of relevancy so naturally, this resulted in the decision to not go the traditional route of converting **H** into a vector through pooling. Instead, they utilize per-label attention which selects the best k-grams with respect to the relevancy per predicted label. Finally, they Softmax the attended output, and take the Sigmoid to compute the final probability distribution of the classes.

This model resulted in the CAML model producing a micro-F1 score of 63.30% across the top 50 labels in MIMIC III [2, 7]. Another useful metric that they performed at state of the art was their Macro-AUC and Micro-AUC of the ROC curve. They had a Macro-AUC of 88.40% and a Micro-AUC of 91.60%. An added benefit of the attention mechanism is the additional interpretability of the

[1]Rolled-up ICD-9s are codes in which only the first three digits are left

model. They also explored extending their model with regularized attention to force codes with similar descriptions to have similar attention mechanisms but saw a slight decrease in their micro-F1 score.

2.1.3 MVC-LDA

Sadoughi et al. built upon the work done by Mullenbach et al. and proposed a Multi-View Convolution Model with Label Dependent Attention (MVC-LDA) [8]. They first take the initial document and encode words into some continuous embedded space with the CBOW Word2Vec method [8, 6]. These embeddings are then passed to the convolutional layer which is comprised of multiple convolutional filters of varying kernel widths. The purpose of this is to select the most relevant n-grams from the text which can have different lengths with respect to the word neighborhood. These differing kernels would have the ability to capture this. The convolutional layer's output is then max-pooled and passed to the attention pooling layer. They then apply an attention mechanism in a similar manner to Mullenbach et al. to attend to the most informative sections of text for each label. Finally, the predicted code is then generated with a Sigmoid non-linearity where an output of $> .5$ signifies the code as being assigned and otherwise is not.

This work also added an additional variant of their MVC-LDA model, utilizing a regularized attention, forcing classes with similar descriptions to be closer together. They called this variation MVC-RLDA and found a significant performance increase over the non-regularized attention method. The MVC-RLDA currently holds the state of the art with a micro-F1 score of 55.85% across all codes and 67.41% across the top 50 most frequent codes. It also holds the state of the art for Precision@5 at 64.11%.

3. Method

3.1. Preprocessing

We follow the same preprocessing methods used by previous works on this problem to allow for a direct comparison of modeling differences. Much of this pre-processing was performed in CAML by Mullenbach et al., and we were able to utilize their implementation [7, 3]. All tokens without any alphabetical characters are discarded and documents are truncated to 2500 tokens. The remaining tokens are then lower-cased. Tokens that appear in fewer than 3 training samples are anonymized using an 'UNK' token, because it would be difficult to train meaningful embeddings on tokens with so few occurrences. We use Word2Vec with the CBOW implementation to train word embeddings of dimension 100 upon the corpus of discharge summaries in our training set [6].

3.2. The Transformer

The current state-of-the-art upon our dataset is the work performed by Sadoughi et al. using multi-view convolutions with attention. A drawback of this work is that CNNs can only capture the local context within the width of the convolution while long-term dependencies in text can often be informative. The transformer model was originally introduced by Vaswani et al. for the task of language translation, but has since been successfully applied to a number of other tasks [9]. This model is built upon a mechanism they proposed called self-attention which allows each word to simultaneously attend to all other words in a sequence. This method has been shown to be much better at capturing long term relationships in text than CNNs and LSTMs. In addition to this, the attention mechanism makes it more interpretable than CNNs and LSTMs.

We propose to adapt the transformer encoder to the task of multi-label document classification for the purpose of ICD-9-CM code assignment. We believe that its ability to capture long term dependencies in text will enable it to outperform the previous CNN-based models. To adapt the transformer encoder to this problem we will have to make a number of modifications to the model. The current transformer encoder outputs a representation for each word in the input sequence and we would need to aggregate these inputs in order to make the classification decision. We believed that using an attention mechanism for each label as was done in the previous work would allow us to do this effectively in an interpretable manner. In adapting the transformer model to our task, we use the open source Pytorch implementation of the transformer that is available from the Open-Source Toolkit for Neural Machine Translation[4].

3.2.1 Transformer Architecture

This work only uses the encoder module from the transformer proposed by Vaswani et. al [9]. As such, we will only describe the architecture of the encoder. The transformer encoder is a stacked model composed of B identical blocks where each block has its own set of parameters. These blocks are composed of two primary components, a self-attention layer and a feed forward layer with one hidden state.

3.2.2 Self-Attention

Attention mechanisms are commonly used with language tasks to allow models to directly attend to relevant sections of text. The work by Vaswani et al. proposed self-attention, which defines a mapping from a query and a set of key-value pairs to an output. This output is computed as a weighted sum of the values where the weights are determined by a function between the query and keys. The

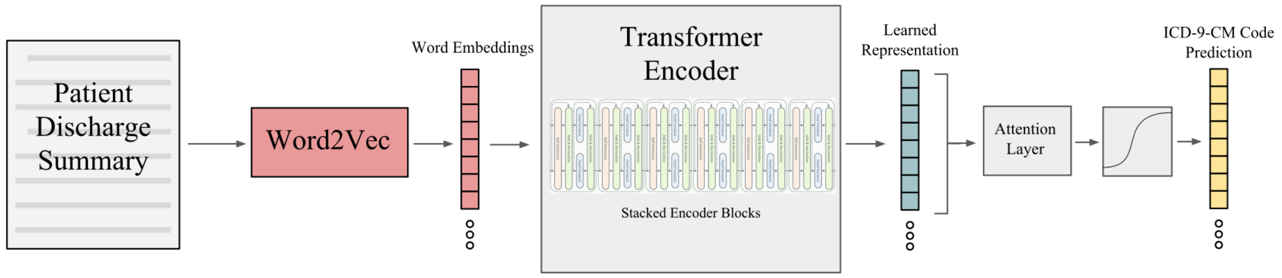

Figure 1. Our proposed architecture

designers of the transformer model decided to use the dot product between a query and all keys and then scale it by the dimensionality of the queries and keys, d_k, to develop a score over every value. After scaling it, they would take the softmax of the values to calculate the weights for the final weighted average. They could calculate the attention for multiple queries simultaneously by packing all queries, keys, and values into matrices Q, K, and, V. Using these matrices, the final output matrix can be calculated using the following equation.

$$\text{Self-Attention}(Q, K, V) = \frac{QK^T}{\sqrt{d^k}}V \tag{1}$$

The designers also introduced multi-headed attention which involves using multiple self-attention mechanisms in parallel. This allows the different heads of the model to attend to different features from the text. The outputs of these different attention heads are concatenated and then projected using a learned linear projection to achieve the final encoded value.

3.2.3 Feed-Forward Layer

The output tokens from the self-attention layer are then passed independently and separately through a feed-forward network. This feed-forward network consists of two linear transformations with a ReLu activation function between them. This function can be given below:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2}$$

The output from that would then be fed to the next block of the encoder which would have an identical architecture, but different weights. This is done until the final layer which produces the output that we use for prediction in this work. This architecture of a single block is laid out in Figure 2.

3.2.4 Stacked Convolutions

In this work we do make one modification to the transformer architecture originally proposed by Vaswani et. al [9]. The original model includes a feed forward neural network with one hidden state that is identically applied to each token after the self attention layer. This is equivalent to two convolutional layers of width one. We follow a modification made by previous work on relation extraction by Verga et. al and add a convolutional layer with a kernel width of 5 between the two existing convolutional layers [10]. These stacked convolutions can be represented with the following equations where $C_w()$ denotes a convolutional operator with width w:

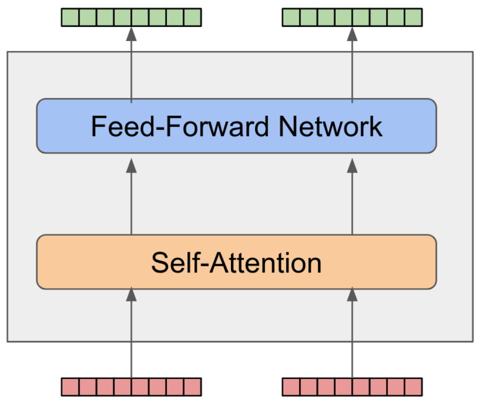

Figure 2. A single block of the encoder

$$t^{(0)} = \text{ReLU}(C_1(x))$$
$$t^{(1)} = \text{ReLU}(C_5(x))$$
$$t^{(2)} = \text{ReLU}(C_1(x))$$

Like the previous work, we found this variant to improve performance. We experimented with other values for the width of the kernel but found that a width of 5 consistently led to the best improvement. We also experimented with adding multiple convolution layers, but again found this to perform slightly worse. We believe that this change allows the transformer model to more directly capture local context, which is often especially important for language tasks. This modification can be seen in 3.

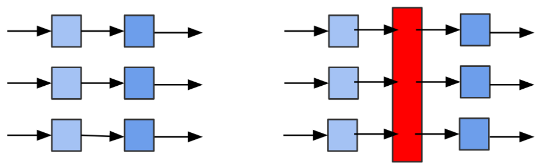
Figure 3. A comparison between the original feed forward architecture with one hidden state (left) and the modified feed forward architecture with an added convolutional layer (right)

3.3. Positional Encoding

Because the transformer model contains no recurrence and, as such, has no innate notion of position, we add positional encodings to the word embeddings before inputting them to the model. The original work explored two different methods for developing these encodings. One of these methods involved learning a position embedding for each position encountered during training time, and the other involved using sine and cosine equations of different frequencies. We use the second method in this work and develop positional encodings according to the formulas

$$\begin{aligned} \mathrm{PE}_{(pos,2i)} &= \sin(pos/10000^{2i/d_{model}}) \\ \mathrm{PE}_{(pos,2i+1)} &= \cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{3}$$

where pos is the position and i is the dimension. So, each dimension of the positional encoding corresponds to a different sinusoid. The authors of the original work found this method to effectively encode the sequential order of the text and it worked effectively with this work as well.

3.4. Final Attention Layer

The prior work that used CNNs utilized a separate attention mechanism for every ICD-9 code to allow their models to attend to different parts of the text based on the label that they were predicting. Because there is a diverse range of labels that often refer to completely unrelated things, this improves the model's capability to differentiate between what spans of text are predictive for the different labels. We follow the earlier work by Eisenstein et al. [7] and use the same attention mechanism, but apply the per-label attention mechanism to the output of the final transformer layer rather than the output of a CNN.

The attention mechanism involves learning a separate vector $u_\ell \in \mathbb{R}^d$ for each label where d is the dimensionality of the encoded words outputted by the model. The attention score for each outputted word is calculated by taking the dot product of the the learned vector with each encoded word outputted by the model. This can be done efficiently by treating the output of the transformer as a matrix $H \in \mathbb{R}^{dxN}$ where N is the length of the input and then computing the matrix-vector product $H^t u_\ell$ for every label ℓ.

A distribution over the words of the document is then calculated by taking the softmax of this output. This can be given by

$$a_\ell = \mathrm{SoftMax}(H^t u_\ell) \tag{4}$$

where a_ℓ is the attention vector for a given label ℓ. This is used to calculate the final value, v_ℓ used for the prediction of each label ℓ. This can be written as

$$v_\ell = \sum_{n=1}^{N} a_{l,n} h_n \tag{5}$$

where the attention weights are used to compute a different weighted average of the encoded output of the transformer for each label. The final value for each label is then passed to a sigmoid to calculate the probability of each labels' assignment.

3.5. Drawbacks

One drawback of the transformer model as compared to LSTMs and CNNs is that it does not scale as well to longer inputs due to the computational cost of self-attention. This caused some early concerns about how well it would be able to handle the sequence lengths of 2500 tokens that are needed for this task. We found that the transformer model was able to scale up to handle the sequences required for this task, although it was quite memory intensive. Due to the memory limitations of our hardware, we were only able to test the model with a small number of layers and a small batch size. However, this did not appear to be a problem as increasing the number of layers and the batch size seemed to slightly decrease performance. Our work suggested that these limitations of the model did not harm its performance as we originally feared.

4. Experimental Results

4.1. Dataset

Most of the previous work in ICD-9 Code assignment has used the top 50 most frequent codes for evaluation. To do this, our data is distilled down to only items that contain one of the top 50 labels. We then train, validate, and test on this subset of the data to better compare our results with other works. The splits used in this work are the same as those used in the works we compare against. This makes the evaluation of our results straightforward. This breakdown of our cohort can be seen in Table 2.

Table 2. Dataset size using the Top 50 codes

Training	$8,067$
Testing	$1,730$
Validation	$1,575$

4.2. Performance

We evaluated our model on a number of different testing metrics widely employed by previous research. These include Micro-F1, Macro-F1, Precision@5, and the micro and macro AUC of the ROC curve. The macro averaged metrics are calculated by averaging the metrics computed for each label, while the micro averaged metrics are computed by aggregating all predictions across all labels and then computing the metrics.

In Table 3, we display these results comparing against the top performing models and illustrate the improvement in performance, specifically in the Macro-F1, Macro-AUC, and Micro-AUC. As of 2019, MVC-RLDA [8] held the current state of the art for Micro-F1, Macro-F1, and Precision@5, though it did not report it's Macro or Micro AUC of the ROC. Our model improved upon the Macro-F1 score of MVC-LDA from 59.65% and 61.47% to 62.32%. Additionally, comparing to CAML [7], the state of the art model going into 2018, we saw an improvement in Macro-AUC from 88.40% to 90.55% and in Micro-AUC from 91.60% to 92.32%

4.3. Training Specifics

While we tried many different variants, our current best performing model in terms of the Macro-F1 and Macro-AUC had the following hyperparameters:

Initially, we thought that having 2 Transformer encoder layers would be the ideal amount of complexity however, we found empirically through testing layer counts of $1-3$, that having a single layer performed best. An additional observation for a model variant with the Dropout set to 0.2 and with 2 encoder layers, with all other parameters equal, resulted in a Micro-precision boost from 65.20% from the model with performance in Table 3 to 70.95%. For applications where a high degree of the codes predicted is more crucial, this type of performance may be preferred over a higher recall.

We also tried varying the number of attention heads from 2 to 10. However, we found that there was little performance change in these different variants.

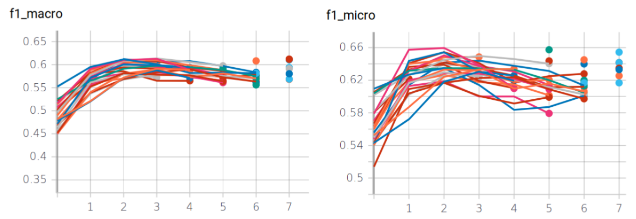

Figure 4. Macro-F1 (left) and Micro-F1 (right) Scores versus epoch at training time.

In terms of model convergence, we measured testing convergence with respect to Micro AUC of the ROC. As presented in Figure 4, we plotted approximately 40 different training variants performance in terms of Macro-F1 and Micro-F1 scores. From this plot, we can deduce that on average, the model converges within 3-4 epochs [3]

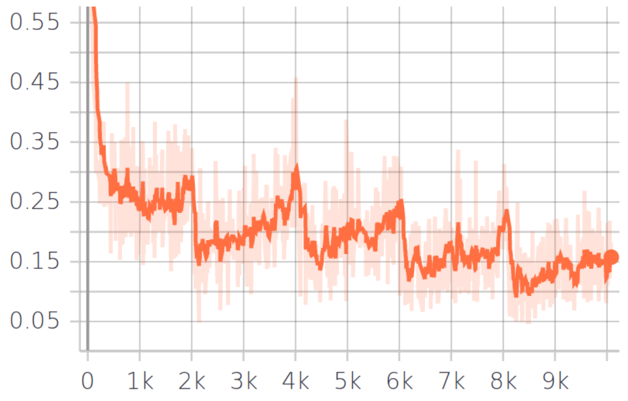
Figure 5. An example of a training loss curve versus iteration from training the Transformer Adaptation

We can also view the training convergence occur in Figure 5 on an iteration basis where each Epoch has approximately $2,000$ iterations. The spiked nature of this graph reinforces the idea that the longer the sequence, the harder it is to predict the corresponding label. This is due to the fact that within each Epoch, the sequences are sorted by length in ascending order. In terms of training hardware, the Transformer adaptation could fit fully on a GTX 1080 Ti with a batch size of 4.

5. Conclusion

In this paper, we demonstrated a novel adaptation of the Transformer encoder [9] for the ICD-9 code assignment task. The adaptation included pre-processing the discharge summaries with Word2Vec with continuous bag of words, following the same methodology as Mullenbach et al [7, 3, 6]. We then took these embeddings and passed them into a modified transformer encoder with an added convolutional layer. This was then passed through a final attention layer, the sigmoid was computed, and the final probability for each code was predicted. This architecture ultimately resulted in outperforming current state of the art models in terms of Macro-F1, Macro-AUC of ROC, and Micro-AUC of ROC. Our code is publicly available on GitHub here.

6. Future Work

This work demonstrated that the transformer performs very well for the task of ICD-9 code assignment. However, there are a number of further modifications that could be explored that could potentially increase the performance of the

[3]The performance of epoch 0 is measured upon the completion of the first epoch.

Table 3. Performance comparison between state of the art models and the proposed Transformer Adaptation on Top 50 ICD-9 Codes

Model	Micro-F1	Macro-F1	P@5	Macro-AUC [2]	Micro-AUC
DR-CAML [7]	63.30%	57.60%	61.80%	88.40%	91.60%
MVC-LDA [8]	66.82%	59.65%	64.43%	-	-
MVC-RLDA [8]	**67.41%**	61.47%	**64.11%**	-	-
Transformer Adaptation	65.58%	**62.32%**	61.94%	**90.55%**	**92.32%**

Table 4. Hyperparameters for our best performing model

Hyperparameter	**Value**
Encoder Layers	1
Number of Attention Heads	4
Dropout	0.4
Learning Rate	0.0005
Batch Size	1

model. Both of the prior works on this topic explored methods to regularize the final attention layer using the textual descriptions for the ICD-9 codes and found that doing so improved performance. Following this earlier work would likely lead to improved performance for the model explored in this work as well.

The work by Sadoughi et al. [8] also observed a statistically significant correlation between the length of the discharge summary and the number of codes assigned. As a result of this, they added a bias term conditioned on the input length to the final sigmoid layer of their model. They found this to help avoid problems with under-coding that resulted from the sparse nature of ICD-9 code assignment. This is something that could also be applied to our model that would likely help improve our precision.

Additionally, the prior work has also explored different experimental settings such as making predictions over the entire label space. Exploring this would allow us to evaluate how well our model works when working with a more diverse label space. This would allow for a more complete comparison to the prior work.

7. Individual Contributions

Justin Lovelace

- Helped develop initial idea
- Helped apply preprocessing techniques used in previous works
- Adapted the transformer model to work with the task of ICD-9 code assignment
- Helped train Transformer Adaptation

Ryan Wells

- Helped with initial brainstorming phase and problem statement
- Added CAML's implementation of Discharge Summary Preprocessing
- Helped train Transformer Adaptation
- Set up Tensorboard for PyTorch

References

[1] T. Baumel, J. Nassour-Kassis, M. Elhadad, and N. Elhadad. Multi-label classification of patient notes a case study on ICD code assignment. *CoRR*, abs/1709.09587, 2017.

[2] A. Edward William Johnson, T. Joseph Pollard, L. Shen, L.-w. Lehman, M. Feng, M. Ghassemi, B. Edward Moody, P. Szolovits, L. Anthony G. Celi, and R. G. Mark. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3:160035, 05 2016.

[3] jamesmullenbach. caml-mimic. `https://github.com/jamesmullenbach/caml-mimic/`, 2018.

[4] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*, 2017.

[5] M. Laboratory For Computational Physiology. The mimic iii clinical database, 2015.

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

[7] J. Mullenbach, S. Wiegreffe, J. Duke, J. Sun, and J. Eisenstein. Explainable prediction of medical codes from clinical text. *CoRR*, abs/1802.05695, 2018.

[8] N. Sadoughi, G. P. Finley, J. Fone, V. Murali, M. Korenevsky, S. Baryshnikov, N. Axtmann, M. Miller, and D. Suendermann-Oeft. Medical code prediction with multi-view convolution and description-regularized label-dependent attention. *CoRR*, abs/1811.01468, 2018.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[10] P. Verga, E. Strubell, and A. McCallum. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. *CoRR*, abs/1802.10569, 2018.