TEAM3: ICD CODE PREDICTION FROM MEDICAL RECORDS (NLP FOR HEALTHCARE) PROJECT REPORT FOR CSE 6250: BIG DATA FOR HEALTH

Ayush Kumar*, Priyank Chhipa*, Sagar Sumit*
*Georgia Institute of Technology
Atlanta, USA
`akumar689@gatech.edu, pchhipa3@gatech.edu, ssumit3@gatech.edu`

ABSTRACT

Automatic ICD coding is an important text classification task in clinical domain that assigns relevant ICD-9 codes for patient visits that describe the diagnosis and procedure treatments given to the patient. Annotating these codes is labour intensive task and heavily rely on domain knowledge. In this work, we propose a hierarchy augmented network trained on joint-loss framework leveraging the hierarchy of labels to predict the full ICD-9 codes given a patient ICU discharge summary record. With initial experiments, we see that joint loss framework, helps in achieving up to 2.2% better micro-F1 score on full set prediction of ICD-9 codes. In another setup, we note 7% absolute improvement in micro-F1 score on using fine-tuned BERT embeddings on a feed-forward network baseline. Label-wise document attention boosts the performance of the system by 6% on micro-F1 score. The presentation for this work is available at *YouTube* link. The code for this work can be found at this *GitHub* repository.

1 Introduction

International Coding of Diseases (ICD) is the diagnostic classification standard for all clinical and research purposes. Its application range from monitoring the incidence and prevalence of diseases globally on a macro level, to observing reimbursements and resource allocation on a micro level. It is estimated that about a quarter of hospital spending in the United States is spent on billing and insurance related activities. Coding diagnosis and procedures through human efforts is not only time-consuming but also costly and often erroneous. With huge amount of structured and unstructured patient health records available, in this project, we attempt to use big data and natural language processing (NLP) techniques to map clinical notes to ICD-9 medical codes.

Automatic ICD coding is a challenging task for three reasons: a) Each patient note is mapped to multiple labels, hence this transforms to a multi-label classification; b) The label set of ICD-9 taxonomy is in the range of 10000 labels for flat full set with skewed distribution; and c) The clinical notes contain a lot of additional information, variations in spellings and acronyms which makes it difficult to transform the document in the uniform feature space.

In this work, we explore three directions of experiments for ICD code labeling. Firstly, we use different architectures to train the system for multi-label classification. Secondly, we note that ICD-9 codes can be grouped into different hierarchies (Ref: 3.2). Most of the existing work ignore this auxiliary information and train networks on flat-labels. We, instead, ingest the auxiliary information in the hierarchy of labels to train a hierarchy augmented network with joint loss to predict flat ICD-9 codes. Lastly, we use different embeddings, namely Word2Vec [1] and BERT [2] based embeddings for clinical text. Our experiments show positive direction in utilizing the joint loss network and BERT based embedding over the MLP baseline.

*Equal Contribution

2 Literature Survey

Most of the prior research have used supervised classification setup for ICD coding. Work by Pinheiro et al. [3] combined three classifiers to assign ICD codes based on inpatient discharge summaries. However, their work was limited to assigning one code to each patient. Perotte et al. [4] worked on multi-label ICD coding using flat and hierarchical SVMs demonstrating that hierarchical SVM achieves better performance. Gradually with development of deep learning techniques, works on ICD coding expanded to use recent architectures comprising convolutional neural networks (CNNs) and recurrent neural networks (RNNs). Mullenbach et al. [5] uses Convolutional Attention of Multi-Label classification of ICD-9 codes based on text discharge summaries. To address the issue of large label space, authors use code textual descriptions to guide model to learn similar parameters for codes having similar textual descriptions. Further, authors use per-label attention to introduce interpretability of discharge summaries resulting in one or more ICD codes. Attention selects important keyphrases that plays a major role in determining a particular ICD code. Xu et al. [6] uses multimodal data of structured, unstructured and tabular data to identify the associated ICD-10 codes. Specifically, authors train three separate models for structured, unstructured and tabular data, and finally train an ensemble classifier to combine the predictions. Li and Yu [7] uses Multi-Filter Residual Convolution Network to predict ICD codes with the hypothesis that single filter fails to capture the information across texts with varying lengths. Song et al. [8] shows zero-shot learning of ICD codes using latent feature generation framework that leverages the hierarchical information of ICD codes. Rios et al. [9] proposes a framework for zero-shot and multi-shot ICD code classification by matching discharge summaries in EMRs to feature vectors for each label obtained by exploiting structured label spaces with graph CNNs. Rios et al. [10] performs transfer learning from PubMed dataset and use freezed convolution layer for MIMIC datasets.

3 Methodology

In this section, we explain the different lines of experimentation that we have explored to determine the ICD-9 codes from the given discharge summary. Primarily, we explore three directions of work.

3.1 Experiments with multiple architectures

- Feed-forward network (MLP): We train feed-forward network as the baseline architecture for the multi-label text classification task.
- Convolutional Neural Network (CNN): CNN is one of the most popular architecture to train systems for text classification. We use CNN to compare with the MLP baseline, as well as with current state-of-the-art systems.
- Transformers: Transformers [11] is a self-attention mechanism-based architecture well suited for language understanding. Since, they dont require processing to be sequential, they have replaced the RNN cells. We use Transformer to learn document representation.

3.2 Experiments with aggregating information from label hierarchy

MIMIC-III dataset contains patient records labelled with ICD-9 codes. These ICD-9 codes can be grouped together under different hierarchies. To leverage the hierarchical information, we utilize two hierarchies present in the ICD-9 codes. In a simple setup, The three level of hierarchies are defined as:

- Flat Labels/Hierarchy: The full set of ICD-9 codes represented by 3 to 6 characters is termed as flat hierarchy. In MIMIC-III v1.4 dataset, we have 8908 unique flat labels.
- Leading Labels/Hierarchy: The labels represented by characters preceding the decimal in flat representation constitute the leading hierarchy. We have a total of 1159 unique leading labels.

3.3 Experiments with different word embeddings

We use word embeddings as feature to train the neural architectures for predicting ICD-9 codes. In our experiment, we use two embedding methods: a) word2vec and b) BERT. We use the discharge summaries to train word2vec model. For BERT, we use standard BERT embedding, and a fine tuned clinical BERT embedding [12] as detailed in subsection 4.6.

The network architectures of the models are outlines in Table 1. Additionally, in linear layers of MLP and CNN models, we use ReLU activation while post convolution, we use Tanh activation. We use binary cross entropy loss function to train the network.

Sl. No	Approach	Network Parameters
M1	MLP, Flat Loss, word2vec	input ->(512, 1024, 1024, 512, flat_class) ->output
M2	MLP, Joint Loss, word2vec	input ->(512, 1024, 1024, 512-H, [(512-H: flat_class), (512-H: leading_class)]) ->output
M3	CNN - 1D, Flat Loss, word2vec	input ->(conv: [kernel size = 5, stride=1, num filters = 1024], pooling: max, fc: (512, flat_class))->output
M4	CNN - 1D, Joint Loss, word2vec	input ->(conv: [kernel size = 5, stride=1, num filters = 1024], pooling: max, fc: (512, flat_class), (512, leading_class)) ->output
M5	MLP, Flat Loss, generic BERT	input ->(512, 1024, 512, flat_class) ->output
M6	CNN - 2D, Flat Loss, generic BERT	input ->(conv: [kernel size=(5,emb_size), stride=1, num filters=5], pooling: max-2d (2,1), fc: (3120, 1024, flat_class)) ->output
M7	MLP, Flat Loss, clinical BERT - all notes	input ->(768, flat_class) ->output
M8	MLP, Flat Loss, clinical BERT - discharge summaries only	input s->(768, flat_class) ->output
M9	CNN-1D MultiKernel with Attention, - Flat Loss, word2vec	input ->(conv: [kernel size=[5,9,15], stride=1, num filters=100] label-attention) ->output
M10	CNN-1D MultiKernel with Attention, - Joint Loss, word2vec	input ->(conv: [kernel size=[5,9,15], stride=1, num filters=128] flat-label-attention, leading-label-attention) ->output
M11	Transformer, Joint Loss, word2vec	input ->(4-layer transformer block) ->output transformer block => (self-attention, fc:(768, 3062, 768), batch norm)
M12	Transformer, Joint Loss, PubMed-word2vec	input ->(4-layer transformer block) ->output

Table 1: Network architectures. *flat* represents the setup where flat labels are used, while *leading* represents the setup where leading hierarchy of labels are used as auxiliary information to train the network.

3.4 Label Attention Mechanism

Since this is a multi-label classification setup, different words and phrases can be significant for individual labels. Hence, with a motivation to learn label specific important terms in the discharge summaries, we apply a label-wise attention on the document representation. Similar to Mullenbach et al. [5] and Li et al. [7], we apply attention layer on the intermediate document representations, $D \in R^{n \times m}$, where n is the number of words in the document and m is the length for document representation wrt each word. Specifically, with a set of l labels, the attention layer is applied using following set of transformations, where A represents the attention weights for each pair of word in the document and ICD-9 code:

$$C = DU; U \in R^{m \times l} \tag{1a}$$

$$A = softmax(C); A^{n \times l} \tag{1b}$$

$$V = A^T R \tag{1c}$$

The matrix V is finally sent through a final hidden layer to predict multi label ICD-9 codes.

3.5 Loss Function

For training, we treat the ICD coding task as multi-label classification task and use binary cross entropy (BCE) loss for optimization:

$$L(y, \theta) = -\sum_{i=1}^{l} y_i log(\hat{y_i}) + (1 - y_i) log(1 - \hat{y_i}) \tag{2}$$

In case of multi-loss network, the loss function defined by the weighted summation of loss from different hierarchy of labels (we use flat and leading hierarchy of labels). Also, in our experiments, $\alpha = 1$ and $\beta = 1$ give the best results on the flat label predictions.

$$L(y_t, \theta) = \alpha L_t(y_f, \theta) + \beta L_l(y_l, \theta) \tag{3}$$

4 Experiments

4.1 Dataset

We use the clinical notes data present in the MIMIC-III (Medical Information Mart for Intensive Care III) database. The dataset consists of de-identified health-related data of 46,520 distinct patients admitted to Beth Israel Deaconess Medical Center in Boston, Massachusetts from 2001 to 2012. Since, our goal is to classify clinical notes into multiple ICD-9 codes, so we chose to work with *NOTEEVENTS, DIAGNOSES_ICD and PROCEDURES_ICD* tables.

NOTEEVENTS table contains over 2 million records of clinical notes of different patients over all of their admissions. *DIAGNOSES_ICD* table contains over 650,000 records of ICD-9 diagnosis codes for patients. *PROCEDURES_ICD* table contain over 240,000 records of ICD-9 procedure codes for patients.

4.2 Experimental Setup

Preprocessing pipeline and machine learning models are implemented in Python 3.6. To setup the preprocessing pipeline and perform feature engineering, we use PySpark package on a 16GM RAM, 4 CPU core machine. We use PyTorch to build the machine learning models outlined in the section 3. The models are trained on an external Linux-based virtual machine (VM) provided by Google Cloud Platform (GCP). We use GCP's *n1-highmem-8* VM type, which has 8 virtual CPUs, 52GB RAM, with 1 Nvidia Tesla P4 GPU for training of deep learning model.

4.3 Data Cleaning and Preprocessing

Following the convention in [5], we focus only on discharge summaries. This allows us to work with the clinical notes gathered after the patient has been discharged. The preprocessing steps are as follows:

- Filter the records in NOTEEVENTS whose category is discharge summary.
- Since the clinical text is often large and contains many newline characters, it is cleaned using regular expression operations.
- Merge the PROCEDURES_ICD and DIAGNOSES_ICD codes (reformatted to place decimal point at the correct position), and then join the merged dataframe with the cleaned notes dataframe by patient.
- Collect all the ICD-9 codes in a list while grouping by patients and their visits.

The final dataframe is divided into train, valid and test split of 47723, 1631 and 3372 discharge summaries respectively, such that patients across each dataset are unique. In total, there are 8908 ICD-9 labels spanning across diagnosis and procedure codes in the full dataset. This entire pipeline is implemented in PySpark module.

4.4 Feature Extraction

Since, we are experimenting with different architectures, each has its own unique feature extraction process. However, certain steps were common, which we have listed below:

1. Create vocabulary: Based on the clinical notes after preprocessing, the text is tokenized and a vocabulary of tokens is created. For word2vec based model, we use PySpark's CountVectorizer[2] API, which extracts a vocabulary from text document collections. Additionally, we ignore the terms which have not appeared in at least three documents. For BERT and Clinical BERT based models, we use the standard BertTokenizer and vocabulary of 28,996 words.
2. Vectorization: The embedding for the above tokens are generated depending on the model (Word2Vec or BERT). For word2vec based model, we use PySpark's Word2Vec[3] API that transforms the token into 100-dimension vector. For BERT and Clinical BERT based models, we use the standard 768-dimension vector for each token.
3. Input features: For each example of clinical text (document), we treat the token or sentence level embedding as the input feature. ICD-9 labels are one-hot encoded.

[2] `https://spark.apache.org/docs/latest/api/python/pyspark.ml.html#pyspark.ml.feature.CountVectorizer`

[3] `https://spark.apache.org/docs/latest/api/python/pyspark.ml.html#pyspark.ml.feature.Word2Vec`

Model No.	**Network Architecture**	**Embedding**	**Loss**	**AUC**		**F1**		**P@n**	
				Macro	**Micro**	**Macro**	**Micro**	**n=8**	**n=15**
M1	MLP	word2vec	Flat	0.837	0.975	0.037	0.336	0.457	0.352
M2	MLP	word2vec	Joint	0.844	0.976	0.037	0.344	0.467	0.360
M3	CNN - 1D	word2vec	Flat	0.835	0.976	0.041	0.416	0.589	0.451
M4	CNN - 1D	word2vec	Joint	**0.883**	**0.982**	0.040	0.438	0.608	0.466
M5	MLP	BERT - generic	Flat	0.784	0.966	0.010	0.268	0.355	0.279
M6	CNN - 2D	BERT - generic	Flat	0.837	0.975	0.031	0.338	0.461	0.353
M7	MLP	BERT - fine-tuned on all notes	Flat	0.778	0.963	**0.079**	0.372	0.287	-
M8	MLP	BERT - fine tuned on discharge	Flat	0.799	0.966	**0.079**	0.407	0.401	-
M9	CNN-1D Attn	word2vec	Flat	0.852	0.979	0.073	0.484	0.650	0.512
M10	CNN-1D-Attn	word2vec	Joint	0.865	0.981	0.064	**0.495**	**0.657**	**0.515**
M11	Transformer	word2vec	Joint	0.817	0.970	0.038	0.333	0.461	0.356
M12	Transformer	PubMed-word2vec	Joint	0.846	0.975	0.043	0.357	0.495	0.379
R1	*CAML - CNN*	word2vec	Flat	0.895	0.986	0.088	0.539	0.709	0.561
R2	*MultiRes-CNN*	word2vec	Flat	0.91	0.986	0.085	0.552	0.734	0.584

Table 2: Results of models (M) and reference state-of-the-art models (R), *flat* represents the setup where flat labels are used, while *joint* represents the setup where leading hierarchy of labels are used.

4.5 Word2Vec Embedding Based Model

For experimentation we use 100 dimensions of word embeddings for word2vec based models, use Adam optimizer with learning rate of 3e-4, and dropout of 0.3 across the models. We also find that weighing the positive labels to a higher value helps to achieve better micro-F1 score. In our experiment, using a label weight of 5.0 for flat loss models gives the best performance. For the multi-loss models, the weight of positive labels for flat hierarchy is 3.0 while weight of positive labels for leading hierarchy is 2.0.

4.6 BERT Embedding Based Model

- **Generic BERT**: BERT [13] is an Encoder model using Transformers to learn contextual relations between tokens (words or sub-words) in a text. The Tokenizer uses a greedy longest-match-first algorithm using the given vocabulary. The bert model is pre-trained on Nueral Machine Translation dataset for English to German and French from Wikipedia and BooksCorpus. For our experiments we use BERT model for token embeddings replacing the word2vec embeddings from earlier architectures keeping rest of parameters same. In addition, we experiment with CNN with 2D-convolutions of filter size (5, bert_emb_size) with 5 filters so that each filter learns over phrases parts of embedding differently.

- **Clinical BERT and Fine-tuning**: Clinical BERT [12] is a contextual word embedding model for clinical text. Our intuition behind fine-tuning a pre-trained model is that a domain-specific model would yield performance improvements as compared to generic models. Moreover, since the pre-trained model weights already encode information about clinical text, it would take much less time to train our fine-tuned model for multi-label text classification as well as it would prevent overfitting on the training data. While fine-tuning, the model is fed a patient's clinical notes and the ICD-9 codes classification is done by using a composition of clinical BERT model with a linear layer (of 768 hidden units) on top of the pooled output. The model parameters are fine-tuned to minimize the binary cross entropy loss with relative class weights in the loss function. Alsentzer et al. [12] released two clinical BERT models, one trained on all the clinical notes and the other trained only on discharge summaries. We fine-tuned both for our downstream task. To avoid memory issues, we use a maximum sequence length of 384 with a batch size of 32. We use Adam optimizer with an initial learning rate of 5e-5 and a warmup rate of 0.1. The model was trained for 5 epochs. We also use a dropout of 0.1 and Gaussian Linear Error Units (GELUs) as activation [14].

4.7 Transformer Based Model

Transformer[11] is a model to handle ordered sequences of data, such as natural language, for various tasks such as machine translation and text summarization. Since they do not require that the sequence be processed in order, these models have replaced Recurrent Networks. For our experimentation, we use 4-layer transformer block architecture learnt on top of word embedding and a joint-loss function of flat and leading labels. For learning, we use Adam optimizer with learning rate as 3e-4, and dropout of 0.3 across both the models. The transformer model was experimented with 2 different word embeddings, namely word2vec and PUBMed-word2vec[15], which is word2vec embedding trained on PUBMed dataset.

4.8 Evaluation Metrics

Based on literature survey, we find that authors benchmark the systems on three evaluation metrics which are mentioned below. We would compare our approaches with existing works using these evaluation metrics:

- Macro and Micro F1-Score: Micro-averaged values are calculated by treating each (text, code) pair as a separate prediction. Macro-averaged values are calculated by averaging metrics computed per-label.
- Precision at n (P@n) score: The score presents the fraction of the *n* highest scored labels that are present in the ground truth. This score is useful to evaluate the effort to correct machine mistakes by human annotators if they are presented with top *n* results. We report P@n for the values of *n*=8 and 15.
- Macro and Micro AUC score: We measure area under receiver operating curve to evaluate the model.

5 Results and Analysis

5.1 Overview

Overall, the we have best micro-F1 score of 0.495 by CNN-1D model trained on word2vec embeddings with label-wise document attention *M10*) trained on a joint-loss network. Same model achieves best P@n scores of 0.657 and 0.515 for $n = 8$ and $n = 15$ respectively. The best AUC scores are obtained from CNN-1D model trained on word2vec embeddings trained on joint-loss, with micro AUC being 0.883 and macro AUC being 0.982. Overall, our experimental results are still inferior to the state-of-the-art results.

Based on our analysis, we see these two explanations that hold for these two results: a) We did not tune the hyperparameters to the best possible setup due to limited resources and time. Our experiments were limited to tuning only learning rate, dropout and class weights for positive labels. Tuning other hyperparameters like embedding dimensions, kernel size, number of filter maps and weight decay can give additional gains in the performance., b) We could not experiment as well as we would have wanted with computationally complex architecture like fine-tuned BERT with CNN and transformer architecture because of computational challenges. The embedding size of 768 for fine tuned BERT embedding for long doccuments along with CNN architecture took a much longer time to train and hence, we could not have faster experimentation cycle with these network setups. However, based on our experiments, we note some positive directions which are outlined in the subsequent sections.

5.2 Analysis of joint loss network

Comparative results $((M1, M2), (M3, M4), (M9, M10))$ show that a hierarchy augmented network trained on joint-loss framework leveraging the hierarchy of labels outperforms the flat-loss network by upto 2.2%. This shows that adding auxiliary information helps system to better learn flat labels. In future, we can expand on this work to incorporate cross learning mechanism in addition to joint-loss to better learn both flat and leading labels.

5.3 Analysis of document representations

Even though, the model trained with word2vec embeddings gave best results in our experiments, we see the limitations in learning word2vec representations from discharge summaries. Based on comparison between a word2vec model trained only on discharge summary and a word2vec model trained with a larger in-domain dataset PubMed, we note gains with latter representation by 2.4% on micro-F1 score. This shows that a generic in-domain (PubMed) representation is better than one trained with a smaller (MIMIC-III) dataset. Further, in a different experimental setup, we see that a fine-tuned BERT model gives huge gain of 7% on micro-F1 score compared to word2vec embedding trained on MLP baseline ($M1$ vs $M8$).

5.4 Analysis of label-wise document attention

Attention helps to learn distinct document representation per-label and hence, in this particular case of large space skewed distribution, alleviates the challenges of vanilla CNN network. We note that label-wise attention helps by 5% in micro-F1 score ($M3$ vs. $M9$ and $M4$ vs $M10$).

6 Future Directions

Based on our current set of experiments, we see that utilization of joint loss helps to improve the micro-F1 scores. Furthermore, fine tuned clinical BERT used for document representation boosts the result compared to word2vec representation. We intend to perform these experiments:

- We note positive gains with joint-loss network. In future, we would like to explore alternative loss functions which inherently captures the hierarchy of labels. Additionally, a cross-learning network that learns flat and leading labels simultaneously can also be explored.
- Transformer trained with PUBMed-word2vec produces better results than word2vec trained on disccharge summary of MIMIC-III dataset. We would like to explore PUBMed embedding with other architectures.
- We tried to use graph based label embeddings in the joint loss model setup to see if we can better leverage the hierarchical information in label codes. We could not complete and train the whole network due to paucity of time, however, we are releasing the label emebddings [4] based on 2-layer graph convolutional network [16] of parent child relationship between ICD9 codes. The methodology to generate these embeddings is adopted from [17].We hope the embeddings can be useful for further exploration with not only CNN networks [17] but also RNN networks as done in the work by [18].

7 Challenges and Learning

- Scientific Challenges
  - Domain and Data: Clinical notes are often long, difficult to review, and contain information with significant variability and complex domain entities. There is a need to understand how clinicians review the notes and combine domain knowledge with deep learning models.
  - Multi-label Classification: Automatic ICD classification entails huge and sparse multi-label space with imbalanced classes. It is very challenging for non-grounded deep learning models to classify with high accuracy.
  - Document Representations: We used word2vec, PubMed, BERT and Clinical BERT embeddings. Domain-specific embeddings performed much better, however, BERT-based were also computationally heavy.
- Implementation Challenges
  - Dataset Size: While leading labels data was just a few hundreds of MBs, flat labels data was at least 10 times more and altogether we dealt with a dataset of close to 5GB.
  - Memory Issues: With around 50k documents, where each document has on average 1200 tokens, and each token further takes up the size of corresponding embedding dimension of the model. We frequently ran into CUDA memory issues. This reinforces the fact that deep learning models are becoming computationally expensive day by day.
  - Epoch Time: Even on the kind of GPU we used for our experiments, each epoch time ranged between 35 minutes to 4 hours for different experiments.

8 Effort Estimations and Team Contributions

Overall, the team agrees that every member has been proactively and equally involved in the experimentation, discussions and write-ups. Specifically, individual contributions can be summarized as:

- Contributions from Ayush Kumar
  - Word2vec pipeline (1 week): Task was to setup the pipeline for word2vec embeddings. I did initial experiments with different size of word embeddings.

[4]`https://drive.google.com/open?id=1gVmQJgheciLOSG5AVipoIe13uGKqvR_y`

  - MLP and 1-D CNN pipeline (2 weeks): This task involved setting up training architecture for MLP and 1-D CNN pipeline which included developing model codes, setting up batch training and saving models.
  - Multi-Loss framework (2 weeks): Based on the hierarchy of labels, the task was to setup the pipeline to incorporate hierarchy augmented training and loss.
- Contributions from Priyank Chhipa
  - BERT pipeline (2 weeks): This task involved setup pipeline for BERT based model pipeline. The initial experiments with BERT were using Feed Forward Network.
  - 2D-CNN pipeline (2 weeks): In this week, I experimented with using 2D Convolution after BERT along with Joint loss. In this task, multiple bert pre-trained models were also experimented.
  - Transformer Architecture (1 week): This task involved experimenting with transformer-based network using word2vec and PUBMed embedding.
- Contributions from Sagar Sumit
  - PySpark pipeline (2 weeks): Main effort went in data exploration and learning the efficient usage of PySpark's APIs, which have been used not only for ETL tasks but also for feature engineering.
  - Clinical BERT pipeline (2 weeks): PyTorch version of Google's BERT implementation [5] was leveraged, however, the main effort went into debugging the steps from data processor to training.
  - Graph Label Embeddings (1 week): Here, the main effort went into understanding the graph convolutional network architecture and how that can be applied to generate label embeddings given the hierarchy in ICD9 codes.

References

[1] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[3] Roberto H. W. Pinheiro, George D. C. Cavalcanti, and Ing Ren Tsang. Combining binary classifiers in different dichotomy spaces for text categorization. *Appl. Soft Comput.*, 76:564–574, 2019.

[4] Adler J. Perotte, Rimma Pivovarov, Karthik Natarajan, Nicole Gray Weiskopf, Frank D. Wood, and Noemie Elhadad. Diagnosis code assignment: models and evaluation metrics. *JAMIA*, 21(2):231–237, 2014.

[5] James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. Explainable prediction of medical codes from clinical text. *CoRR*, abs/1802.05695, 2018.

[6] Keyang Xu, Mike Lam, Jingzhi Pang, Xin Gao, Charlotte Band, Piyush Mathur, Frank Papay, Ashish K. Khanna, Jacek B. Cywinski, Kamal Maheshwari, Pengtao Xie, and Eric P. Xing. Multimodal machine learning for automated ICD coding. *CoRR*, abs/1810.13348, 2018.

[7] Fei Li and Hong Yu. ICD coding from clinical text using multi-filter residual convolutional neural network. *CoRR*, abs/1912.00862, 2019.

[8] Congzheng Song, Shanghang Zhang, Najmeh Sadoughi, Pengtao Xie, and Eric P. Xing. Generalized zero-shot ICD coding. *CoRR*, abs/1909.13154, 2019.

[9] Anthony Rios and Ramakanth Kavuluru. Few-shot and zero-shot multi-label learning for structured label spaces. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3132–3142. Association for Computational Linguistics, 2018.

[10] Anthony Rios and Ramakanth Kavuluru. Neural transfer learning for assigning diagnosis codes to emrs. *Artif. Intell. Medicine*, 96:116–122, 2019.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[5] `https://github.com/huggingface/pytorch-pretrained-BERT`

[12] Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. Publicly available clinical BERT embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics.

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

[14] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[15] SPFGH Moen and Tapio Salakoski2 Sophia Ananiadou. Distributional semantics resources for biomedical text processing. *Proceedings of LBM*, pages 39–44, 2013.

[16] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017.

[17] Anthony Rios and Ramakanth Kavuluru. Few-shot and zero-shot multi-label learning for structured label spaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

[18] Congzheng Song, Shanghang Zhang, Najmeh Sadoughi, Pengtao Xie, and Eric Xing. Generalized zero-shot icd coding, 2019.