Project Report, CS598 DL4H in Spring 2023

Manu Vinod Shesha
manuv3@illinois.edu

Group ID: 96
Paper ID: 187
Presentation link: https://youtu.be/CuFg1O4xUAw
Code link: https://github.com/manuv3/cs598-dl-project

1 Introduction

This work reproduces the idea and results in the original paper: *Automated ICD-9 Coding via A Deep Learning Approach*[3].

The original paper aims to automate the extraction of ICD-9 (Ninth Revision of International Classification of Diseases) codes from patient discharge summary, through application of general text processing technique called Document-to-Vector (D2V) in combination of Convolution Neural Network (CNN).

The extraction of ICD-9 codes from free-text Discharge summary is needed to do billing, as well as raising insurance claims with the provider. Today, the procedure to extract medical codes from patient discharge summary is largely a manual effort, undertaken by hospital's medical record department personnel. This has two problems:

- The process is very slow and inefficient, causing delay in the patient discharge process.
- The process requires specialized knowledge making it costly, and sometimes error prone.

The authors describe a novel approach to this problem through usage of DL techniques. They treat it as a task of multi-label classification (of ICD-9 codes). Specifically, the authors combine two approaches to produce vector embedding of documents, which are further used in multi-label classification:

- CNN (Convolution Neural Network) to discover, and extract **local features** in text. We can Intuitively, the local context of the text, like phrases describing a medical concept, should be important in deriving related ICD codes.
- D2V (Document to Vector) [2] embedding technique to capture the **global features** of the document. D2V extends upon Word2Vec, and (unlike CNN) takes order of words into account. It should be noted that D2V unsupervised learning approach.

2 Scope of reproducibility

The paper claims that proposed DL model combining Doc2Vec (D2V) and CNN, generates better embedding for documents, which in-turn perform better in multi-label classification task of identifying ICD-9 codes, when compared to traditional ML-based models.

More specifically, it claims:

- Higher Micro-average F1-score in multi-label ICD-9 classification task for the proposed model when compared to baseline models: Flat-SVM and Hierarchical-SVM.

Model	Precision	Recall	F1-score
flat-SVM	0.635	0.158	0.253
hierarchical-SVM	0.415	0.280	0.335
D2V+CNN	0.486	0.351	**.408**

All values are Micro-averaged.

- Each of the components: CNN and D2V provide important contribution to the overall accuracy of the model (, with CNN being more important). The model accuracy degrades significantly when either of these components are removed.

Model	Precision	Recall	F1-score
Only CNN	0.440	0.366	0.399
Only D2V	0.375	0.261	0.308
D2V+CNN	0.486	0.351	.408

All values are Micro-averaged.

2.1 Addressed claims from the original paper

- Higher Micro-average F1-score in multi-label ICD-9 classification task for the proposed model when compared to baseline models: Flat-SVM and Hierarchical-SVM.

- The model accuracy degrades significantly when either of the components: D2V or CNN are removed.
- In addition, we also build a different variation of the model which uses CNN with attention (instead of CNN with D2V), as part of additional work to verify if the performance can be improved.

3 Methodology

3.1 Model descriptions

The model is a DL network with two ”logical” components: Encoder and Classifier.

The Encoder generates fixed-length embedding for a given discharge summary document.This component consists of two ”logical” sub-components: D2V and CNN.

The ”D2V” sub-component first trains (as pre-processing step) Doc2Vec model to learn input document vectors of length `128`, in an unsupervised way. It then fine-tunes these vectors in a supervised way using a fully connected layer of `64` neurons, followed by a non-linear activation like ReLU (Rectified Linear Unit), and finally, a dropout layer which regularizes the output of fully-connected layer by stochastically dropping different dimensions of the output vector.

The ”CNN” sub-component uses Word2Vec as pre-processing step to build word vectors (of size `100`) for the whole vocabulary of the collective corpus of documents. For each document, all the vectors corresponding to the contained words, are ”stacked” as a matrix, to represent the given document. Batches of these document matrices are used as input to the CNN sub-component. This sub-component actually comprises of 3 single-layer multi-channel CNN models, similar to reference implementation in [1]. Three CNN models correspond to 3 kernel sizes of `3 (x 100), 4 (x 100), and 5 (x 100)`, with `64` output channels each. 2D CNN layer in each model is followed by a MaxPool layer to perform temporal pooling. The outputs of each of these CNN models are concatenated to generate the output vector per document of size `192 (3 models * 64 channels each)`.

The ouput vectors from the two sub-components (D2V and CNN) are concatenated to produce the final vector for each document in the batch. Ths final vector size is `256 (64 from DNN + 192 from CNN)`.

The Classifier performs multi-label classification of ICD-9 codes. This component consists of a fully connected layer with sigmoid activation. This layer generates the final output of size `6984 (total number of ICD-9 codes)`. Each dimension (representing an ICD-9 code) is assigned a probability by sigmoid activation.

Architecture is summarized in diagram 1.

The objective is to do end-to-end training of this model. Doc2Vec and Word2Vec models are trained unsupervised, in the pre-processing step. During the supervised learning phase, the above described components are trained for multi-label classification task using back-propagation. The gaol is to achieve similar or close enough performance in terms of micro-averaged F1-score, as achieved in original paper.

The model is medium complexity model in terms of the number of parameters to train.

CNN layer (3x100, 4x100, 5x100 sized 2D-kernels with 64 out-channels each)	76992
D2V Fine-tune layer (128 input dimension, 64 neurons)	8256
Fully-connected classification layer (256 input dimension, 6984 neurons)	1794888

An additional model (not part of the original paper) was created to study the impact of attention over plain CNN. The idea is inspired from CAML[4] architecture which leverages attention weights (along with other techniques) to derive ICD-9 codes from MIMIC-III discharge summary reports. The archiecture of this model consists of a single layer multi-channel 1D CNN, an attention mechanism, and finally, the classification component consisting of a fully connected layer with a Sigmoid classifier. Refer to diagram 2. The objective of this experiment was not to build a finely tuned model, but to check if the performance (in terms of micro f1-score) can be improved using attention mechanism.

The table below details the number of parameters in this additional model.

CNN layer (3-sized 1D-kernel with 100 in-channels and 50 out-channels)	25050
Attention matrix (6984 output states x 696 temporal hidden states)	349200
Fully-connected classification layer (50 input dimension, 1 output dimension)	51

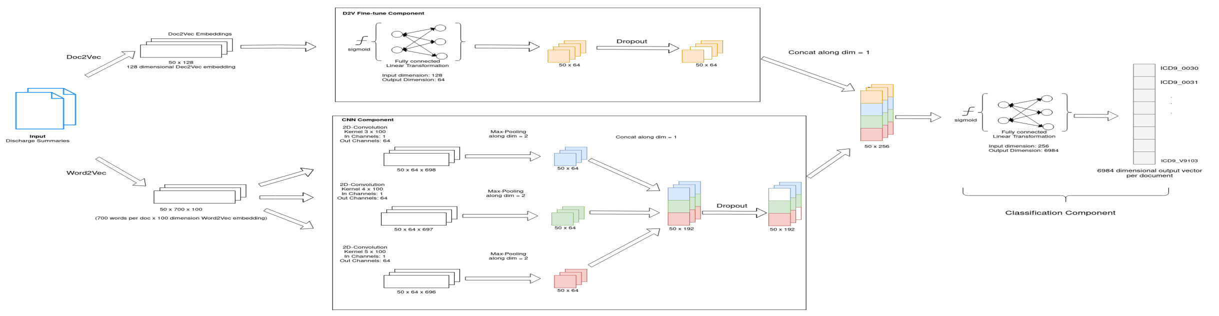

Figure 1: D2V + CNN Model Architecture

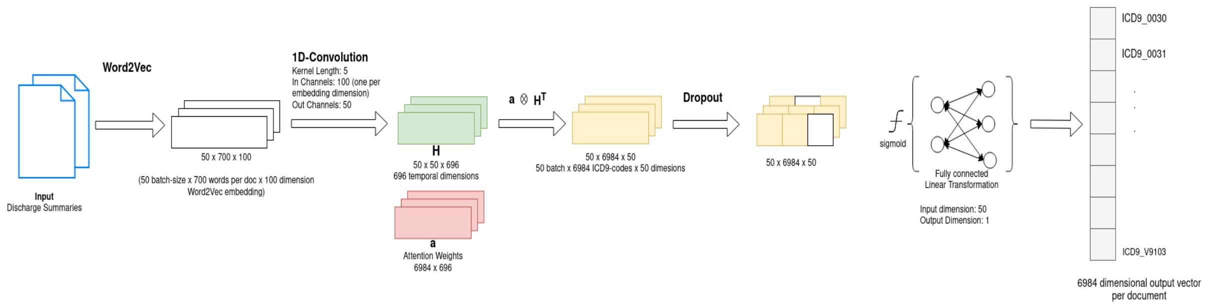

Figure 2: CNN + Attention Model Architecture

3.2 Data descriptions

The model is trained and tested on MIMIC-III MIMIC-III (Medical Information Mart for Intensive Care) database. The data is publicaly available, upon credentiating user identity on Physionet, and completing the mandatory training: Data or Specimens Only Research.

Specifically, following datasets in MIMIC-III were used:

- DIAGNOSES_ICD.csv: Each row in this file maps HADM_ID (Hospitalization ID) of a patient with a unique ICD9_CODE. We transform ICD9_CODE to one-hot encoding and group them per per HADM_ID to generate multi-hot encoding.

Total # codes	6984
Avg. # codes per patient	11
Max # of codes per patient	39
Min # of codes per patient	1

- NOTEEVENTS.csv: Each row maps HADM_ID (Hospitalization ID) with a TEXT (free text Discharge summary) field.

Total # discharge summary	52691
Avg. # words per discharge summary	1524
Max # of words per discharge summary	7980
Min # of words per discharge summary	9

3.3 Hyperparameters

For the reproducability, following hyperparamters were tuned:

- Probabilities for dropout layers in D2V and CNN components: The original paper found the optimum value of dropout probability to be `0.70`. But in this reproducibility work, the best performance was achieved at `0.20` dropout probability.

- Threshold value for binary classification: The value between 0 and 1, represents the threshold to assign class 0 or 1 for a given ICD-9 code, in the final classification layer. The original paper found the optimum threshold value as `0.20`. The same was found to be optimum in this reproducibility work. Lower threshold value was resulting in higher recall but lower accuracy. Similalry higher threshold resulted in higher accuracy but lower recall. At 0.20 threshold, accuracy and recall achieved harmonic balance to result in higher f1-score.

Many potential hyperparameters like number of units in fully connected layers (in D2V fine-tuning, and in classification component), convolution kernel sizes, Doc2Vec/Word2Vec embedding dimensions, and max number of tokens per document, were used as described in original paper.

Many ”potential hyperparameters like type of activation function (ReLU), learning rate for backward propagation (0.001), etc. were chosen based on ”conventional” wisdom and kept constant throughout, due to time constraint, and resource limits.

3.4 Implementation

The implemenation has been entirely done by the team. To the best of knowledge of the team, no implementation code by authors of original paper is available publicly.

The implementation uses Jupyter notebooks.

The notebook data_preprocessing.ipynb imports the MIMIC-III datasets (`DIAGNOSES_ICD.csv` and `NOTEEVENTS.csv`), applies necessary preprocessing and saves the final datasets.

- Filtering unnecessary columns from datasets. From `DIAGNOSES_ICD.csv`, only `HADM_ID, ICD9_CODE` columns are needed. From `NOTEEVENTS.csv`, only `HADM_ID, TEXT` columns are needed.
- Grouping `ICD9_CODE` per hospital visit, and representing them as multi-hot vector.
- Pickling the transformed Pandas Dataframes and saving on disk.

The notebook w2v_d2v.ipynb generates and saves following:

- Gensim Word2Vec model based vectors for all the tokens in the vocabulary of our corpus.
- Gensim Doc2Vec model based vectors for all the documents in our corpus.

(The above two notebooks are not needed to run the main notebook dl_model.ipnyb, as all the pre-processed data generated by these two notebooks have been made saved and made available to dl_model.ipnyb)

The notebook dl_model.ipynb trains and validates the DL models (as described in the model section). Following models have been included in this notebook:

- DL full model (D2V + CNN)
- D2V-only DL model (ablation)
- CNN-only DL model (ablation)
- CNN with attention DL model (additional work)

Link to source code: `https://github.com/manuv3/cs598-dl-project/tree/main/src`
Jupyter Notebook runnable on Google Colab Free Tier: `https://drive.google.com/file/d/1i3IcWbIW6hZZL73wg4DmeCjiz5Sa8_as/view?usp=sharing`

3.5 Computational requirements

Two coumptational environments were used in this work:

- Personal Machine:
  - CPU: `Intel® Core™ i7-10750H × 12 processor with 16.0 GiB Memory`
  - GPU: `NVIDIA GeForce GTX 1650 with 4.0 GiB Memory`
- Google Collab Free with ”standard” GPU support

The computational requirements for different steps are described below:

- MIMIC-III datasets import and preprocessing: All the steps were performed on personal machine on CPU.

Memory used	10 GiB
Time taken	3-4 minutes
Disk space needed for input datasets	3.75 GB
Disk space needed for output DataFrames	876 MB

- Word2Vec and Doc2Vec model training and vectors generation: All the steps were performed on personal machine. Gensim implementaion of Word2Vec and Doc2Vec support streamed source. So, there is no need to load the whole corpus in memory. However, there is no GPU support in Gensim yet.

Word2Vec embeddings generation time	37 minutes
Doc2Vec embeddings generation time	51 minutes
Memory used	1.5 GB
Disk space needed for output embeddings	70.1 MB

- Training the full Deep Learning model: The training could be easily performed on personal machine.

System RAM used	7.5 GB
GPU RAM used	1.5 GB
Epochs	20
Time taken to train and evaluate	20-30 minutes

  Multiple such runs were executed for hyper-parameter tuning.

- Training partial model with only D2V (for abalation): During the training of this model we just need to tune the parameters of 2 fully connected layers, which is not very computationally intensive. This was easily achieved on personal machine:

System RAM used	4.6GB
GPU RAM used	1.2GB
Epochs	40
Time taken to train and evaluate	56 minutes

- Training partial model with only CNN (for abalation): This involved training a single 2D CNN layer parameters and one fully connected layer parameters. This was easily achieved on personal machine:

System RAM used	7.1 GB
GPU RAM used	1.2 GB
Epochs	15
Time taken to train and evaluate	16 minutes

- Training a different model with CNN with attention (for additional work): This involved training a single layer of 1D multi-channel CNN, attention layer and final fully connected layer. Owing to the number of parameters (`374301` total), this required 6 GB of GPU RAM. The Google Collab Free version was sufficient to train this model.

System RAM used	7.1 GB
GPU RAM used	5.6 GB
Epochs	20
Time taken to train and evaluate	51 minutes

In general, personal machine was enough for the reproducing the models described in the original paper, unlike mentioned in the proposal where it was anticipated that the size of input samples and the number of parameters will require cloud computational environment.

4 Results

The results validate the main claim made by the paper, which is that the combined D2V + CNN model performs the multi-label ICD9 classification task with better micro-averaged f1-score, in comparison to traditional ML models: flat-SVM and hierarchical-SVM.

4.1 Result 1

DL model (based on D2V and CNN components), was reprodcued successfully, to generate document vectors which capture local and global characteristics of the text, and perform multi-label classification task of identifying the associated ICD9 codes.

Precision	Recall	F1-score
0.482	0.358	0.411

All values are Micro-averaged.

threshold: 0.20, CNN dropout: 0.20, D2V dropout: 0.30, epochs: 20

- The F1-score of `0.411` is better than flat-SVM (`0.253`) by `38%` and hieerarchical-SVM (0.335) by `18%`.
- The F1-score of the reproduced model is similar (infact, little better) than the one obtained by authors (`0.408`), in their work, while still maintaining comparable levels of accuracy, and recall.

4.2 Result 2

A partial DL model (without the CNN component), was reproduced successfully as part of abalation, to validate the importance of CNN component.

Precision	Recall	F1-score
0.362	0.257	0.300

All values are Micro-averaged.

threshold: 0.20, dropout: 0.10, epochs: 40

The F1-score of `0.300` is a bit lesser than corresponding score obtained by original authors (`0.308`) by `2.5%`. Nevertheless, this result clearly indicates the severe degradation from the performance of combined model, emphasizing the importance of CNN component.

4.3 Result 3

A partial DL model (without the D2V component), was reproduced successfully as part of abalation, to validate the importance of D2V component.

Precision	Recall	F1-score
0.467	0.350	0.400

All values are Micro-averaged.

threshold: 0.20, dropout: 0.20, epochs: 15

The f1-score of `0.400` of this partial model is very close to that obtained by original authors `0.399`. This result indicates the `2.6%` and `3.1%` degradation in micro f1-score and accuracy respectively, compared to combined model, which supports the claim by the authors that D2V component is important, but its contribution is lesser than CNN component.

4.4 Additional results not present in the original paper

An alternate model based on CNN + Attention was created to demonstrate the effectiveness of leveraging attention instead of max-pooling of CNN hidden states.

Precision	Recall	F1-score
0.524	0.367	0.432

All values are Micro-averaged.

threshold: 0.30, kernel size: 5, dropout: 0.10, epochs: 20

This "new" model did not undergo extensive hyperparameter tuning, but it was still able to surpass the performance of reproduced (main) model:

- The F1-score of `0.432` is better than the main model (`0.411`) by `5%`. It also significantly improved accuracy by `21%`.
- The downside is that this model required more than double the time to train and validate.

5 Discussion

The original paper is reproducible to a large extent. Based on the results achieved, perfomance claim on the main model has been reached by reproduced model. Moreover, it has been conclusively proved that the proposed model improves on the performance of baseline models by 15-36%.

The results also validates the overall approach taken:

- Maintain the quality and quantity of data, to ensure that some important features are not missed due to any data loss.
- Do not alter the main parameters of the model (as proposed by authors) like word/document embedding lengths, kernel sizes, count of activation units, number of layers, etc.
- Vary the hyperparameters like dropout rate, classification threshold etc. to achieve the desired performance metrics.

One weakness in this work was the lack of peer review (due to single member team), resulting in total absence of feedback loop. This caused some wastage of time and computational resources. In the first iteration of this work, the CNN component was coded as 1D convolution (with single input channel) over concatenated word vectors of total dimension of `70000`, per document. This was extremely inefficient process resulting in significant GPU RAM usage as well as long training duration of about `5` hours. In second iteration, this was changed to 2D convolution with multi-input channels, and this significantly reduced the memory requirements as well as training duration, both by over `90%`.

5.1 What was easy

- It is relatively easy to obtain the raw MIMIC-III data, and be able to extract almost same amount of samples as used by original authors, with basic pre-processing.
- It was easy to train Gensim Word2Vec and Doc2vec models. Gensim library is pretty stable, and intuitive to use, with decent documentation.

5.2 What was difficult

- The authors left some gaps in describing the architecture of the model. For example, the architecture of D2V fine-tuning layer was not clarified. The overall architecture diagram provided by authors is abstract and leaves out finer details like what pre-processing steps were taken before using the documents for training, which activation functions were used in DL layers, etc. So, it was an effort to bridge some missing pieces.
- The size of the model is relatively big, and lots of time was spent during training cycles. As a result hyperparameter tuning is limited by available time.

5.3 Recommendations for reproducibility

- The authors should provide more detailed architecture of the DL model. For example, where exactly are the dropout layers located in the model, what are the actual activation functions used, etc.

- The authors should give more detailed description of the implementation. For example, what was their training environment, what data pre processing steps did they perform, how many epochs did they run, etc.

6 Communication with original authors

No communication was attempted with original authors.

References

[1] Yoon Kim. Convolutional neural networks for sentence classification, 2014.

[2] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents, 2014.

[3] Min Li, Zhihui Fei, Min Zeng, Fang-Xiang Wu, Yaohang Li, Yi Pan, and Jianxin Wang. Automated icd-9 coding via a deep learning approach. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 16(4):1193–1202, 2019.

[4] James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. Explainable prediction of medical codes from clinical text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1101–1111, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.