CS598 Deep Learning For Health Care Final Project in Spring 2022

Ameet Deulgaonkar and Rahul KB
{ameetd2,fnuk2}@illinois.edu

Group ID: 160
Paper ID: 135, Difficulty: easy
Presentation link: https://www.youtube.com/watch?v=9w9ju7wyfj8
Code link: https://github.com/amyth18/CS598-Deep-Learning-Final-Project

1 Introduction

Predicting future health conditions of a patient based on medical history can be highly useful in providing timely, effective and efficient medical care. Many studies have been conducted in this regard, both using traditional machine learning and deep learning methods.

The paper[4] under discussion proposes a deep learning model trained on symptoms extracted from the patient discharge summaries as features to predict medical conditions.

The novelty of the method lies in the approach taken to represent symptoms. The symptoms are represented in 2 forms, a Word2Vec embedding to capture the symptom-symptom interaction and a uniquely computed TF-IDF like metric to capture the correlation strength between disease and symptom.

Experiments on the MIMIC III data set have shown this method performs better than state of the art methods in the area of disease inference using clinical texts, which we intend to reproduce in this study.

2 Scope of reproducibility

The main claim of the paper is that by using a clever symptom entity extraction method (for e.g. using MetaMap[8]) and a combination of 2 Bi-LSTM networks, where one is trained on Word2Vec representation of extracted symptoms and other on TF-IDF like metric, we get far better performance as opposed to using full-text representations like WordSeq or DeepLabeler[13].

2.1 Addressed claims from the original paper

Specifically, we will verify the following claims made in the paper.

- The proposed model and method produces better precision, recall and F-1 score compared to full-text models[13, 2] that use all of the text from discharge summary, both using macro and micro averaging strategies as this is a multi-label classification problem.
- The proposed model i.e. using a combination of 2 Bi-LSTM networks, one trained on Word2Vec and one trained on TF-IDF representation of symptoms performs better than using just one of the representations as measured by the same metrics mentioned above.

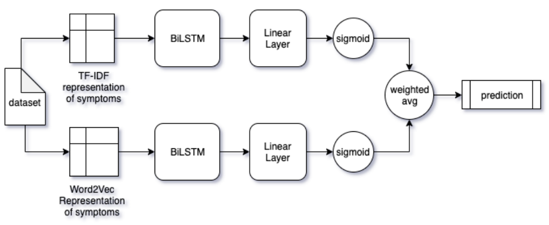
3 Methodology

We implemented the proposed model from scratch as the authors had not shared their code. The following sections provide details about our current implementation.

3.1 Model descriptions

The objective of the model is to predict diseases based on the patient discharge summaries in natural English text. The focus of the paper is on predicting top#50 (or top#100) most frequently diagnosed diseases in a hospital admission. The following diagram depicts the high level architecture of the proposed model:

Figure 1: Model Architecture



The model consists of the following components.

1. A Word2Vec embedding generator. In the current implementation we are using Gensim 4.1.2[1]. to generate word embedding using

all the discharge summaries in the data set. Please refer to section 3.2.1 for more details.

2. A symptom extraction module that processes natural language text in discharge summaries and extracts only the terms and phrases that constitute a symptom. Please refer to section 3.2.1 for more details.

3. A TF-IDF vector generator, that produces TF-IDF like vector representation for the symptoms extracted above. Please refer to section 3.2.1 for more details.

4. Two Bidirectional LSTM models. One trained trained on the word embedding and other trained on TF-IDF vectors. The output from these models is passed through a linear layer followed by sigmoid activation layer and then a weighted average is computed between the 2 output vectors to produce the final prediction (a multi-hot vector encoding for top#50 or top#100 diseases).

Summary of model parameter settings

No of trainable parameter	325700
No of layers in BiLSTM	1
size of hidden vector	100
size of word embedding	128

3.2 Data descriptions

The model was trained and evaluated on MIMIC-III data set version 1.4[5]. The data set was downloaded from `https://physionet.org/` after completion of CITI training. The study required us to focus predominantly on the following tables and fields.

1. NOTEEVENTS.csv
   (a) SUBJECT_ID represents the unique ID for a patient.
   (b) HADM_ID represents the unique ID of admission for a given patient.
   (c) TEXT contains the discharge summary for the admission.

2. DIAGNOSES_ICD.csv
   (a) SUBJECT_ID and HADM_ID are same as above.
   (b) ICD9_CODE contains one ICD9 code for a diagnosis made in the admission. The rows SUBJECT_ID, HADM_ID and ICD9_CODE are repeated for each unique diagnosis.

3.2.1 Data Processing

We join the above tables on SUBJECT_ID, HADM_ID and perform the following data filtering and transformation steps:

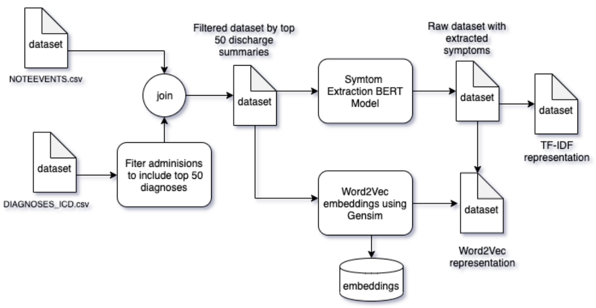
1. We are interested in predicting top#50 (or top#100) most frequently occurring diseases so we select records that contain at least one diagnosis in the top#50 (or top#100) category and discard the rest.

2. We then extract the symptoms from the clinical text. The paper proposes to use a tool called MetaMap[8] for this purpose but we were not able to get this tool working. Instead we used a BERT based uncased clinical NER model[11] to extract symptoms. Further, this model was trained on [i2b2 (now n2c2) data set]: `https://n2c2.dbmi.hms.harvard.edu/`. We are thankful for the huggingface repository and it's contributor *samrawal* for making the model easily accessible.

3. We then transform the extracted symptoms and represent them in 2 vectors formats i.e. a Word2Vec format and a TF-IDF format. To represent symptoms using Word2Vec format we trained a Word2Vec model using Gensim library prior to this step using all the discharge summaries.

The following diagram depicts the entire data processing pipeline.

Figure 2: Data Processing Pipeline


dataset
NOTEEVENTS.csv
dataset
DIAGNOSES_ICD.csv
Fiter adminisions to include top 50 diagnoses
join
Filtered dataset by top 50 discharge summaries
dataset
Symtom Extraction BERT Model
Word2Vec embeddings using Gensim
embeddings
Raw dataset with extracted symptoms
dataset
dataset
TF-IDF representation
dataset
Word2Vec representation

Please refer to the notebook in the shared GitHub repository for more details on the implementation of all data pre-processing steps.

The following table describes the final schema of the data set after all the data processing steps:

Dataset Schema	
Feature	Description
SUBJECT_ID	Identifier of Patient
HADM_ID	Identifier of Admission
SYMPTOMS	vector of symptoms e.g. ['fever', 'cold', ...]
ICD9_CODES	vector of IC9 Codes e.g.['42731', '2762',...]

3.2.2 Dataset Statistics

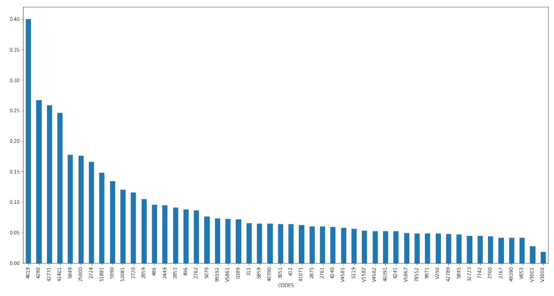
The following table captures important statistics related to our data set:

Table 1: Data set Statistics

No of Records (size)	55988
Discharge Summary: Avg # Words	949
Discharge Summary: Max # Words	5179
Discharge Summary: Min # Words	6
Discharge Summary: 75% # Words	1259
ICD9 Codes: Avg #	38
ICD9 Codes: Max #	166
ICD9 Codes: Min #	7
ICD9 Codes: 75%	50

The following plot shows the frequency of each of top#50 ICD9 codes in hospital admissions. We notice that there is skew in the distribution of disease codes.

Figure 3: ICD9 Code Frequency



3.3 Hyperparameters

We used the same hyper-parameter settings as mentioned in the paper by the authors. We did not perform any further hyper-parameter tuning on our own.

Hyper-parameter settings we used are as follows:

learning rate	0.001
batch size	400
# hidden layers in LSTM	1
# nodes in hidden layer	100
train-val-test split	80-10-10
optimizer	adam
loss function	binary cross entropy
classification threshold	0.2

We did not find any recommendation for training Gensim model for word embedding, so we went ahead with most common recommendations.

min_count	5
embedding size	100 (default)
sample	1e-5

3.4 Implementation

We have implemented all the methods from scratch because the authors had not made their code publicly available . All our code including data preprocessing, model definition, training and evaluation for our main model can be found in the following notebook.

`https://github.com/amyth18/CS598-Deep-Learning-Final-Project/blob/main/Main_Model.ipynb`

We have used the Google's Colab environment for our experiments. We have used the following libraries in our implementation.

pytorch[10]	model development
transformers	Access pre-trained models
pandas[9]	data processing
texthero[12]	data processing
datasets	data processing
gensim	word embedding
scikit-learn	model evaluation

3.5 Computational requirements

Our estimation of computational requirements before the implementation were as follow:

Main Memory	50 GB
GPU Memory	5 GB
GPU time per epoc	300 seconds

We used the Google's Colab pro+ environment for our development. The configuration of our environment is as follows:

Total Main Memory	51.01 GB
Total GPU Memory	15.9 GB

Our resource utilization utilization profile (includes both data pre-processing and training stages) is as follows:

Average GPU Memory Used	39.68 MB
Average Main Memory Used	10.39 GB
Average GPU Time (per epoc)	35.34 sec
Total GPU Time	3533.87 sec

Some observations on our utilization rate.

1. The main (CPU) memory usage is high because the data processing steps are quite data heavy, as we deal with heaps of raw text at this stage.

2. The GPU memory usage is low because our batch size is small providing scope for increasing it further and speeding up training. This could also be because we have extracted only symptoms tokens from the discharge summary text.

4 Results

The following tables show a comparison of the performance of our implementation of the proposed model with the results published by the authors in the paper. The metrics used to evaluate the performance are precision, recall, F1-Score and AUC using 2 different averaging strategies i.e "micro" averaging and "macro" averaging as this is a multi-label classification problem[6].

The following tables provide a comparison of our results to the author's results.

Metrics Using Micro Averaging				
	MiP	MiR	MiF1	AUC
Author	0.496	0.564	0.528	0.859
Our	0.453	0.547	0.496	0.883
Deviation(%)	-8.26	-2.83	-5.87	2.79

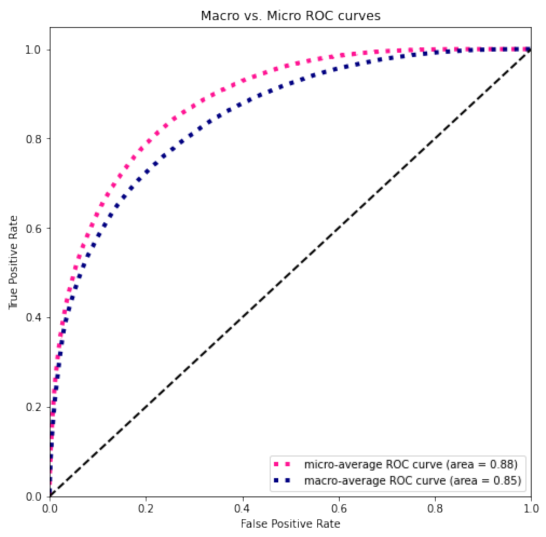
Metrics Using Macro Averaging				
	MaP	MaR	MaF1	AUC
Author	0.464	0.463	0.448	0.773
Our	0.410	0.475	0.422	0.848
Deviation(%)	-10.56	3.23	-4.91	9.83

We observe that the F1-Score (most importantly) for our model has the least deviation (6% at micro and 5% at macro level) from the numbers published by authors, indicating our implementation is reasonably inline with proposal in the paper.

We do a see a noticeable deviation in precision numbers (up to 10% under performance). We provide our detailed analysis on the plausible causes for this in section 5.

The following chart shows the ROC curves for our model for both macro and micro averaging strategies.

Figure 4: ROC Curves



4.1 Result 1

The goal of this experiment was to validate the claim that the proposed model and methods perform better than the models that use the full clinical text (e.g. DeepLabeler[13]).

To perform this experiment we had to implement the DeepLabeler model from scratch. Our implementation of this model can be found at `https://github.com/amyth18/CS598-Deep-Learning-Final-Project/blob/main/Other_Baseline_Models.ipynb`

Summary of DeepLabeler model parameter settings

# of trainable parameter	246674
# of Conv2d Layers	1
# of Conv2d operations	64
Max Pool Size	4
dropout	0.75

The following table provides a comparison of the performance of proposed model with full text model (DeepLabeler).

Comparison with Full Text Model (Micro)				
	MiP	MiR	MiF1	AUC
Proposed Model	0.453	0.547	0.496	0.883
Full Text	0.435	0.568	0.493	0.747

Comparison with Full Text Model (Macro)				
	MaP	MaR	MaF1	AUC
Proposed Model	0.410	0.475	0.422	0.848
Full Text	0.395	0.472	0.419	0.695

The results do indicate that proposed model performs better than full text model, which is inline with the results published by the authors as well.

4.2 Result 2

The goal of this experiment is to validate the claim that the proposed model which uses 2 representation of the extracted symptoms (word2vec and tf-idf) performs better than using just one of the representations with a BiLSTM model.

The following table provides the performance comparison of the proposed model with a BiLSTM + TF-IDF only representation and BiLSTM + Word2Vec only representation.

Micro Averaging				
	MiP	MiR	MiF1	AUC
Proposed Model	0.453	0.547	0.496	0.883
TF-IDF only	0.419	0.561	0.480	0.874
Word2Vec only	0.435	0.542	0.482	0.871

Macro Averaging				
	MaP	MaR	MaF1	AUC
Proposed Model	0.410	0.475	0.422	0.848
TF-IDF only	0.390	0.466	0.397	0.835
Word2Vec only	0.387	0.440	0.382	0.828

The proposed model clearly outperforms the baseline models on the macro numbers but we only notice a marginal improvement on the micro numbers. However, this is inline with results published in the paper.

The code for these experiments can be found on github at https://github.com/amyth18/CS598-Deep-Learning-Final-Project/blob/main/Main_Model.ipynb in section titled ”Evaluation of Baseline Models”.

4.3 Summary

The charts in Figure: 5 and Figure: 6 summarize the comparison of the proposed model to all the baseline models discussed in the previous sections.

Figure 5: Metrics (Macro Averaging)

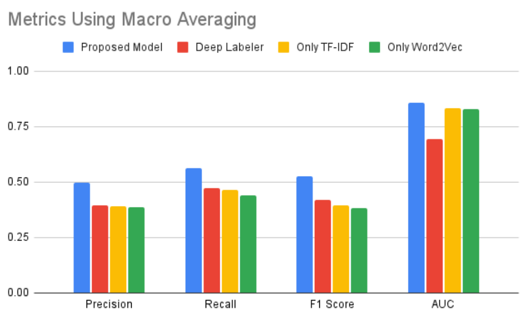
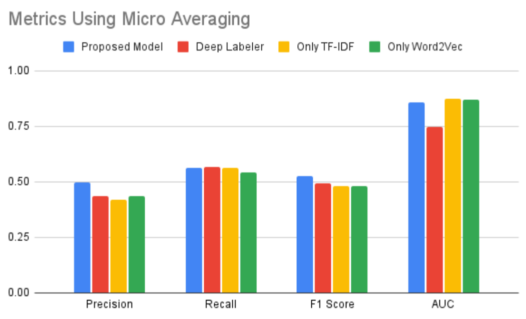


Figure 6: Metrics (Micro Averaging)



4.4 Additional results not present in the original paper

Further, we attempted to study if using a pre-trained transformer network could out-perform the proposed model on predicting diseases. In particular, we explored emilyalsentzer/Bio_ClinicalBERT from the HuggingFace repository [14]. The Bio_ClinicalBERT model was trained on the full discharge summary text from MIMIC III [5]. Our model architecture is a BERT [3] model initialised with model parameters of BioBERT [7].

The code for these experiments can be found on GitHub at https://github.com/amyth18/CS598-Deep-Learning-Final-Project/blob/main/Additional_Experiments.ipynb. The following table captures the performance of this model.

Bio_ClinicalBERT Model Performance				
	Precision	Recall	F1	AUC
Micro Avg	0.415	0.545	0.471	0.859
Macro Avg	0.355	0.440	0.363	0.810

These results further validate the author's claims, that their proposed model performs better than the models that use full clinical text.

5 Discussion

At a high level the results we obtained from our experiments do support the conclusions drawn by the authors in the paper. As seen in the previous section the proposed model performs better than the baseline models on all metrics.

Our numbers were not 100% aligned with authors numbers, especially our precision numbers had a much larger deviation from author's numbers (ours being on the smaller side). We attribute this to the fact that the model relies heavily on the robust extraction of symptom concepts from the discharge summary. The paper proposes to use MetaMap[8] utility for this, but we were not able to set it up and gain expertise on this tool in the given time. Instead, we used a pre-trained BERT model on clinical text as an alternative method to extract symptoms. Our experiments on sample discharge summaries proved this to be an effective alternative, but we suspect this tool could be lacking some capabilities compared to the MetaMap[8] utility hence the drop in accuracy numbers. We did reach out to authors on this issue but failed to get a response.

5.1 What was easy

We could accomplish the following aspects of the reproduction study relatively easily:

1. Though the authors had not shared their code, we could easily re-implement the model from scratch based on the model description in the paper. The architecture diagram was of great help.

2. The methodology to generate feature vectors (i.e. TF-IDF representations and Word2Vec representation) for the extracted symptoms from discharge summary text was clearly articulated and we could easily re-implement the same. However we faced issues in symptom extraction which we discuss in next section.

3. The experimental setup to evaluate performance of proposed model against baseline models was also very clearly documented so we could replicate performance without much hassles.

4. The most important aspect of any machine learning project is availability of data to train and evaluate models. The paper used the widely available MIMIC III data set for all its experiments, something we could easily get access to after completion of necessary training.

5.2 What was difficult

We found the following aspects of the reproduction study relatively difficult:

1. The author's used a tool called MetaMap[8] to extract symptoms concepts from the discharge summaries. Getting this tool installed and making it work for the volume of data we had was a challenge. There were several reasons for this, ranging from the tool being written in an older technology stack to lack of clear documentation of its APIs. The authors did not provide enough details on how exactly they used this tool (e.g. the APIs used) to extract symptom concepts.

2. Also certain data pre-processing steps were not very clear in the paper, for e.g. the authors used only the first 50 symptoms extracted from the discharge summary, but we noticed that using such small number of symptoms actually resulting in model under-fitting the data. It was not clear if there were additional selection criterion employed.

3. The volume of data (i.e. number discharge summaries) was relatively large and data exploded even further when we represented words in Word2Vec format. Even though we considered only discharge summaries with only 50 most commonly occurring diagnoses, the size was still high resulting in longer training times. Eventually, we had to switch to Google Colab Pro+ environment with GPUs and large memory instances to train and evaluate our models in reasonable time.

5.3 Recommendations for reproducibility

We request the authors to add some more details on the data pre-proccessing procedures, especially the step of extracting symptom concepts from discharge summary using MetaMap[8] tool as this is an extremely crucial element towards running the experiments and getting the results that are inline with the author's results.

To students/researchers trying to reproduce this research and are struggling with MetaMap[8] to extract symptoms, we recommend experimenting with pre-trained BERT models (e.g. explore HuggingFace) on clinical text.

6 Communication with original authors

We reached out to authors via the correspondence email address provided in the paper for some clarifications on data processing steps and model architecture but we did not succeed in getting a response from them.

References

[1] *Gensim Word2Vec model API documentation.*

[2] Siddhartha Nuthakki et al. 2020. Natural language processing of mimic-iii clinical notes for identifying diagnosis and procedures with neural networks.

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

[4] Ying Yu Yaohang Li Fang-Xiang Wu Min Li Donglin Guo, Guihua Duan. 2020. A disease inference method based on symptom extraction and bidirectional long short term memory networks.

[5] Pollard T. Mark R. Johnson, A. 2016. Mimic-iii clinical database (version 1.4).

[6] scikit learn. *Plotting ROC curves for multi-label classification models.*

[7] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*.

[8] National Library of Medicine. Metamap: A tool for recognizing umls concepts in text.

[9] Pandas. *Pandas API documentation.*

[10] Pytorch. *Pytorch API documentation.*

[11] samrawal. Pretrained bert model for clinical ner.

[12] Texthero. *Texthero API documentation.*

[13] Min Li; Zhihui Fei; Min Zeng; Fang-Xiang Wu; Yaohang Li; Yi Pan; Jianxin Wan. 2019. Automated icd-9 coding via a deep learning approach.

[14] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing.