CS598 DL4H Reproducibility Project: *Real-world Patient Trajectory Prediction from Clinical Notes Using Artificial Neural Networks and UMLS-Based Extraction of Concepts*

Nicholas DaRosa
ndarosa2@illinois.edu
Group ID: 33
Paper ID: 111
Presentation link: https://youtu.be/pDRzoonVsh8
Code link: https://github.com/ndarosa2/CS598_DL4H_Project

1 Introduction

Clinical notes in electronic health records are unstructured pieces of text that are of increased interest to researchers for their potential in training neural networks for tasks such as mortality prediction and diagnoses code prediction. As a result of this interest, there is much research being conducted into the the optimal method for processing clinical notes before they are used as input into a neural network.
This paper introduces a new specific approach of using the unstructured text in clinical notes to create a set of Unified Medical Language Systems (UMLS) concepts using QuickUMLS [1-4]. This is innovative because previous works emphasize the use of word embeddings and raw sets of extracted concepts instead of generating a set of structured UMLS concepts. These concepts are then used as inputs to different deep learning models such as feed forward neural networks and recurrent neural networks for diagnoses code prediction, mortality prediction, and readmission prediction.

2 Scope of Reproducibility

The central claim of this paper is that using the new approach of a set of structured UMLS concepts in the form of Concept Unique Identifier (CUI) codes as input into a feed forward network (FFN) model for diagnoses prediction and into a gated recurrent unit (GRU) model for mortality prediction performs better than using word embeddings and raw sets of extracted concepts as input to models for diagnoses code prediction and mortality prediction. In addition, secondary claims of the paper are listed in the following subsection.

2.1 Addressed claims from the original paper

1. Using Type Unique Identifiers (TUI) set Beta (85 types) and a threshold similarity of 0.9 creates the dataset with the best parameters for diagnoses prediction.

2. Using clinical notes, which are used to generate a set of structured UMLS concepts in the form of CUI codes, and diagnosis code data, in the form of Clinical Classification Software (CCS) codes, as input into the models performs better than only using clinical notes as input or only using diagnosis code data as input for diagnoses code prediction.

3. With clinical notes (in the form of CUI codes) as input, using a feed forward network (one hidden layer with 10,000 neurons) for predicting diagnoses performs better than using a single layer of GRU (gated recurrent unit) with a hidden size of 200 or a single layer of LSTM (long short-term memory) with a hidden size of 200.

4. Using only clinical notes (in the form of CUI codes) as input into the models performs better than only diagnoses code data (in the form of CCS codes) for diagnoses code prediction, mortality prediction, and readmission prediction.

5. As stated above, the central claim of this paper is that using a set of structured UMLS concepts (in the form of CUI codes) as input into the models performs better than word embeddings and raw sets of extracted concepts (which were used in previous works) as input for diagnoses and mortality prediction.

3 Methodology

3.1 Model descriptions

Three model architectures were used for predicting diagnoses codes, mortality, and hospital readmission. One model is a feed forward network, while

the other two models are recurrent neural networks (LSTM and GRU).

1. Feed Forward Network (FFN): The FFN has an input layer (size ranges from 16,732 to 33,752 depending on number of CUI codes), one single fully connected hidden layer with a size of 10,000 and a dropout fraction of 0.5, and an output layer. ReLu was used as the activation function for the hidden layer while sigmoid was used as the activation function for the output layer. The size of output layer is 269 for diagnoses prediction, 1 for readmission prediction, and 3 for mortality prediction. Binary Cross-Entropy loss is used for the loss function and stochastic gradient descent was used for the optimizer. The number of parameters in the FFN greatly varies depending on the input and output layers; however, as an example, for diagnoses prediction with Dataset *D*, the total number of parameters in the model is 235,330,269.

2. Recurrent Neural Network (RNN): Both versions (LSTM and GRU) of the RNN used an input layer, one single hidden RNN layer (for GRU and LSTM hidden layer size is 200 by default) and an output layer. Similar to the FFN, the size of output layer is 269 for diagnoses prediction, 1 for readmission prediction, and 3 for mortality prediction. ReLu was used as the activation function for the hidden layer while sigmoid was used as the activation function for the output layer. Binary Cross-Entropy loss is used for the loss function and Adam was used for the optimizer. In the original paper, the LSTM model was only used for diagnoses prediction, while the GRU model was used for diagnoses prediction, readmission prediction, and mortality prediction. The number of parameters in the RNNs greatly varies depending on the input and output layers; however, as an example, for diagnoses prediction with Dataset *D*, the total number of parameters in the GRU model is 14,133,069.

3.2 Data descriptions

The original paper uses the publicly available Medical Information Mart for Intensive Care III dataset (MIMIC-III) [5]. The MIMIC-III dataset consists of data collected between 2001 and 2012 for approximately 48,520 patients. The data is anonymized for each patient. Since only patients that have at least two admissions that have corresponding clinical notes were considered for this work, the 48,520 patient dataset was reduced to 7,314 patients. Furthermore, each admission contains a corresponding ICD-9 diagnoses code sequence. On average, there are approximately 13 diagnoses codes per admission for a patient.

I was granted access to the MIMIC-III dataset through PhysioNet. From MIMIC-III, ADMISSIONS.csv, NOTEEVENTS.csv (clinical notes), and DIAGNOSES_ICD.csv are used to generate the datasets. The clinical notes (NOTEEVENTS.csv) are processed into medical concepts using tools that leverage the concepts cataloged in the UMLS Metathesaurus. I was granted a UMLS license through the National Library of Medicine. QuickUMLS is used to extract the medical concepts from the clinical notes, which uses a similarity threshold (default is 0.7 where 1.0 is a perfect match) that compares the clinical notes string to the cataloged concepts to determine which concepts are extracted. Each medical concept has its corresponding Concept Unique Identifier (CUI) code. The threshold can be increased to decrease the number of candidate CUIs. The number of candidate CUIs was also reduced by only allowing CUIs to be candidates whose concept type/category are listed in the two Type Unique Identifier (TUI) lists, named Alpha (47 types) and Beta (85 types). As a result of this processing, each admission has a corresponding set of CUI codes.

The properties of the datasets used in the original paper and if they were successfully reproduced is present in Table 1. Unfortunately due to budget and computational requirements, Dataset B was not able to be reproduced. Furthermore, the data within DIAGNOSES_ICD.csv is used as an optional additional input into the models. The ICD-9 codes contained in DIAGONOSES_ICD.csv are converted into Clinical Classification Software (CCS) codes, which reduces reduces the ICD codes from a set of approximately 15,000 unique codes to just 283 unique CCS codes.

Table 1: The dataset configurations used in the original and reproduction experiments.

Dataset	Threshold	TUI List	# of CUI Codes	Comment
A	0.7	Alpha (47 types)	33,752	Successfully reproduced
B	0.7	Beta (85 types)	39,049	Failed to reproduce
C	0.9	Alpha (47 types)	16,723	Successfully reproduced
D	0.9	Beta (85 types)	22,820	Successfully reproduced

The datasets were split with 20% of the dataset used for testing while the other 80% was used for training.

3.3 Hyperparameters

A combination of the default hyperparameters (e.g. number of epochs, learning rate, optimizer method) as determined by the original paper's python scripts and as specified in the original paper itself were used for reproduction.

In addition to the description given in Section 3.1, for the FFN, by default the batch size is 100 samples, the learning rate is 0.01, and the number of epochs is 5000. Meanwhile, for both the LSTM and GRU models, by default the batch size is 10 samples, the learning rate is 0.01 (diagnoses and mortality prediction) or 0.001 (readmission prediction), and the number of epochs is 1500 (diagnoses prediction) or 100 (mortality or readmission prediction).

3.4 Implementation

Existing code was used to reproduce the results. For the preprocessing of the data, the documentation and code given in the original paper's GitHub was used. Since QuickUMLS is required to generate the UMLS concepts, the original paper's documentation refers to QuickUMLS's instructions on how to install QuickUMLS.

For training the models, the code and associated documentation in the original paper's Github was used. Only minor modifications to the original code was necessary to successfully run the models. One minor modification to the code was to add lines for calculating the number of model parameters. Another modification was the addition of the Pytorch function *torch.no_grad* to the evaluation section of some of the Pytorch scripts. This addition was needed to increase computation speed and to prevent "RuntimeError: CUDA error: out of memory" errors during the evaluation phase of models such as using GRU for readmission prediction.

Links to the original paper's GitHub and the QuickUMLS GitHub are provided in the References section.

3.5 Computational requirements

The computational requirements to fully reproduce this paper vary greatly. Given the local computer's specifications of a RTX 3080 12GB, 6-core/12-thread 5600X, 16GB of DDR4 RAM, and 500GB SATA SSD, the first data processing step of preprocessing NOTEEVENTS.csv using noteEvents_preproc.py, completed in two hours versus the predicted four hours as mentioned in the original paper's GitHub. Furthermore, the second preprocessing step of splitting the output of noteEvents_preproc.py into files of 50MB each completed in approximately 20 minutes versus the predicted one hour. Based on these results, the 6-core/12-thread 5600X and the 16GB of RAM were more than enough for these processing steps.

Unfortunately, the local desktop computer was not able to complete the next preprocessing step within a reasonable time, so a virtual machine was needed with 256GB of RAM and 32 threads. The original paper's GitHub documentation mentions the step for generating CUI codes (quickUMLS_getCUI.py) should take in the range of hours to three days. Not mentioned unfortunately is that the RAM required for this processing step is around 50-70 GB (assuming 32 threads are active) for generating datasets with a threshold of 0.9 (i.e. Datasets C and D) and up to 180GB when generating datasets with a threshold of 0.7 (i.e. Datasets A and B). Generating Dataset C took approximately 8 hours to complete while generating Dataset A required 41 hours of real time with 32 threads at 100% utilization.

For modeling purposes, the FFN were typically trained with 5000 epochs and 5 fold cross validation. On a RTX 3080 12GB, each epoch required approximately 0.5-1 second of real time depending on the number of parameters. Meanwhile, the GRU and LSTM were trained with typically 1500 epochs with each epoch requiring 1-4 seconds depending on the number of parameters. The linux command "time" was used for recording real, user, and system time. These outputs for most of the model training and some of the data preprocessing steps are in the file Raw Results in the associated GitHub.

4 Results

Overall, the claims and associated results presented in the original work have been successfully reproduced. A comparison of the original work's results and the reproduction results are presented below. In addition to the tables provided, the raw output for the statistics and the time required for the experiments are provided in the file Raw Results in this project's associated Github.

4.1 Result 1

The first claim that using TUI set Beta and a threshold of 0.9 creates the dataset with the best parameters for diagnoses prediction is supported by the reproduction results. As in the original paper, the model runs that were completed used the FFN architecture with default hyperparameters with Dataset A, C, or D as input for diagnoses prediction. The results comparing the original paper's results and the reproduction's results are provided in Table 2. The table displays the precision metrics, recall metrics, and the Area Under the Receiver Operating Characteristic Curve (AUC-ROC).

Precision at N (P@N) is the number of correct codes in the top-N recommendations divided by N.

Recall at N (R@N) is the number of correct codes in the top-N recommendations divided by the number of correct codes.

Table 2: Diagnoses prediction metrics Precision@, Recall@, and AUC-ROC computed over the four dataset configurations using a FFN. Reproduction results are compared to original paper results.

Source	Dataset	P@1	P@2	P@3	R@10	R@20	R@30	AUC-ROC
Original	A	0.732	0.671	0.624	0.382	0.563	0.677	0.901
Reproduction	A	0.716	0.652	0.607	0.375	0.557	0.673	0.906
Original	B	0.742	0.679	0.631	0.387	0.570	0.683	0.905
Reproduction	B	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Original	C	0.729	0.672	0.627	0.385	0.567	0.681	0.903
Reproduction	C	0.738	0.679	0.631	0.387	0.570	0.683	0.909
Original	D	0.750	0.688	0.638	0.392	0.576	0.689	0.911
Reproduction	D	0.750	0.686	0.637	0.391	0.575	0.688	0.911

As shown in Table 2, for the datasets that were successfully reproduced, all seven metrics for the reproduction results are within 2% of the original paper's results with most being within 1% of the original results. Note that in both the original paper and in the reproduction, only the results for the FFN are shown since the FFN architecture consistently performed better than the LSTM and GRU architectures for diagnoses prediction.

As a result of this small difference between the results of the original and reproduction which is likely due to randomness, the reproduction results support the original paper's results. Furthermore, since Dataset D has the greatest value for all metrics in both the original and reproduced results, the paper's claim that using TUI set Beta and a similarity threshold of 0.9 creates the dataset with the best parameters for diagnoses prediction is in agreement with reproduction results.

4.2 Result 2

The reproduction results support the second claim that using clinical notes (in the form of CUI codes) and diagnosis code data (in the form of CCS codes) as input into the models performs better than only using clinical notes as input or only using diagnosis code data as input for diagnoses code prediction.

Table 3: Metrics comparison for inputs CUI codes only, CCS codes only, and CUI codes combined with CCS codes for diagnoses prediction using the FFN model with default hyperparameters and Dataset D.

Input	P@1	P@2	P@3	R@10	R@20	R@30	AUC-ROC
Orig. CUI	0.750	0.688	0.638	0.392	0.576	0.689	0.911
Repr.CUI	0.750	0.686	0.637	0.391	0.576	0.688	0.911
Orig. CUI+CCS	0.778	0.723	0.677	0.414	0.597	0.706	0.913
Repr. CUI+CCS	0.778	0.722	0.675	0.414	0.597	0.706	0.916
Orig. CCS	0.728	0.679	0.640	0.390	0.561	0.668	0.905
Repr. CCS	0.728	0.690	0.647	0.396	0.571	0.667	0.906

As seen in Table 3, all seven metrics for the reproduction results are within 2% of the original's results with most being within 1% of the original results. Also evident in Table 3, is that in agreement with the original results, using CUI codes and CCS codes as input improves the Precision@ by 4% and the Recall@ by 3% compared to only using CUI codes as input. This increase in Precision@ and Recall@ is even greater when compared to only using CCS codes as input.

As a result of this small difference between the results of the original and reproduction which is likely due to randomness, the reproduced results support the original paper's results. Furthermore, since the run that used both CUI data and CCS data as input has the greatest value for all metrics in both the original and reproduced results, the paper's claim is in agreement with reproduction results.

4.3 Result 3

The third claim that with using CUI codes generated from clinical notes as input, using a FFN model for predicting diagnoses performs better than using a single layer of GRU with a hidden size of 200 or a single layer of LSTM with a hidden size of 200 is supported by the reproduction results.

Table 4: Comparison of the two RNN architectures (LSTM and GRU) against the FFN architecture for diagnoses prediction using Dataset D as input.

Architecture	P@1	P@2	P@3	R@10	R@20	R@30
Orig. LSTM	0.5874	0.5483	0.5142	0.3117	0.4687	0.5764
Repr. LSTM	0.5622	0.5217	0.4895	0.3056	0.4612	0.5677
Orig. GRU	0.5404	0.5085	0.4798	0.3001	0.4584	0.5712
Repr. GRU	0.5248	0.5004	0.4641	0.2880	0.4493	0.5648
Orig. FFN	0.750	0.688	0.638	0.392	0.576	0.689
Repr. FFN	0.750	0.686	0.637	0.391	0.576	0.688

As demonstrated in Table 4, all recall and preci-

sion metrics for the reproduction results are within 1% of the original paper's results for the FFN model and are within 4% for the GRU and LSTM models. The increase in percent difference between the original results and reproduction results for the LSTM and GRU models is likely due to 5 fold cross validation being used for FFN model but it was not used for the GRU model or LSTM model. This lack of 5 fold cross validation follows what was done in the original paper. Also shown in Table 4 is that, in agreement with the original results, the FFN performed better than both the LSTM model and GRU model across all metrics. According to reproduction results, on average the FNN performed better than the LSTM and GRU in terms of Precision@ by 27% and 32% and in terms of Recall@ by 21% and 25%, respectively.

As a result of this small difference between the results of the original and reproduction, the reproduction results support the original paper's results. Moreover, since the FFN has the greatest value for all metrics in both the original and reproduction results, the original paper's claim is in agreement with reproduction results that using the FFN model is best for diagnoses prediction.

4.4 Result 4

The reproduction results support the fourth claim that using only clinical notes (in the form of CUI codes) as input into the models performs better than only diagnoses code data (in the form of CCS codes) for diagnoses code prediction, mortality prediction, and readmission prediction is support.

Table 5: Comparison of only using CUI codes as input and only using CCS codes as input into the models. As in the original paper, note that only the best performing architecture for each task is shown.

Task	Metric	Orig CUI	Repr CUI	Orig CCS	Repr CCS
Diagnoses Pred. (FFN)	P@1	0.750	0.751	0.728	0.728
	P@2	0.688	0.686	0.679	0.690
	P@3	0.638	0.637	0.640	0.647
	R@10	0.392	0.391	0.390	0.396
	R@20	0.576	0.575	0.561	0.571
	R@30	0.689	0.688	0.668	0.677
	AUC-ROC	0.911	0.911	0.905	0.906
Readmission Pred. (FFN)		0.717	0.707	0.596	0.594
Mortality Pred. (GRU)		0.922	0.925	0.799	0.799

Table 5 demonstrates that all reproduction results for the metrics shown are within 2% of the original paper's results. According to AUC-ROC reproduction results, the diagnoses prediction FFN model, the readmission prediction FFN model, and mortality prediction GRU model using only CUI codes as input performed better than their corresponding CCS codes only as input counterparts by 0.6%, 17% and 15% respectively. Consequently, the reproduction results support the original paper's results. Moreover, since the models that used CUI codes as input performed better than the models that used CCS codes as input, the original paper's claim is in agreement with reproduction results that using only clinical notes (in the form of CUI codes) as input into the models performs better than only diagnoses code data (in the form of CCS codes) for the three tasks.

4.5 Result 5

The fifth and central claim that using a set of structured UMLS concepts (in the form of CUI codes) as input into the models performs better than word embeddings and raw sets of extracted concepts (which were used in previous works) as input for diagnoses and mortality prediction is supported by the reproduction results. Table 6 provides a summary of the original paper's results compared to reproduction results for the best performing models. Meanwhile, Table 7 is a comparison of Precision@1 results for diagnoses prediction among competing configurations presented in previous works.

Table 6: Summary of original and reproduction results for diagnoses, mortality, and readmission predictions

Task	Architecture	Input	Orig AUC-ROC	Repr AUC-ROC
Diagnoses Pred.	FFN	CUI Only	0.911	0.911
		CUI+CCS	0.913	0.916
	GRU	CUI+CCS	0.872	0.875
Mortality Pred.	FFN	CUI+CCS	0.871	0.862
	GRU	CUI+CCS	0.925	0.927
Readmission Pred.	FFN	CUI Only	0.717	0.707
		CUI+CCS	0.719	0.708
	GRU	CUI+CCS	0.534	0.627

Table 7: Comparison of Precision@1 results for diagnoses code prediction among competing configurations presented by Sushil et al.

Architecture	P@1
Original FFN (CUI+CCS)	0.778
Reproduction FFN (CUI+CCS)	0.778
Bag of Words (BoW)	0.701
Stacked Denoising Autoencoder BoW (SDAE-BoW)	0.650
doc2vec	0.681
(doc2vec,SDAE-BoW)	0.679
Bag of Words over CUI (BoCUI)	0.710
SDAE-BoCUI	0.665

Table 6 demonstrates that overall the results were successfully reproduced and align with the original paper's results. Using the results from Table 6 in conjunction with Grnarova et al.'s AUC-ROC average of 0.891 and Dubois et al.'s (a GloVe word

embedding configuration) AUC-ROC average of 0.90 for mortality prediction, the GRU model with CUI codes and CCS codes data as input presented in this work outperforms these competing configurations by 3.5% and 2.7%, respectively, with a AUC-ROC of 0.925 for the original results and 0.927 for the reproduction results [6-7].

As shown in Table 7, both the original and reproduction results for the diagnoses prediction FFN model outperform competing configurations in Precision@1 [8]. The closest competitor is BoCUI, but this paper's method still outperforms BoCUI at Precision@1 by 9%.

5 Discussion

The original paper was reproducible, but the computational requirements for some aspects provide a great barrier to entry. Some aspects of reproducing the results from the original paper were relatively straightforward, while other aspects were marred with technical difficulties. Other than the generation of Dataset B and the associated experiment for it, all experiments were successfully reproduced. Furthermore, the reproduced results largely agree with the original results within a few percentage points and as such support the claims of the original paper.

5.1 What was easy

Since the original authors provided ample detail in the original paper and the corresponding GitHub, following and implementing the steps needed to install QuickUMLS, process the MIMIC-III dataset, and run the experiments were relatively easy (in terms of next steps and commands) with only minor hiccups such as not all required dependencies being installed.

5.2 What was difficult

The main difficulty in reproducing results was the computational requirements and other technical difficulties and hiccups. Based on information in the original paper and the original paper's GitHub documentation, it was believed that the reproduction of the results was computationally feasible given the resources available for reproduction (i.e. local desktop computer), but this presumption was only partially correct. Neither the original paper nor the the original paper's GitHub documentation mention the necessary storage, memory, or specifications of the central processing unit (CPU) used. As a result of the lack of documented computational requirements, many processes (e.g. quickUMLS_getCUI.py) would start only for the process to be killed hours later because a bottleneck occurred such as RAM being saturated. This resulted in hours of training time and Google Cloud Platform credits being wasted. Another technical difficulty was that many hours were spent attempting to have Pytorch use the V100 that was a part of the Google Compute Engine virtual instance that was being rented. This problem was never successfully resolved, so my local desktop's RTX 3080 was used instead but it ran into CUDA out of memory errors for some of the RNN scripts and so workarounds and script modifications were needed.

5.3 Recommendations for reproducibility

Overall, the original authors did a fantastic job in providing proper guidance for reproducibility; however, there is still room for improvement in the area of computational requirements. The primary area needed for improvement is a more detailed description of the computational and time requirements needed to successfully process the data and run all experiments. Although some description of the hardware used was described in the original paper and time estimates for some of the data preprocessing steps, no detail was given on the RAM or VRAM (video RAM) requirements for the data preprocessing or the modeling stage. Since no RAM or VRAM requirements were given, time and money was wasted on virtual instances with not enough RAM (even 128GB of RAM was not enough to generate Dataset A) and workarounds were needed to reduce VRAM usage for the RNN's. Furthermore, no time estimates were given for the modeling stage. As a result, time and money was wasted attempting to reproduce datasets and results that were not feasible on the current hardware (whether local or through a virtual machine) being used. Therefore, it is recommended in the future to provide RAM requirements for the data preprocessing stage and VRAM (video RAM) requirements and time estimates for the modeling stage.

References

1. Zaghir, J., Rodrigues-Jr, J.F., Goeuriot, L. et al. Real-world Patient Trajectory Prediction from Clinical Notes Using Artificial Neural Networks and UMLS-Based

Extraction of Concepts. J Healthc Inform Res 5, 474–496 (2021). https://doi.org/10.1007/s41666-021-00100-z GitHub link: https://github.com/JamilProg/patient_trajectory_prediction

2. Unified medical language system (UMLS). https://www.nlm.nih.gov/research/umls/index.html, accessed April, 2022

3. QuickUMLS GitHub link. https://github.com/Georgetown-IR-Lab/QuickUMLS, Accessed April, 2022

4. Metathesaurus's unique identifiers. https://www.nlm.nih.gov/research/umls/newusers/onlinelearning/Meta005.html, accessed April, 2022

5. MIT's MIMIC-III. https://mimic.physionet.org/, accessed April, 2022

6. Grnarova P, Schmidt F, Hyland SL, Eickhoff C (2016) Neural document embeddings for intensive care patient mortality prediction. arXiv:1612.00467

7. Dubois S, Romano N, Kale DC, Shah N, Jung K (2017) Learning effective representations from clinical notes. Stat 1050:15

8. Sushil M, Šuster S., Luyckx K, Daelemans W (2018) Patient representation learning and interpretable evaluation using clinical notes. J. Biomed. Informatics 84:103