Reproducibility Project Report for CS598 DL4H in Spring 2022

Dinesh Agarwal and Jessica Nwaogbe
{dinesha3, nwaogbe2}@illinois.edu

Group ID: 80
Paper ID: 51, Difficulty: Hard
Presentation link: https://www.youtube.com/watch?v=wGusC7OSzrY
Code link: https://github.com/da230896/G-BERT

1 Introduction

This project is an exploration of the research conducted and published in the paper, "Pre-training of Graph Augmented Transformers for Medication Recommendation" (Shang et al., 2019b). The research paper proposed a new model called G-BERT (Graph-augmented BERT (Devlin et al., 2019), i.e. GNN and BERT) to address the challenges faced in deep learning in providing effective medication recommendations from patient EHR data.

G-BERT handles two main limitations of prior deep learning studies in medication prediction: selection bias and lack of hierarchical knowledge. Other existing works tend to discard large amounts of EHR data (e.g. patients with only one hospital visit) to meet standard criteria (e.g. patients with more than one hospital visit), leading to selection bias. They also tend to fail in retaining the hierarchical nature of medical ontology in their model training; thus, missing valuable hierarchical knowledge.

To address these challenges, G-BERT combines pre-training techniques to utilize each visit of EHR data and graph neural networks to produce more accurate medical code representations.

According to the paper, the pre-trained BERT along with the graph neural network, G-BERT, has higher accuracy in medication recommendation than all other baselines: Logistic Regression, GRAMs (Choi et al., 2016b), LEAP (Zhang et al., 2017), RETAIN (Choi et al., 2016a), and GAMENet (Shang et al., 2018), as evidenced by higher Jaccard, PR-AUC, and F1 scores.

2 Scope of Reproducibility

Based on the complexity of the task we have scoped our attempt to reproduce the original paper. We have implemented G-BERT model and two baselines which are Logistic Regression and GAMENet on MIMIC-III data-set. MIMIC-III (A. et al., 2016) is the data-set that has been used in G-BERT paper. Finally, we expanded on the study with the following ablations: 1. Removal of the pre-training-based, transfer-learning approach on single-visit patients, 2. Removal of graph ontology embeddings

2.1 Addressed claims from the original paper

We tested following claims from the original paper:

- Claim 1: High performance of G-BERT on MIMIC-III data-set
- Claim 2: Comparatively better performance of G-BERT than baselines
- Claim 3: Benefits from structural tasks like ontology embedding, along with benefits of a transfer-learning-based pre-training task

3 Methodology

We have adopted the approach of implementing the data pipeline, model, and several baselines on our own since the author's code (https://github.com/jshang123/G-Bert) lacked adequate documentation to understand the behaviour of the code. Instead, we relied on the details from the research paper and

made a few of our own assumptions. In our implementation, we have used various resources like pytorch, PyTorch-Geometric, pandas and other tools like Visual Studio Code to do the same.

3.1 Model descriptions

The model used in the original paper comprises of an initial stage where graph neural networks for both diagnoses codes and medications codes are employed. The nodes in the graphs are the codes from two standard medical ontologies: ICD-9 for diagnoses and ATC4 for medication. In particular, the number of nodes in each graph is equal to the number of unique words in the corresponding medical ontology plus the "general" category terms derived from the medical ontology hierarchy. Convolution is done on the graph neural networks (one for ICD-9 codes and one for ATC4 codes) to produce embeddings in a staged fashion (more details in the paper). These embeddings are fed into a BERT (Devlin et al., 2019), a transformer encoder model. The output at the "[CLS]" token from BERT is used as the visit embedding. BERT with GNN is to be referred to as G-BERT.

This visit embedding is used in the pre-training of G-BERT for double prediction and self-prediction objectives. To note, in fine-tuning (final training), the model has only the objective of predicting medication.

3.2 Data descriptions

The original paper uses medical data obtained from MIMIC-III, a publicly-available clinical database of EHR data from patients admitted into the critical care units of a hospital. In particular, the paper's code base uses the DIAGNOSIS and PRESCRIPTIONS tables from this data-set. We were able to obtain the original data-sets through PhysioNet credentialed access. For our experiments, we also obtained the ADMISSIONS table to help with data processing.

The ADMISSIONS table provides general information about each patient's admission (or visit) to the hospital. Each admission is designated with a unique ID (HADM_ID). Information about the admission time, discharge time, as well as demographic data is included in this table. The DIAGNOSIS table contains the ICD-9 codes assigned to each patient at each hospital visit. The PRESCRIPTIONS table contains the medication orders prescribed to each patient at each hospital visit. Information about NDC codes, drug names, dosages, and other medication-specific information is included in this table. In the original paper and in our experiments, NDC codes are mapped to ATC4 codes to derive the hierarchical structures for the ontology embeddings (for more details refer to our repository, specifically Part-1 here).

The data-sets were originally formatted as CSV files, which we processed as .pkl files (available here): one for patients with single hospital visit and one for patients with multiple visits.

Stat	Single-Visit	Multi-Visit
# of patients	29189	5917
avg # of visits	1.0	2.68
avg # of unique dx per visit per patient	11.44	14.22
avg # of unique rx per visit per patient	21.21	22.56
unique # of dx	6191	4551
unique # of rx	398	385

Table 1: These statistics were obtained from the .pkl files for single/multiple visits.

Most of the statistics in Table 1 are close to the values generated from the original CSV files from MIMIC-III (e.g. # of multi-visit patients, average # of visits, # of unique diagnoses and medications). The closeness gives us confidence that we are processing data in the right fashion. This is important since we implemented the whole pipeline with our own understanding. However, compared to the original paper's statistics, several of our statistics were off (e.g. average # of unique rx per visit per patient, unique # dx/rx). We hypothesized that the average number of dx/rx is off because the authors considered non-unique, repeating codes as distinct within a visit. Moreover, the # of unique dx/rx we obtained is more than that of the original paper since they truncated visits

to just 24 hrs. We have kept things more on as is basis; hence, the larger number of unique # dx/rx.

One can compare the statistics from the original paper (Shang et al., 2019b), where they mention the stats in Table 2. For details about stats from the original MIMIC-III CSVs, one can head to this link and refer to section Stat-3.

3.3 Hyper-parameters

Below are details about various hyper-parameters in the our implementation of the G-BERT:

- For Graph (Ontology Embedding): We used GATConv from PyTorch-Geometric with dropout of 0.1. Internal representation is of size 100 and number of heads are 5 for convolution. Output channel size is also 100 (20*number of heads = 20*5 = 100)
- For BERT: BERT is implemented using Transformer Encoder. It has 2 layers of Transformer encoder with 4 heads at each layer. Internal representation size is 300. Output representation size is also 300
- Prediction layer for pre-training: 4 linear layers since there are two pre-training objectives for each ICD-9 and ATC4-based visit embedding. Input representation is of same size as G-BERT output size whereas output size is based on the vocab size. Batch size for pre-training is 100.
- Prediction layer for training: 1 linear layer with input representation of same size as G-BERT output size and output size is based on the vocab size of medication (i.e. unique medication to predict). 0.5 is the probability threshold to consider medication as recommended. Since the number of visits can be different for each patient, it was difficult to batch patients.
- Other important hyper-parameters: Learning rate for Adam: 0.001, but some experimentation was done using 5e-4 as well. Train-evaluation-test set ratio is 0.6:0.2:0.2. Also, probability threshold is 0.5 to convert output probability into binary values (1/0).

3.4 Implementation

Our repo (Agarwal and Nwaogbe, 2022) is here. As mentioned before, we only took reference from the author's code (Shang et al., 2019a) present in their GitHub repo and wrote our own code after developing basic concepts from them. Moreover, there were few assumptions in the author's implementation that were not mentioned in the paper. Coding on our own allowed us to implement our own interpretation of the main paper's ideas. For example, according to the paper, single-visit patients are those who have a single hospital admission, which has been used as-is in our implementation; however, the author's implementation uses a 24-hour cutoff.

Data Processing: To process the MIMIC-III data-set, we needed NDC-to-RX-Norm mappings and then, RX-Norm-to-ATC4 mappings. Though the mapping files were not checked-in to the author's repo, they use standard mapping files found here and here. These files are in GAMENet's official implementation repo, which happens to be one of the baselines for G-BERT as well. Once data was in the format we needed, vocab was created for the ICD-9 and ATC4 codes in the files: "unique-icd.csv" and "unique-atc4.csv" respectively. Please note, only the top 2000 most frequent ICD-9 codes were considered for the ICD-9 vocab. This is in line with the author's implementation. After pre-processing the data, we stored the results in "single_visit.pkl", "multi_visit.pkl", and "multi_visit_temporal.pkl". Details about how each file is created are in the jupyter notebook, main.ipynb.

GNN and BERT: In our implementation, we used the message passing graph attention network from PyTorch-Geometric to do GCN on the medication and diagnosis ontology. The ICD-9 ontology tree has 2640 nodes (leaf nodes + internal nodes + special tokens). Similarly, the ATC4 ontology tree has 681 nodes (leaf nodes + internal nodes + special tokens). The leaf node count in the ATC4 ontology tree is 413 (can also be thought of as unique ATC4 codes). This seems to be a little off from Table 1 but within tolerable margin. In BERT, we used the "[CLS]" token to generate the visit

embedding. This visit embedding is used to predict ATC4 codes for that visit in the training phase and the pre-training phase for self-prediction and double-prediction. More details can be viewed here in the jupyter notebook, main.ipynb.

Feed Forward Networks(FFN): We have an FFN comprising of linear layers on top of the G-BERT output. In the pre-training phase, we take the BERT output for double prediction and self prediction. Pre-training FFN has ReLU activation function.

Similarly, in the training phase, we use an FFN comprising of a single Liner Layer and a Sigmoid function to generate the output probability. Only those medications that have a probability higher than a certain threshold are to be considered as recommended. Input to the training-phase FFN is the concatenated vector consisting of the averaged T-1 visits' embedding vector (one vector each for medication and diagnosis codes) and the T^{th}-visit visit embedding vector (using only diagnosis code for T^{th} visit). Thus, the input size is 3 times the output size of G-BERT.

Training Procedure: As authors have mentioned in the paper, training is done in alternation with pre-training with 5 epochs each and 15 times overall.

Baselines: We have implemented two baseline models, Logistic Regression and GAMENet (Shang et al., 2018). For Logistic Regression, we used grid search over a typical range of hyper-parameters to search for the best hyper-parameter value that results in an L2 norm penalty. For our GAMENet implementation, we took reference from the course's homework module on GAMENet. In the original, both patient EHR data and drug-drug interaction (DDI) knowledge were leveraged for the medication recommendations. So for fair comparison to the G-BERT model, we removed the DDI component and only considered patient EHR data. The same data-sets used in the G-BERT model were also used in the Logistic Regression and GAMENet implementation.

Number of Parameters: The number of parameters as per the paper is 3 million; however, in our implementation, we were able to produce a model with only $\sim$ 2.5 million parameters. Also in our initial implementation, the number of parameters were close to 7 million. This can be explained by the fact that earlier we did not filter out the less frequent ICD-9 codes. As a result, by taking all the unique ICD-9 codes as input ($\sim$ 6K) to the vocab, without discarding less frequent ones, we had almost 3 times the vocab size. This translates to almost 2.5 times the number of parameters as per the original model due to embedding parameters.

Assumptions: The authors did not mention this in the paper but it is understood from the author's implementation (Shang et al., 2019a) that they rejected less frequent codes for more frequent ones. Thus, when we considered only the most frequent 2000 ICD-9 codes, we had to mark this as an assumption we made. Other assumptions made during implementation are listed out in the README.md of our repo. One can visit our repo to read in detail about them.

3.5 Computational requirements

Initially, we wanted to use the HAL system for our experiments to ensure that we had adequate computational resources for training. We tried accessing the HAL system from NCSA, which has V100 GPUs. However, HAL does not come pre-installed with PyTorch-Geometric. Since creating the custom environment was taking time, we shifted our training to locally-available resources. We analysed and concluded that our locally-available resources (personal computers without GPU) were a sufficient substitute for the HAL system (due to the low computational requirements of the paper) with expected training times of $\sim$ 1.25 day.

Our most recent implementation took 15 hrs of CPU time for pre-training with peak memory utilisation of $\sim$ 16MB using a local device with a configuration of 16 GB RAM and 2.3 GHz, and an 8 core Intel i9 processor. Moreover, fine-tuning time was also close to 15 hrs on the same machine. Some other details regarding computational requirements are as follows:

- Total number of epochs: For pre-training, we did $15 * 5 = 75$ epochs and

the same for training. For details, refer to "Training procedure" in Section 3.4

- Run-time per epoch: For pre-training and training, it took about 12 minutes. Though pre-training had more data, it was batched so it was fast.

Our estimates were close to the actual pre-training, training, and evaluation time. One reason why training/pre-training seems to be manageable on CPU itself is that the data-set that we are using only has a few thousands records.

4 Results

4.1 Result 1

Claim 1 is verified since our model has high Jaccard/PR-AUC/F1 scores on MIMIC-III data-set; which are even higher than the scores presented in paper. We surprisingly had an 86% higher Jaccard score (refer to Table 2 and Table 3).

Methods	Jaccard	PR-AUC
G-BERT(Author)	0.46	0.70
G-BERT(Ours)	0.86	0.98

Table 2: Performance (Jaccard/PR-AUC) on Medication recommendation task

Methods	F1	# of Parameters
G-BERT(Author)	0.62	3 Million
G-BERT(Ours)	0.92	2.47 Million

Table 3: Performance (F1/# of Parameters) on Medication recommendation task

4.2 Result 2

We were able to reproduce two baseline models, Logistic Regression and GAMENet, which verified Claim 2 of the paper. Our implementation of G-BERT was able to perform better than both of the baselines (refer to Table 4 and Table 5 for baseline performance and Table 2 and Table 3 for G-BERT performance).

4.3 Result 3

Claim 3 was partially verified since one of the ablations yielded results opposite to our expectations.

Methods	Jaccard	PR-AUC
GAMENet^{-}(Authors)	0.44	0.67
GAMENet^{-}(Ours)	0.26	0.33
Logistic Regression (Ours)	0.24	0.52

Table 4: Performance (Jaccard/PR-AUC) on Medication recommendation task, GAMNet^{-}=W/O DDI

Methods	F1	# of Parameters
GAMENet^{-}(Authors)	0.60	5.5 Million
GAMENet^{-}(Ours)	0.42	961K
Logistic Regression (Ours)	0.35	-

Table 5: Performance (F1/ # of Parameters) on Medication recommendation task, GAMNet^{-}=W/O DDI

- Ablation 1: We removed pre-training (P^{-}) from the end-to-end training procedure in this ablation. We saw a drastic fall in the Jaccard score. This confirmed that pre-training helped improve the downstream task accuracy. For actual scores, refer to Table 6 and Table 7.

- Ablation 2: We replaced the Graph Ontology Embedding (G^{-}), which was implemented using the GATConv module from PyTorch Geometric Library, with a torch.nn Embedding module. The output dimensions were kept the same. We saw a similar performance and in fact, a little better performance than our G-BERT model without any ablation. For actual scores, refer to Table 6 and Table 7.

Methods	Jaccard	PR-AUC
G-BERT(Ours, P^{-})	0.46	0.70
G-BERT(Ours, G^{-})	0.92	0.98
G-BERT(Ours)	0.86	0.98

Table 6: Performance (Jaccard/PR-AUC) on Medication recommendation task, P^{-}=W/O Pre-training, G^{-}=W/O Ontology Embedding

Methods	F1	# of Parameters
G-BERT(Ours, P^{-})	0.62	2.47 Million
G-BERT(Ours, G^{-})	0.96	2.45 Million
G-BERT(Ours)	0.92	2.47 Million

Table 7: Performance (F1/# of Parameters) on Medication recommendation task, P^{-}=W/O Pretraining, G^{-}=W/O Ontology Embedding

5 Discussion

Result 1: Results from our implementation of the model are better than that of the paper. This can be reasoned by the fact that we have more data per prediction. That is, the author's implementation seems to only keep the first 24-hrs of each admission, whereas we have kept the whole visit data. Possibly, due to this extra information, we have better performance. Apart from this, we witnessed a big drop in performance when we did not take the 2000 most frequent ICD-9 codes. A plausible explanation for this would be that the number of unique ICD-9 codes came out to be $\sim$6K which lead to a huge number of parameters($\sim$ 8 Million) for the embedding layer and learning a high number of parameters require data which we only have in few thousands.

Result 2: Logistic regression (LR) was performed using Grid Search with typical parameters with L2 penalty. Results are in line with the hypothesis and LR results form a good baseline for the whole experimentation. GAMENet performed better than LR and overall, G-BERT performed better than the baselines, thus verifying Claim 2.

Result 3: The ablation study clearly confirms the main idea of the paper, i.e pretraining improves the downstream task accuracy. However, similar or better performance was observed after removal of the ontology embedding. Our hypothesis is that due to reduced number of parameters, a reduction of almost 20K parameters (Refer Table 7), the model was able to generalize better. On the other hand, the t-SNE plot tells that better clustering is observed with the Ontology Embedding than without; thus, the Ontology Embedding makes prescriptions more interpretable. Refer to Figure 1 and Figure 2.

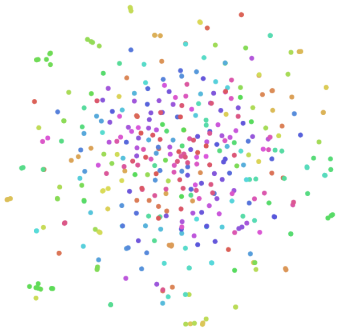

Figure 1: t-SNE for Graph Ontology Embeddings, clusters appear at the boundary

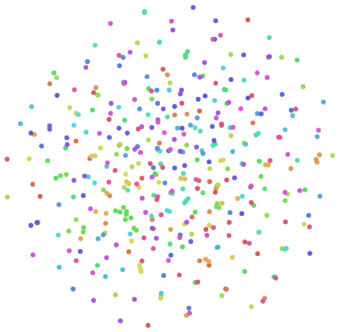

Figure 2: t-SNE for Embeddings without Graph Ontology, no clustering seen

5.1 What was easy

The G-BERT paper was considered a hard paper as per the sign-up sheet. Once we went through the paper and divided the tasks, we gradually got a hang of the paper and were able to implement the model ourselves. We studied the MIMIC-III data-set in-depth and were also able to develop the data processing pipeline on our own.

However, this being our first such project, it was difficult to pinpoint something that we felt was easy. Though, we can say that since the data was smaller in number compared to what other research papers of this nature usually process, training was fairly simple as we were able to train the model on local resources.

5.2 What was difficult

We were optimistic that having access to the author's code would simplify our tasks but

the code seemed to lack structure and documentation about how to run it. Hence, we were only able to use the code for referential purposes. This made the project implementation fairly complex. Processing the data-set was also complex since we needed to distinguish between single-visit and multi-visit patients. Folks unfamiliar with pandas might have a difficult time coming up with such processing methods. Also, implementing the Graph Attention Module required knowledge of PyTorch-Geometric, which we did not have a strong-hold on. Furthermore, the author's implementation (Shang et al., 2019a) made some assumptions that were not mentioned in the paper, which forced us to make our own judgement calls about which paths to follow.

5.3 Recommendations for reproducibility

We would highly suggest the authors to have more documentation and better structured code. Moreover, there were some data preprocessing steps in the code that were not mentioned in the paper, like filtering visits to the first 24 hours. It would be highly relevant to mention such crucial steps in the paper as well.

6 Communication with original authors

We began our communication with the original authors during the early phases of implementation, when we were implementing the data pipeline. However, since we did not receive any response from them, we could not extend our communication with them.

References

Johnson A., Pollard T., and Mark R. 2016. Mimic-iii clinical database (version 1.4). *PhysioNet*.

Dinesh Agarwal and Jessica Nwaogbe. 2022. G-bert implementation for cs-598.

Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F. Stewart, and Jimeng Sun. 2016a. RETAIN: interpretable predictive model in healthcare using reverse time attention mechanism. *CoRR*, abs/1608.05745.

Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F. Stewart, and Jimeng Sun. 2016b. GRAM: graph-based attention model for healthcare representation learning. *CoRR*, abs/1611.07012.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Junyuan Shang, Tengfei Ma, Cao Xiao, and Jimeng Sun. 2019a. G-bert implementation.

Junyuan Shang, Tengfei Ma, Cao Xiao, and Jimeng Sun. 2019b. Pre-training of graph augmented transformers for medication recommendation. *CoRR*, abs/1906.00346.

Junyuan Shang, Cao Xiao, Tengfei Ma, Hongyan Li, and Jimeng Sun. 2018. Gamenet: Graph augmented memory networks for recommending medication combination.

Yutao Zhang, Robert Chen, Jie Tang, Walter F. Stewart, and Jimeng Sun. 2017. Leap: Learning to prescribe effective and safe treatment combinations for multimorbidity. In *KDD 2017 - Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1315–1324. Association for Computing Machinery. Funding Information: Acknowledgement: This work was supported by the National Science Foundation, award IIS-#1418511 and CCF-#1533768, Children's Healthcare of Atlanta, Google Faculty Award and UCB. Dr. Stewart was supported by NIH RO1 HL116832. Yutao Zhang and Jie Tang are supported by National Natural Science Foundation of China (61561130160), National Social Science Foundation of China (13&ZD190), and the Royal Society-Newton Advanced Fellowship Award. Publisher Copyright: © 2017 ACM.; 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2017 ; Conference date: 13-08-2017 Through 17-08-2017.