Reproducibility Project: Pre-training of Graph Augmented Transformers for Medication Recommendation

Prateek Dhiman and Unnati Hasija
{pdhiman2, uhasija2}@illinois.edu

Group ID: 15
Paper ID: 17
Original Paper: [Shang et al., 2019a] https://www.ijcai.org/proceedings/2019/825
Presentation link : https://www.youtube.com/watch?v=H8Zu0Iho0nw
Code link: https://github.com/unnatihasija1/DLHTeam15.git

1 Introduction

As part of CS 598 Deep Learning in Healthcare Spring 2023 Semester, we have chosen as our research paper **"Pre-training of Graph Augmented Transformers for Medication Recommendation"** [Shang et al., 2019a].

This research paper discusses the importance of medication recommendation in healthcare and the limitation of existing models like GAMENet [Shang et al., 2019b] in utilizing electronic health records (EHR) . This research paper addresses two aspects lacking in GAMENet i.e. missing patients with single visit and diagnosis hierarchy .This research paper highlights these two important gaps listed below.

1. **Selection bias** - Existing models would ignore patients with single visits as part of selection criterion. This tends to form a signification part of EHR data and caused a selection bias.

2. **Missing hierarchical knowledge** - These models did not leverage the diagnosis hierarchy in the representational learning process.

This research proposed to overcome these gaps by building a new model called G-BERT. This new models is engineered from two core deep learning networks, namely Graph Neural Networks(GNN) and Bidirectional Encoder Representation from Transformers (BERT) [Devlin et al., 2019]. GNN provides efficient means for medical ontology embedding and representation while BERT excels at natural language tasks like medication recommendation in this case. G-BERT was pre-trained on the visit embedding. Finally a prediction layer was added to fine tune the medical recommendation model.

G-BERT was the first such model to bring the a pre-trained large language model schema into healthcare domain while having better benchmarks than existing models.

Achievements
- **Improved** baseline GBERT **KPIs** F1 by 2.1%, PR AUC by 1.7% , Jaccard by 3.3% in our experimental setup via ablations.
- **Reproduced** the **accuracy** and other metrics within 99,46% of PR AUC, 99.13% of F1 Score, and 99.13% of Jaccard score values of original model.
- Built a **prototype called GGPT** which proposes initial steps towards replacing BERT architecture with GPT2 architecture.

2 Scope of reproducibility

At the time of publishing in 2019 [Shang et al., 2019a] connected two cutting edge technologies to achieve more accurate medication recommendation. The main claim of this research paper is stated in **Section 2.1** below.

Our approach is discussed in detail in **Section 3 Methodology** and associated **results in Section 4**. The scope for this project is limited to the following items and has been fully completed.

1. **Method Replication and retraining** of G-BERT with as near accuracy and KPIs as possible on local machines and Azure VMs.

2. **Compare baseline** performance by verifying and running other models mentioned in the paper like [Zhang et al., 2017], RETAIN [Choi et al., 2017a], and GAMENet [Shang et al., 2019b] for medication recommendation.

3. **Ablation on Transformer Architecture**

4. **Ablation on feasibility with GPT - GGPT** A Prototype made with a potential of improvement.

2.1 Addressed claims from the original paper

We addressed the following claims:

- The proposed pre-training approach using graph augmented transformers (Model G-Bert) has higher prediction accuracy than another state of art models like RETAIN [Choi et al., 2017b], and GAMENet [Shang et al., 2019b] for medication recommendation
- The proposed G-Bert model performs better with graphs and pre-training on EHRDataset.

3 Methodology

To verify that the proposed G-BERT model with hierarchy medical ontology and pre-training works as claimed, we started with replicating G-BERT in local machines (laptops) and on Microsoft Azure for further bench marking.

In order to fully address the claim, it was imperative that we also validate other models in a similar environment. Our reproducibility is divided into following main parts.

1. **Model replication and retraining** : We used the author's code with slight modification and updates. We were able to successfully replicate this step. See section 3.4 Implementation for more details.
2. **Baseline comparison** with other models mentioned in paper : We used standard code provided by authors of other models and with slight modification and adaptations. See section 3.4 Implementation for more details.
3. **Ablations** with Transformer Architecture
   (a) **Impact with changing Layers and Attention mechanism**
      i. **GATConv to GCNConv**: The intention to try this ablation was to prove the effect of the attention mechanism and the impact of using changing weights compared with fixed weights. GAT: uses a self-attention mechanism to learn weights for aggregating information from neighboring nodes while GCN(Graph Convolutional Network) uses a convolutional operation to aggregate information from a node's immediate neighbors. Accuracy metrics with GCNConv were less than that with GAT.
      ii. **GATConv to GTNConv** : This ablation was attempted because GTN(Graph Transformer Network) is known to be more scalable as it is also based on Transformer architecture and is flexible and can be adapted to different types of graph data. Since the results of GCNConv proved to be worse than GAT, the intention was to try GTN, since it still uses an attention mechanism. From an execution perspective, it was faster. Every epoch took 4 minutes as opposed to 7 minutes for GATConv on the same GPU instance. We also trained the G-Bert model after increasing the attention heads from 4 to 6. The code for both GCN and GAT can be found in graphmodels.py
   (b) **Changing aggregation function** : aggr_out from sum to mean for the MessagePassing class. As we know, the aggr parameter controls how the messages from neighboring nodes are aggregated before being passed to the update function. It determines whether the messages are added or averaged across the neighboring nodes. Changing the 'aggr' from add to mean can reduce sensitivity to outliers and is computationally efficient. It did not improve the prediction accuracy.
   (c) **Increasing Attention heads and Dropout probability**: This was done as part of the hyperparameter tuning. See section 3.3 for more details.
4. **Ablation** on feasibility with GPT2-based model: As an extension to G-Bert and for our feasibility study, we also propose a new model: GGPT which combines GNN with GPT2 model. We believe, using a Graph Neural Network (GNN) with GPT-2 for medical recommendations can also be a promising approach. GPT-2 has a large model size and a high number of parameters (1.5 billion) which may allow it to capture more complex relationships between medical concepts and generate accurate recommendations. GPT-2 includes an autoregressive language modeling component. and can be advantageous for generating medical recommendations. To implement this,

we follow the same approach as the one followed for G-Bert.

3.1 Model descriptions

The original models used in this paper are

1. **Graph Neural Network (GNN)** : GNN is a neural network architecture designed for graph-structured data analysis. It consists of multiple layers that process nodes and edges to update their representations based on neighboring nodes and edges. The main objective of GNN is to predict node or edge labels, and the number of parameters in a GNN depends on the specific architecture and the size of the input graph.

2. **BERT** (Bidirectional Encoder Representations from Transformers) is a pre-trained natural language processing model based on the Transformer architecture, which uses self-attention mechanisms to process sequential data. Its architecture consists of a multi-layer bidirectional Transformer encoder, allowing it to consider both the left and right contexts of each token during training. The number of parameters in BERT varies depending on the size of the model, ranging from 110 million parameters in BERT-Base to 340 million parameters in BERT-Large

3. **GPT2** (Generative Pre-trained Transformer 2) is based on the Transformer model and was trained on a massive amount of text data to predict the next word in a sequence. Like BERT, GPT2 also has contextual understanding, large-scale language modeling and it's also open source. GPT2 can also be fine-tuned on specific tasks. GPT-2 has 1.5 billion parameters.

Further, the paper refers to state of art models like Logistic Regressions(LR) , LEAP [Zhang et al., 2017], RETAIN [Choi et al., 2017b], GRAM [Choi et al., 2017a], and GAMENet [Shang et al., 2019b]. Details for these are available in the references section.

3.2 Data descriptions

MIMIC-III [Johnson et al., 2016] dataset has over 40000 patient information and over 2 million clinical notes. After completing CITI Data or Specimens only Research training and getting access to Physionet, we got access to MIMIC-III dataset. Additionally, the authors of the paper have provided a notebook (EDA.ipynb) which uses necessary EHR-relevant files from MIMIC-III dataset and presents them in a convenient easy-to-use pickle format for the models. These files are PRESCRIPTIONS.csv, DIAGNOSES_ICD.csv, PROCEDURES.csv

For this project, we were able to use the .pkl files directly. Here are the data statistics (dx for diagnosis, rx for medications) [Shang et al., 2019a]

Stats	Single-Visit	Multi-Visit
# of patients	30745	6350
avg # of visits	1.00	2.36
avg # of dx	39	10.51
avg # of rx	52	8.80
# of unique dx	1997	1958
# of unique rx	323	145

3.3 Hyperparameters

Hyperparameters can have a significant impact on the performance of a neural network. The model when trained using run_pretraining.py takes the hyperparameters from config.py file and builds the model based on the same. For our ablation study, we changed the following hyperparameters:

Attention heads: In general, increasing the number of attention heads can be beneficial for improving the performance of the model, but it is important to carefully balance the trade-offs between model capacity, interpretability, stability, computation time, and risk of overfitting. In our case, changing the attention heads with the original model, did not improve the performance.

We also trained the G-Bert model with GTN after increasing the attention heads from 4 to 6, the performance proved to be better. See the results section for more details.

Dropout probability: As we know, increasing the dropout probability can result in a more regularized model that is less prone to overfitting, but may also result in a decrease in performance on the training data. A higher dropout probability can slow down the convergence of the model during training, as the network may need more iterations to learn the correct weights. Following the principle of dropout annealing, we attempted to change hidden-dropout-prob and attention-probs-dropout-prob from 0.5 to 0.1 with a decay of 0.05. Changing the dropout probability did not improve the performance of the model.

3.4 Implementation

We followed a hybrid approach for code reusability, we used the existing code for our hypotheses and for proving claims after adapting it for our environment. For ablations with G-Bert, we did some modifications to the existing code: we wrote code for GCNConv and GTNConv and replaced them with GATConv for our ablations. The code for the same can be found in graphs_model.py.

We also attempted to develop a new model as an extension to the original paper: Graphs(GNN) with GPT2 model: GGPT. We know, BERT is trained using a masked language modeling (MLM) task, predicting masked words in a sentence, while GPT-2 is trained using a causal language modeling (CLM) task, predicting the next word in a sequence given the previous words. The code for the same is provided in GGPT.ipynb.

Key implementation details for G-BERT and other models are.

G-BERT :

- Git clone the original repository
- Updated/installed missing libraries from Readme.md created a new requirement.txt
- Updated Torch.load function to enable the code to run on local laptop despite of CPU parameters.
- Update jaccard_similarity_score to jaccard_score as per the newer version of sklearn.metrics
- update scatter_ method for scatter.py file method in torch_geometric version 1.0.3 library as shown below to fix run time error with run_gbert.py.
- Initial test run via *python run_gbert.py –model_name GBert-predict –use_pretrain –pretrain_dir ../saved/GBert-predict –graph*
- Test run with re training without graphs via *python run_gbert.py –model_name GBert-predict –do-train*
- Test run with graphs via *python run_gbert.py –model_name GBert-predict –do-train –graph*

Ablations to G-Bert: GTNConv: GTN implements an 'add' aggregation to compute node embeddings and applies a leaky ReLU nonlinearity. Like GAT, it also uses a learnable attention mechanism to weight the importance of different nodes in the neighborhood aggregation. However, there's no normalization using attention coefficients using the Softmax function or dropout regularization generalization.

GCNConv: class implements the Graph Convolutional Network (GCN) operator. This operator applies a simple linear transformation followed by a non-linearity to the input graph to propagate messages from neighboring nodes. It supports a single head and concatenation of the outputs from different GCN layers.

GAMENet : The paper also claims that it works better than GAMENet, which basically is the method to recommend accuracy and safe medication based on memory neural networks and graph convolution network by leveraging EHR data and Drug-Drug Interaction (DDI) data source.

- *python train_GAMENet.py –model_name GAMENet –ddi (training with DDI knowledge)*
- *python train_GAMENet.py –model_name GAMENet (training without DDI knowledge)*

We also needed to re-execute the EDA.ipynb with slight modifications to regenerate the .pkl files for GAMENet model to execute.

RETAIN: RETAIN has been shown to be effective in predicting clinical outcomes such as mortality, readmission, and disease onset. To prove the claim from the original paper, we reused the code for Retain from the GAMENet repository after changing the EDA.ipynb and regenerating the .pkl files. This was needed because the .pkl files were generated using Python 3.7 and was used in Python 3.8

GGPT : Based on recent developments like BioGPT based on GPT2 architecture, we attempted to build a protptype to replace all references of BERT with GPT2 Architecture. and train new model on EHRDataset for a single visit. We utilized the existing code as the base and made neccessary modifications in Embedding classes to utilize GPT2 configuration instead of BERT specific configurations. The embedding were utilized in GGPT2 Pretrained model which was initially trained on single visits. The code for this model is available in our Git Repo as a jupyter

notebook

We obtained the score for the above models as mentioned in Section 4 Results.

3.5 Computational requirements

The research paper [Shang et al., 2019a] used Pytorch [Paszke et al., 2019] and trained / fine tuned the model on an Ubuntu 16.04 with 8GB memory with Nvidia 1080 GPU. Our initial approach was to acquire similar hardware from Google Colab or Azure.

However as we went ahead with the project, we initially setup the research paper on our local laptops. Initial retraining with 20 epochs on our laptop with following configuration took around 10 hours to run.

Local Configuration. Processor 11th Gen Intel(R) Core(TM) i7-11370H @ 3.30GHz 3.30 GHz Installed RAM 32.0 GB System type Windows 11 64-bit operating system, x64-based processor

For efficient benchmarking while being cognizant of the cost, we switched to Microsoft Azure dedicated Virtual machine. The overall retraining time with 20 epochs was reduced to 3 hours. Given the time we have and the progress we have made with ablations, we settled on the following computational specifications.
Azure VM Compute specifications (Reference Microsoft Azure VMs)

VM	Core	Memory (GB)	Storage (GB)
Standard _DS3_v2 CPU	4	14	28
NVIDIA Tesla M60 GPU	6	56	380

4 Results

We conducted experiments to prove the claims in the paper and after training and executing the experiments on pre-trained models, we were able to reproduce that G-Bert with pre-training and graphs performs the best as claimed in the paper when compared with G-Bert without graphs or without pre-training or without both graphs and pre-training. We were also able to reproduce that G-Bert performs better than the other attention-based model such as RETAIN and another state-of-art model GAMENet (with or without DDI knowledge).
Results from our experiments can be seen in the Results 1 table. We were able to reproduce the accuracy and other metrics within 99.46% of PR AUC, 99.13% of F1 Score, and 99.13% of Jaccard score values. The accuracy results for GAMENet are a little far from the ones mentioned in the original paper because we generated new .pkl files.

After experimenting with our ablations as described in Section 3: we see that our ablation of changing GAT with GTN, together with increasing the attention heads from 4 to 6 proved to show better prediction accuracy by 0.52% or F1 score, 0.39% PR AUC and 0.56% of Jaccard Score of the original G-Bert metrics. The percentages are calculated based on the accuracy metrics that we were able to replicate on G-Bert. The results for our ablations are in the Results 2 table.

Method Variants	Description
G-BERT$_{G-,P-}$	without training and graphs
G-BERT$_{G-}$	without graphs
G-BERT$_{P-}$	without pre training
GAMENet$_{DDI}$	Gamenet with DDI
GAMENet	Gamenet without DDI

4.1 Result 1

Achieved performance and baseline comparison

Method	F1	PR AUC	Jaccard
GAME Net$_{DDI-}$	0.3596	0.4326	0.2399
GAME Net	0.3497	0.4561	0.2320
RETAIN	0.3127	0.3573	0.2013
G-BERT $_{G-}$	0.6038	0.6824	0.4452
G-BERT $_{G-,P-}$	0.5528	0.6366	0.3928
G-BERT $_{P-}$	0.5505	0.6385	0.3888
G-BERT	0.6065	0.6906	0.4478

4.2 Result 2

Ablation with Transformer Architecture

Method	F1	PR AUC	Jaccard
aggr_out : to mean	0.5710	0.6554	0.4096
aggr_out : to max	0.5550	0.6318	0.3942
Attn head 4 to 8	0.5245	0.6201	0.3644
GAT to GCN	0.5429	0.6275	0.3823
GAT to GTN	0.6063	0.6896	0.4463
GTN Attn heads=6	**0.6121**	**0.6967**	0.4524

4.3 Additional results not present in the original paper: Results 3

Apart from the ablations mentioned above, we also changed the activation functions in graph_models.py from leakyRelu to Mish. We tried this because compared to LeakyReLU, Mish has a more gradual transition from negative to positive values, which can help with the vanishing gradient problem. Though, we understand both Softmax and LeakyRelu are better than Sigmoid, we tried this ablation to understand the problem of vanishing gradient in this scenario.

Method	F1	PR AUC	Jaccard
LeakyRelu to Sigmoid	0.5125	0.5464	0.3533
Softmax to Sigmoid	0.4808	0.5810	0.3254
LeakyRelu to Mish	0.5904	0.6722	0.4298

5 Discussion

We also tried changing the activation functions LeakyReLU to ReLU, LeakyReLU to Sigmoid, Softmax to Sigmoid and LeakyReLU to Mish. Table Results 3 shows the outcome.

5.1 What was easy

The usage of Exploratory Data Analysis (EDA.ipynb) Notebook and the pickle files for single and multi-visit were stored in the GitHub repository. This helped us immensely by eliminating the need to process MIMIC Dataset. Hence the data process was the easy part for us.

5.2 What was difficult

There were a few difficult parts to the project

- Replication with and without GPUs. : Training the model with 5 epochs with 15 iterations as specified in the original paper to be able to re-train, was a time taking process on our Azure VM as well. We had to wait for long hours to see the results. Training the GPT2 model on EHR Dataset on the same machine was even more time taking.
- Architecture Ablations : It took us multiple iterations and tests to improve the GBERT model.
- Replacing BERT with GPT : Both Models have different way of processing and predicting. Hence adaption of the GGPT was another major task for us.

5.3 Recommendations for reproducibility

1. Include the list of requirements using the requirements.txt file.
2. Readme file of Git repo provided minimum information and lacked explaination of various parameter to run GBERT. We have updated the Readme for our project
3. execute the G-Bert with the model already in the repository, use the command:
   cd code
   python run_gbert.py –model_name GBert-predict –use_pretrain –pretrain_dir ../saved/GBert-predict –graph
4. To try the proposed ablations GCN and GTN: in graph_model.py, comment the code where graph is initialized with GAT and uncomment the code
   python run_gbert.py –model_name GBert-predict_qGTN1 –use_pretrain –pretrain_dir ../saved/GBert-predict_qGTN1 –graph

   python run_gbert.py –model_name GBert-predict_qGCN –use_pretrain –pretrain_dir ../saved/GBert-predict_qGCN –graph
5. Execute the new proposed model: execute the Jupyter Notebook: GGPT.ipynb.

6 Contribution Statement

Both project partners contributed equally to the completion of the project. They were in constant discussion to come up with ablations and new feasibility study.

7 Communication with original authors

We reached out to the authors after doing our initial analysis and working on some ablations. The main objective was to show our initial work and seek any early feedback and course correction. We also would have liked to discuss our feasibility study involving Graphs with GPT2 model: GGPT. However the email delivery to two authors failed and we never recieved any response from others.

References

Junyuan Shang, Tengfei Ma, Cao Xiao, and Jimeng Sun. Pre-training of graph augmented transformers for medication recommendation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages

5953–5959. International Joint Conferences on Artificial Intelligence Organization, 7 2019a. doi: 10.24963/ijcai.2019/825. URL https://doi.org/10.24963/ijcai.2019/825.

Junyuan Shang, Cao Xiao, Tengfei Ma, Hongyan Li, and Jimeng Sun. Gamenet: Graph augmented memory networks for recommending medication combination, 2019b.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Yutao Zhang, Robert Chen, Jie Tang, Walter Stewart, and J. Sun. Leap: Learning to prescribe effective and safe treatment combinations for multimorbidity. pages 1315–1324, 08 2017. ISBN 978-1-4503-4887-4. doi: 10.1145/3097983.3098109.

Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F. Stewart, and Jimeng Sun. Gram: Graph-based attention model for healthcare representation learning, 2017a.

Edward Choi, Mohammad Taha Bahadori, Joshua A. Kulas, Andy Schuetz, Walter F. Stewart, and Jimeng Sun. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism, 2017b.

Alistair E W Johnson, Tom J Pollard, Lu Shen, Li-Wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. MIMIC-III, a freely accessible critical care database. *Sci Data*, 3:160035, 24 May 2016.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.