Final Project Draft

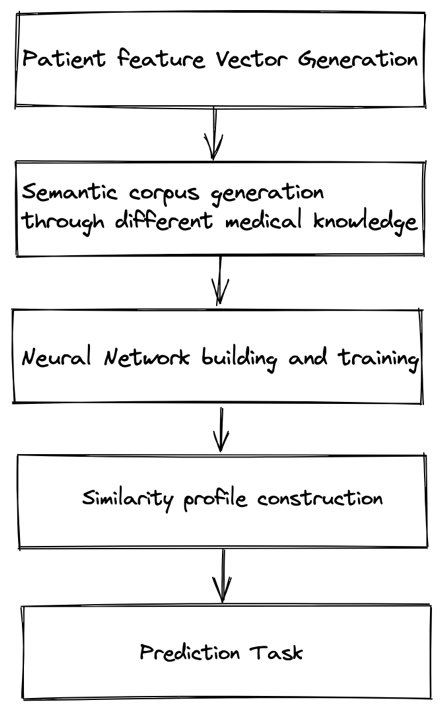
Rezana Ganie and David Alejandro Henriquez  
[rganie3@gatech.edu](mailto:rganie3@gatech.edu) and [dhenriquez3@gatech.edu](mailto:dhenriquez3@gatech.edu)

***Abstract—***Sample Max 5 pages excluding references and appendix

# Introduction

A clear, high-level description of what the original paper is about and what is the contribution of it.

The paper we selected to reproduce is titled; AI-Driven Clinical Decision Support: Enhancing Disease Diagnosis Exploiting Patients Similarity. In this paper, the authors intend to create a Clinical Decision Support System (CDS) that identifies diseases and suggests treatments. Besides, the CDS uses heterogeneous data from multiple sources. For this work, the author focuses on creating a framework physicians can use to that predict a new patients diagnosis based on historic patient symptoms and preliminary diagnosis. The goal of this paper is to build a fast and accurate CDS that is better than existing CDS’s which is limiting by only considering a single medication condition or are built on decision rules.

The paper proposes a CDS framework which consists of 5 steps. First, build a patient feature vector composed of symptoms and preliminary diagnosis from HER (historic) data in sentence formats.

Second, generating a semantic corpus through different medical knowledge (30 million documents from scholarly articles in PubMed and clinical notes in the MIMIC-III Clinical Database). This step included taking data from here.

Third, a single-layer Neural Network is **used** to create word embeddings. This was not created as part of the framework but rather the paper uses a library BioSentVec (an unsupervised version of Fast-Text, and an extension of word2vec (CBOW) to sentences). The reason for this includes the trade-off (speed and complexity) between simple word embeddings and RNNs, LSTMs. BioSentVec produce 700 dimension vectors for sentences (each symptom).

Figure CDS Framework

Fourth, once the vector representations of each patient's symptoms and diagnosis are created a similarity profile is constructed (a cosine-type similarity map is used).

Finally, the fifth step is making the actual predictions by taking the input to predict and computing the similarity between input and historic profiles. In the fifth step, the top k predictions are recorded within a similarity threshold alpha for symptoms and beta for diagnosis.

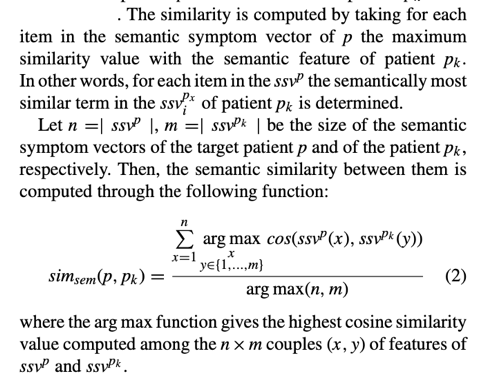
The motivation behind this is “In contrast to more complex neural network based models, one of the core advantages of the proposed technique is the low computational cost for both inference and training”

# Scope of Reproducibility

The results of the experimental evaluation performed with a twofold goal:

* The first one is assessing accuracy via the CDS of the proposed diagnosis prediction approach
* The second goal included assessing how scalable the distributed discovery services

The scope of reproducibility that we committed to include:

1. Creating a similar CDS framework
2. Building a **scalable** system, measured by adding more data and being able to perform the same functions.
3. Implementing a cosine similarity comparator.
4. Using the BioSentVec trained model used in the paper to create symptom vectors.
5. Performing experiments on **accuracy** with Precision and Recall metrics from testing data
6. Reuse of the data extracted from the MIMIIC dataset used by the paper for training and testing.

The results we intend to reproduce are the results concerning the disease prediction based on patient similarity, more precisely the hypothesis that the approach is effective and accurate. The results to reproduce are:

* The precision metric results for different alpha and k.
* The recall metric results for different alpha and k.
* The number of correct predictions for different k and dataset size.
* The scalability testing includes adding more folds of data that is k

In principle, the model would be fed with increasing amount of sample data, with the objective of expanding to the whole MIMIC-III dataset later.

# Methodology

## Model Descriptions

Model description

– Model architecture: layer number/size/type, activation function, etc

– Training objectives: loss function, optimizer, weight of each loss term, etc

– Others: whether the model is pretrained, Monte Carlo simulation for uncertainty analysis, etc.

The pipeline of the model consists of fourth steps: The patient's symptoms and the diagnosis list generator. The feature symptoms list vectorization through the word embedding model BioSent2Vec. The similarity profile construction with the semantic similarity cosine-type function. For a given alpha (symptoms similarity threshold), beta (diagnosis similarity threshold), and k (top k predictions) predicting the patient diagnosis. Finally, the computation of the metrics (precision, recall, and F1-measure).

It is important to recall that the word embedding task uses a pre-trained BioSent2Vec model.

## Data Descriptions

– Source of the data: where the data is collected from

provide the link if possible; if data is synthetic or

self-generated, explain how.

– Statistics: dataset size, cross validation split, label

distribution, etc

– How do you use the data: change the class labels,

split the dataset to train/valid/test, refining the

dataset

The main data source is the MIMIC-III Clinical Database. For the preliminary version of the project, sample data (provided by the article authors) from the MIMIC-III dataset is re-used by us. This data was created by the authors through selecting some rows and saving the data in text file for training and testing. We converted the data to csv formatting for use in our CDS. The data was labelled with ICD9\_code\_diagnosis\_text: [list\_of\_symptoms]. The data did not use correct English Language rules. An example of a row of data:

110244\_Ocl crtd art wo infrct: Gout NOS, Old myocardial infarct.

The sample data consists of 1290 patient admissions with a list of symptoms and diagnoses for each. Besides, this data is spitted 90% for training and 10% for testing.

The final dataset consists of 58976 patient admissions with a list of symptoms and diagnoses for each. This dataset is split into training (80%) and testing (20%).

For each of the datasets, the data is split randomly, and the experiments are performed using 5-fold cross-validation.

## Computational Implementation

– Report the software and hardware implementation (What is your basic coding framework, PyTorch, Tensorflow, etc? What kind of CPU or GPU do you use?)

The main goal of the project was to develop an effective and efficient CDS that produces accurate results with less complexity and time and space costs as compared to pre-existing CDS. This was achieved through intelligently designing a scalable framework that can process huge amounts of data efficiently.   
  
The framework included intelligently storing historic data vectors in order of related closeness and later fetching this with few queries/reads.

Therefore the project was developed in Python using CPU.

– Report hyperparameters including learning rate, dropout rate, number of iterations, training time, etc.

Hyperparameters used :

* k = equaled the number of top similarity matches [1, 5, 10, 15, 20, 25, 30]
* folds = batches of training and testing data [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
* alpha = thresholds used in acceptance of similarity distances between vectors [0.7, 0.8,0.9]

The project was developed in Python using CPU.

## Code

We built a new framework very similar to the original with minor changes, we re-used data for testing and training from the authors. We implemented a patient similarity calculator that matched the paper.

Our framework contained similar 5 step process, however because we re-used testing and training we needed to first convert the input format to csv and then read training and testing data in step1.

Step 2 is the same as original framework.

Step 3 is a further enhancement to storing the data for quick retrieval.

Step 4 uses the same cosine similarity calculation

And step 5 coverts testing data from text to csv, the processes the test set into BioSentVec models and compares the input to our existing historic client base from the framework. Link to code <https://github.com/Rezana20/bd4h-group-project/tree/main/project_draft/code>

The code is written in python and all necessary dependencies are contained in a environment.yml which you can view on our repo.

Download the below trained model from <https://github.com/ncbi-nlp/BioSentVec> “BioSentVec model 21GB (700dim, trained on PubMed+MIMIC-III)”

The readMe.md contains in depth instructions regarding the package and how to run the code

# Results

Report results for all experiments that you run:

We ran an experiment to test the accuracy of the framework. We used Precision calculated as: P= TP / (TP + FP)

TP is the detection of ground truth diagnosis and FP is the number of ground truths mistakenly/incorrectly predicted. We calculate Precision at k. K is the number of top predictions meaning the highest similarity = closest distances of vectors.

Each set of experiments is performed considering 2-fold cross validation. The performance measures reported by the 2-fold cross validation. We then average values in each fold.

In a first set of experiments we aimed to identify the best parameter settings for the prediction algorithm.

– specific numbers (accuracy, AUC, RMSE, etc)

– figures (loss shrinkage, outputs from GAN,

annotation or label of sample pictures, etc) Comparison with the hypothesis and results from the original paper.

# Discussion

The discussion - Make assessment that the paper is reproducible or not.

The paper was fairly difficult for us to reproduce for the following reasons:

* The authors did not do a good job explaining in detail how they implemented the framework so we had to make some assumptions
* The similarity equation was not simple to understand as they used the words cosine similarity but implemented their own variation of what we understand by this (Similarity = (A.B) / (||A||.||B||) )
* They state in the abstract they use a Neural Network, which is misleading because they used a pre-trained model
* Although they had code it was not written using good software engineering principals making it fairly hard to read and understand and having the code posed zero advantage to our implementation

• Explain why it is not reproducible if your results are kind negative.

?

• Describe “What was easy” and “What was difficult”during the reproduction.

Building a scalable application was easy to understand but it was extremely time consuming. The CDS framework proposed was logical and the steps was easy to piece together however exact reproduction was hard because of lack of documentation on the code and lack of explanation of each step in the framework.

• Make suggestions to the author or other reproducers on how to improve the reproducibility.

To reproduce the code, you should upskill on Cython so you could understand how to read their approach. The ReadMe of the code was lacking content and details. This was particularly difficult since the paper did not outline their approach in detail. A lack of documentation.

# References

1. Joyner, D. A., Ashby, W., Irish, L., Lam, Y., Langston, J., Lupiani, I., Lustig, M., Pettoruto, P., Sheahen, D., Smiley, A., Bruckman, A., & Goel, A. (2016). Graders as Meta-Reviewers: Simultaneously Scaling and Improving Expert Evaluation for Large Online Classrooms. In *Proceedings of the Third Annual ACM Conference on Learning at Scale*. Edinburgh, Scotland.