Stevens Institute of Technology

Discover the hidden message of UMLS NCI ontology

Team members:
Qicheng Zhang
Ruixi Wang
Xin Zhao

Course: BIA 667
Advisor: Rong Liu
Date: May 14, 2021

TABLE OF CONTENTS

Introduction

Written during the novel corona-virus pandemic outbreak, this paper aims to accelerate new drug discovery for alleviating the suffering of patients. By applying artificial intelligence models in the medical industry such as node embedding and networks embedding, we are not only able to model disease relationship based on pairwise similarities but also identify drug and diseases phenotypic relationships. As the drugs related to certain diseases or genes are measured, we can lower the cost on R&D of medical science. In other words, the ultimate goal of this project is to benefit drug innovation and development by applying data mining and constructing deep learning model on literature document for effective drug discovery.

Preliminary Literature Review

Various previous studies often model disease relationships based on pairwise similarities. By implement pairwise distance algorithms in NLP, previous disease phenotype networks can successfully identify relations between some diseases and drugs. However, pairwise similarities approach does not preserve the content that how diseases and drugs are related. In another word, the information of diseases phenotype, genes, and other related knowledge is missing. Accordingly, Yang Chen and Rong Xu (2016) continues to contribute research along this line, and they develop a context-sensitive networks (CSNs) approach for disease genetics prediction. The CSNs approach significantly improved the performance of gene prediction in cross validation. Their success in disease genetics prediction provides enormous values for modeling disease relationships. Thereafter, Mengshi Zhou, Chunlei Zheng, and Rong Xu (2020) develop a phenome-driven drug target prediction system based on CSNs approach for predicting novel genetic target for drugs. Therefore, the CSNs approach can be widely used for modeling diseases relationship. The advances in disease phenotype networks means that more research is needed for not only predicting disease gene but also accelerating new drug discovery.

Objectives and Expected Contribution

We generate two research questions:

1. Disease phenotype networks play an important role in computational approaches to identifying new disease-gene associations. But how the similarities are connected andunder what specific context?
2. There are some translational gaps between animal experiments and clinical outcomes inhumans, and the challenge is how to minimize these gaps in order to have better results?

We expect this work can make the following contribution:

1. We will develop a new strategy or upgrade an existing model to create disease associations,so we can have more accurate and faster prediction on new drug discovery.
2. We expect to contribute or develop methods or approaches for disease phenotype networks. This network can provide insight of relationship between diabetes and related drugs. We hope our research can provide additional aid for other research such as diabetes genetic prediction.

Dataset description

In this research study, we had basic understanding of Unified Medical Language System (UMLS), and we've been collecting entities and their relations from UMLS. By using neo4j database with python, we are able to visualize the knowledge graph from UMLS. The picture below is the visualization of drug discovery subgraph:

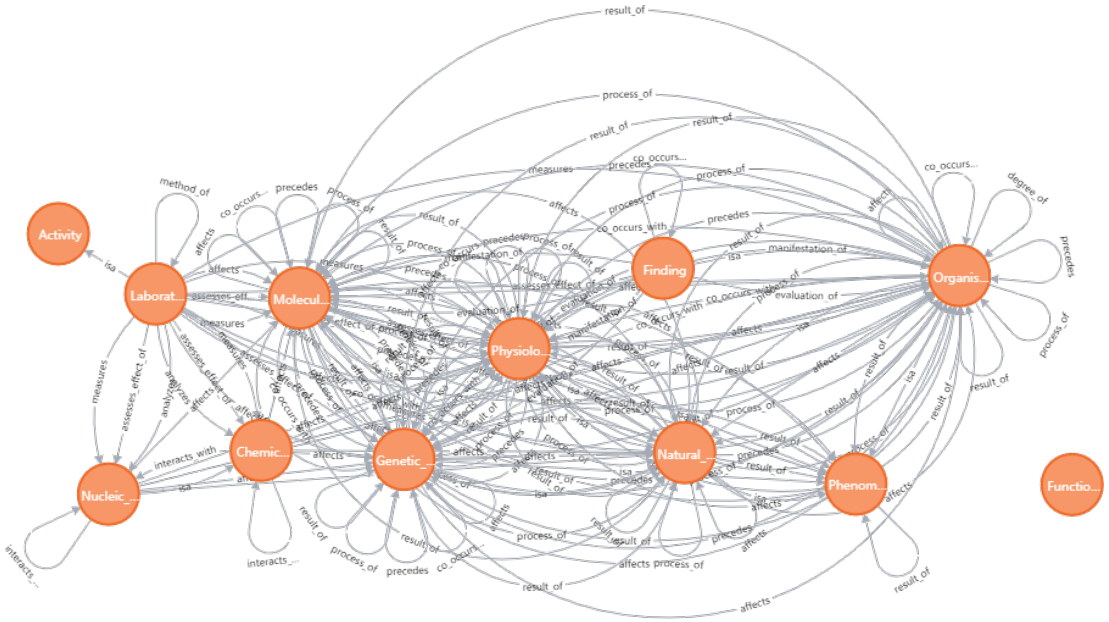

Figure 1, knowledge subgraph

Next, we used a Java toolkit called umls-to-graph-master that can generate knowledge graph of UMLS ontologies. By importing the file "MRHIER.RRF" from UMLS database to the Java coding, we generate 30 knowledge graphs for different ontologies. Among the 30 ontologies, we are attracted by the NCI ontology (National Cancer Institute). According to National Cancer Institute [Internet], "NCI Thesaurus covers vocabulary for cancer-related clinical care, translational and basic research, and public information and administrative activities. NCI Thesaurus provides definitions, synonyms, and other information on nearly 10,000 cancers and related diseases, 17,000 single agents and related substances, and other topics related to cancer and biomedical research. Its multidisciplinary team of editors add about 700 new entries each month." Accordingly, our team decides to use NCI knowledge graph as our dataset for the model.

We discovered 312695 hierarchical relationships in NCI knowledge graph. The output resource file format is a two-column tab-delimited file; the first column is a parent CUI and the second column is a comma-separated list of child CUIs. Note that instead of being stored as strings ("C4228946") the CUIs are stored as integers to save space. For example, the CUI listed as 543 in the subgraph files corresponds to the CUI listed

as C0000543 in the UMLS files.

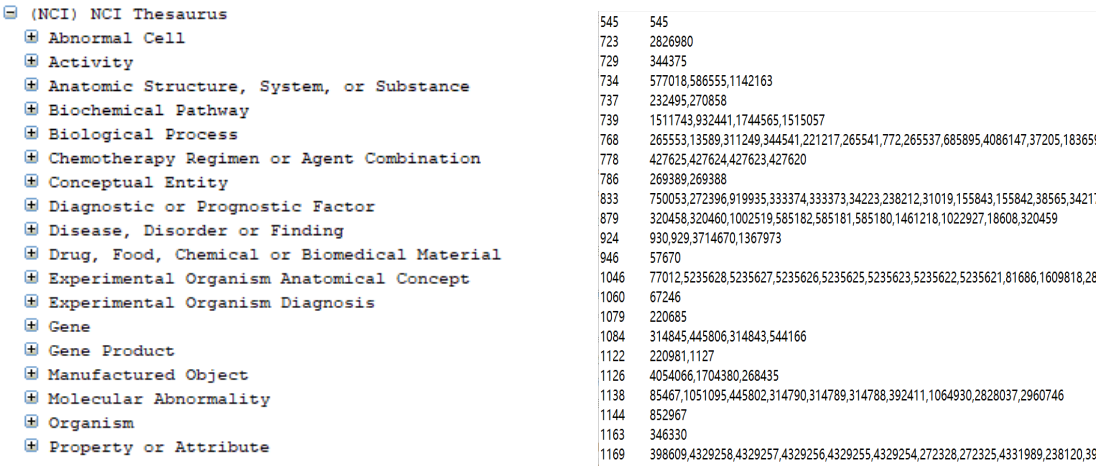

Table 1, dataset description

Data preprocessing

To prepare the NCI data as the proper structure for further node embedding, we must transfer the two-column tab-delimited format into pair-wised relations between parent CUI and corresponding single child CUI. This is achieved by using loop under certain conditions after split the children CUI: if there are more than one children CUI, pairing the parent CUI with each child; otherwise return the pair-wised parent and the child only. Finally store the revised version NCI dataset.

Methodology

We use node2vector neural network to embedding NCI knowledge subgraph. The node2vec is a 2-step representation learning algorithm. It uses random walks to generate data from a knowledge graph. Then, the data is used to learn an embedding vector for each node in the graph. Each node is a unique token in a dictionary that has size equal to the number of nodes in the graph. Then, we utilize the node embeddings as feature vectors for link prediction algorithm. Thus, we can learn unseen relations from link prediction classifier. In addition, we also build deep graph CNN model to classify enzymes or non-enzymes protein. It provides additional information to help explore unseen relations in NCI knowledge subgraph.

Node embedding algorithm

We use Node2Vec representation learning with stellargraph components to calculate node embeddings. The purpose of Node2Vec is to run random walks on the NCI knowledge graph to obtain content pairs, and then using these to train a Word2Vec model.

The first step is to convert our dataset to Stellargraph format. After conversion, our dataset has 151065 nodes and 170661 edges. We set default weight 1 for all edges. Then, we use edge splitter to split our dataset into test graph, test examples, and test labels. We do the same process again with test graph to perform a train test split to produce train graph, train set of link examples, and set of link labels for model selection. A train graph is for computing node embedding. A training set is a classifier of positive and negative edges that were not used for computing node embeddings. It will be later used for link prediction algorithm. The model selection test set is for choosing the best classifier. Lastly, the test graph is to compute test node embeddings. The table below is the summary of different splits:

Split	Number of Examples	Hidden from	Picked from	Use
Training Set	23038	Train Graph	Test Graph	Train the Link Classifier
Model Selection	7680	Train Graph	Test Graph	Select the best Link Classifier model
Test set	34132	Test Graph	Full Graph	Evaluate the best Link Classifier

Table 2, summary of splits

We train the Node2Vec algorithm in two steps. Firstly, we generate a set of sources, target node pairs through starting the biased random walk with a fixed length at per node. The starting nodes are taken as the target nodes and the following nodes in biased random walks are taken as context nodes. We set random walk parameter p and q equal to 1 simultaneously. We set 128 dimensions of node2vec embeddings. We set number of walks from each node to 4, and length of each random walk to 10. Ideally, the number of walks and length is the bigger the better. However, our dataset is relatively large. Accordingly, to save computation cost, we decide to select a relatively small number of walks and length. The second step is to train a word2vec model for

pair prediction. The predictive value is obtained by performing the dot product of the 'input embedding' of the target node and the 'output embedding' of the context node. We decide to set the context window size for word2vec model as 100, and just set to compute 1 epoch for SGD to save computation cost. Lastly, we let multiprocessing cpu_count() function to decide the number of workers for word2vec model.
Finally, we trained two embedding models with training set and testing set. The number of random walks for both training and testing are 604260 respectively.

Link prediction algorithm

The objective of the link prediction algorithm is to explore unseen relations based on our NCI dataset. To achieve this, we set binary operator, 1 is positive, 0 is negative to represent the relations between each node. Accordingly, we can predict all the possible relations for targeting node with some entities. If we provide a pair of nodes, the model should be able to predict how likely they are connected.
There are three primary steps for link prediction model. The first step is to train classifiers with node embeddings. The second step is to select the best performance classifier. Eventually, evaluate the selected classifier on unseen data to validate its ability to generalize.
For training the classifiers, we calculate link-edge embeddings for the positive and negative edge samples by applying a binary operator on the embeddings of the source and target nodes of each sampled edge. Given the embeddings of the positive and negative examples, we can train a logistic regression classifier to predict a binary value indicating whether an edge between two nodes should exist or not.
For evaluating the performance of the classifier, we choose 4 different operators on the training data with node embeddings calculated on the train graph, and select the best classifier. The 4 operators are Hadamard, L1, L2, and average operator.
According to Avnish, "Hadamard product of two vectors is very similar to matrix addition, elements corresponding to same row and columns of given vectors/matrices

are multiplied together to form a new vector/matrix." L1 Norm is the sum of the magnitudes of the vectors in a space. It is the most natural way of measure distance between vectors. The L2 norm calculates the distance of the vector coordinate from the origin of the vector space. As such, it is also known as the Euclidean norm as it is calculated as the Euclidean distance from the origin. Lastly, the average operator is just to calculate the average of the magnitudes of the vectors in a space. As shown in table3, the best operator for our model is L1 norm, which has the ROC/AUC score of 0.83. After we select the best model, we use the test set of embeddings to calculate a final evaluation score. The ROC AUC score on test set using L1 norm is 0.809.

name	ROC AUC score
operator_hadamard	0.789021
operator_l1	0.834384
operator_l2	0.823165
operator_avg	0.685158

Table 3, ROC AUC score for different operators

The picture below is the visualization of our link prediction model. Since we set 128 dimensions for link embeddings, we can use PCA to dimension reduction to 2 dimensions for visualization purpose. The blue points are positive edges, which means two nodes are connected based on our model's prediction, the red points are negative edges, which means the edges should not exist.

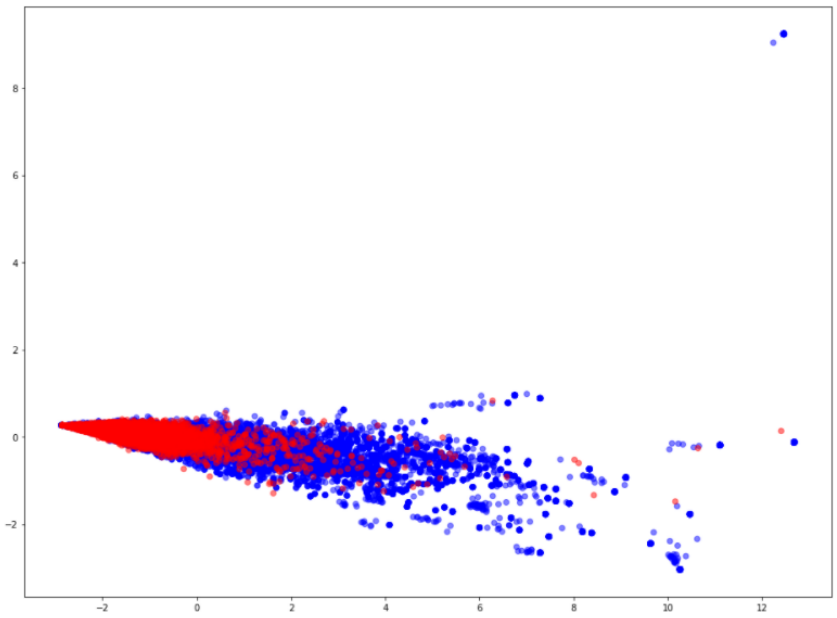

Figure 2, Visualization of link prediction model

Deep Graph CNN algorithm

The objective of using Deep Graph CNN is to train a deep learning model that uses the graph structure of the data together with any information available for the graph's nodes, e.g., chemical properties for the compounds in proteins, to predict the correct label for a previously unseen graph in the training and validating model.

The first step is to import PROTEINS dataset which is exactly a format that DGCNN can ingest. Each graph represents a protein and graph labels represent whether they are enzymes or non-enzymes. The dataset includes 1113 graphs with 39 nodes and 73 edges on average for each graph. Graph nodes have 4 attributes (including a one-hot encoding of their label), and each graph is labeled as belonging to 1 of 2 classes.

In order to feed data to the model that we will create later, we use a data generator called the PaddedGraphGenerator to create the Keras graph classification model. Noticed that the model's input is the graph represented by its adjacency and node features matrices. The first four layers we create are Graph Convolutional using the adjacency normalization D-1 A where A is the adjacency matrix with self-loops and D is the corresponding degree matrix. Those four layers each have 32, 32, 32, 1 units and tanh activations. Then we add one dimensional convolutional layer as the following layer followed by a MaxPool1D layer. This step compresses and smoothies the data. Max-pooling, which is a form of non-linear down-sampling. Partitions then put data into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum value. Next is a second Conv1D layer that is followed by two Dense layers the second used for binary classification. The convolutional and dense layers use relu activation except for the last dense layer that uses sigmoid for classification. We add a Dropout layer after the first Dense layer to fight overfitting. Finally, we create the Keras model and prepare it for training by specifying the loss and optimization algorithm.

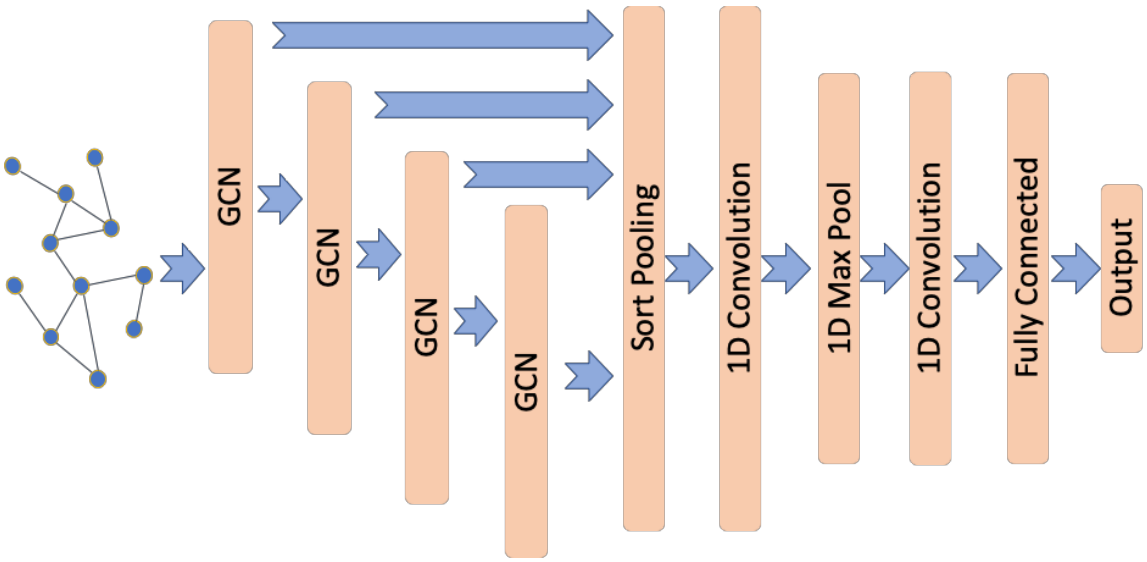

Figure 3, the Keras graph classification model structure

We can now train the model using the model's fit method with training sets. We are going to split the data into 90% for training and the remaining 10% for testing. Given the data split into train and test sets, we create a new generator object that prepares the data for training. We create data generators suitable for training at the model by calling the latter generator's flow method specifying the train and test data. In order to take shorter time to complete the training, we set epochs to a relatively smaller value.

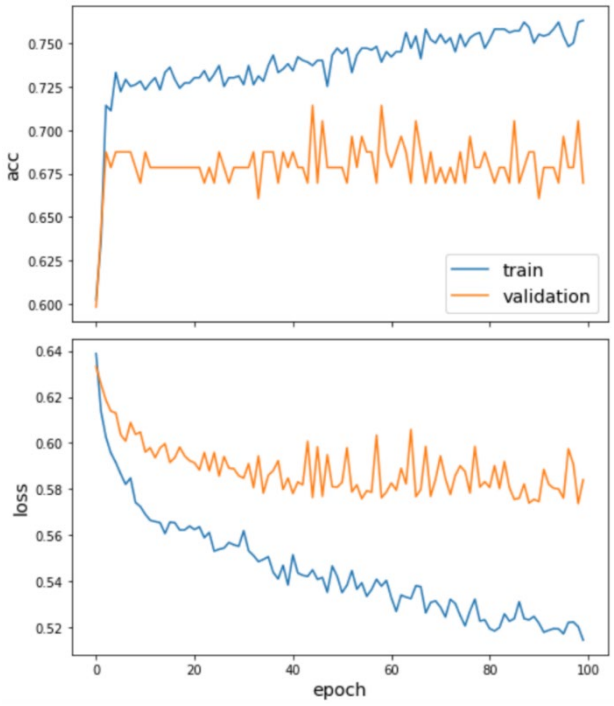

Figure 4, Training validation acc and loss

Finally, we calculate the performance of the trained model on the test data. The results are 0.5841 for loss, and 0.6696 for accuracy. It's easy to conclude that the overall performance of the trained DGCNN model on the test data can predict the trends of binary variable. Even though the DGCNN model cannot precisely predict whether a chemical compound represented as a graph is an enzyme or not based on collection of graphs and label, it does much better job in labeling the new graphs compared with randomly identification. The possible reason of undesired performance is that the computation of some graph properties such as graph moments using message passing networks is impossible without a certain minimum depth. Overall, we currently lack the understanding of which graph properties can be represented by shallow GNNs, which ones require deep models, and which ones cannot be computed at all.

Conclusion

Since we have built a link prediction model, which can predict unseen paths or edges for two random nodes. For example, if the nodes of two cancers are provided, our link prediction model should be able to find out whether they are connected, if they are, then by how much.

Deep Graph CNN model, on the other hand, can predict if the unseen protein is enzymes or non-enzymes. With the help of DGCNN model, we'll be able to explore more NCI knowledge hidden message. Because enzymes are proteins that act as biological catalysts, which can accelerate chemical reactions. When some treatments don't require a fast reaction from drugs, we should be able to figure out what is the suitable drug for a certain disease. For example, Nelarabine, labelled on FDA, is approved for Leukemia, and it contains enzymes, because this kind of cancer requires a mass of reactions during the treatment. If we use a non-enzymes drug on Leukemia, the problem will be unforgivable.

To sum up, with the combination of link prediction model and DGCNN model, now we have made the prediction more accurate and cautious, so it is a worthwhile direction of research. Nothing is more important than curing cancers and saving lives.

Future work

Once the related data of some specific cancer can be captured, for a specific disease, we are able to model cancer relationship between it and other disease based on node embedding, which is beneficial for researchers and clinicians to trace the source of disease and to venture some possible therapeutic treatments. Then further exploring the unseen relations between the target cancer and other NCI nodes from link prediction classifier and digging out the new drugs composition for curing patients. After digging out the components of several effective drugs, we then use deep graph CNN model to check if the protein in any drug contains enzymes or not. Once the drugs is applied to the clinical treatment, comparing the effect between drug with no enzyme and with enzyme;Therefore we can conclude whether enzyme in drugs for specific cancer is more efficient.

Brief project schedule

Our project can be divided into the following tasks and shared among our team members:

Task	Assignee
(1) Data collection	Xin Zhao, Qicheng Zhang
(2) Data preprocessing	Ruixi Wang
(3) Node embedding	Xin Zhao, Ruixi Wang
(4) link prediction	Xin Zhao
(5) Deep graph CNN	Qicheng Zhang, Ruixi Wang

(6) Poster creation	All
(7) Research report writing	All

References:

1. Chen, Y. and Xu, R., 2017. Context-sensitive network-based disease genetics prediction and its implications in drug discovery. Bioinformatics, 33(7), pp.1031-1039.
2. Zhou, M., Zheng, C. and Xu, R., 2020. Combining phenome-driven drugtarget interaction prediction with patients' electronic health records-based clinical corroboration toward drug discovery. Bioinformatics, 36(Supplement_1), pp.i436-i444.
3. National Cancer Institute [Internet]. Bethesda (MD): National Cancer Institute (US). NCI Enterprise Vocabulary Services (EVS); [updated 2015 Jun 12; cited 2017 Oct 12]. Available from: https://www.cancer.gov/research/resources/terminology
4. Avnish. "Part 14 : Dot and Hadamard Product." Medium, Linear Algebra, 10 Aug. 2019, medium.com/linear-algebra/part-14-dot-and-hadamard-product-b7e0723b9133.