A study of Graph input data for deep learning models

1. Introduction and Problem Statement

In today's big data world, AI impacts all of our lives, such as drafting emails, helping us navigate, and recommending us movies of our choice. Deep learning models are trained to perform many different tasks, from playing games to personalized patient care. However, there is still a lot of scope for improvement. In order to make improvements to the model, researchers add more data to the model or increase the complexity of the model. Many research shows that adding more data or increasing model complexity beyond a threshold does not improve model performance and in fact, it decreases model interpretability. Another way to improve the performance of the model is to design the data in a different way which will be more useful for the model. We will see a concrete example of how graph structures can be useful for models.

Obtaining the definition is critical in understanding a new terminology, especially in the scientific community. The lack of precise definitions for terminologies causes many challenges such as discrepancies in understanding terminologies, poor scientific communication, ineffective team collaboration, and hindered discoveries. This problem becomes more severe in emerging research topics, such as COVID-19, in which terminology definitions do not scale to rapidly proposed terminologies. There exist various methods to generate definitions from terminologies such as neural text generation and graph neural networks that leverage relationships between related terminologies which are organized in a graph data structure. Neural text generation is challenging as it remains unclear how to generate text (i.e. definitions) from concise text input (i.e. terminologies). Even though graph neural networks show promising results in definition generation, understanding node relationships and visualizing patterns is challenging in graph structure due to its high complexity.

In our report, we will focus on the below:

1. How a different input data structure can help?
2. How to visualize terminology graph structure effectively?
3. How this visualization relates to the similarity of each pair of terminology definitions?

2. Understanding Data

2.1. Model Performance Data

We used simulated data to create model performance, data size, and interpretability plots. The code shows exactly how the data were simulated. This graph is inspired by the work done by J. Varghese in his paper "Artificial Intelligence in Medicine: Chances and Challenges for Wide Clinical Adoption".

2.2. Graph Data

Terminologies and their corresponding definitions are generally represented in tabular format. Incorporating graphical networks in such data could add more information that will result in better text generation. For our study, we will use the Graphine dataset which contains 2,010,648

terminology definition pairs organized in 227 graphs curated by different domain experts. Each vertex in each graph is associated with terminology and its definition. Each graph is structured from coarse to fine-grained terminologies. We receive the graphs in JSON files in the original dataset as terminologies with their immediate parents. The data will be transformed into matrix form with each cell having a node proximity score between row and column vertices.

3. Matrix-Based Representation

There exist many ways to visualize and analyze networks and graphs in general. The node-link diagrams give comprehensible representations for relatively tiny networks and remain a strong tool for communication. However, obtaining any useful information through node-link diagrams becomes challenging as the size of the graph increases. Several studies have shown that matrices outperform node-link diagrams in a variety of low-level reading tasks, with the exception of pathfinding. Matrices can be utilized to demonstrate high-level patterns by computing good permutations of their rows and columns. Thus, reordering rows and columns help in finding a layout that reveals some structure in the data. Algorithms for ordering the vertices of the graph will be discussed later in the report.

4. Requirements

List of requirements for visualizing graph structure

R1 - Appropriate representation: A representation that is well ordered and easy to understand for large or dense graphs. For our problem statement, terminology definitions can be influenced by nearby vertices' terminology definitions but are not directly linked. We would want to visualize the proximity between two nodes using a Graph Node Proximity Matrix having high scores for vertices that are connected with less number of edges and the highest score for the vertices that are directly connected. Similarly, we want a low score for vertices that are connected via a large number of vertices between them and the lowest score for vertices that are not connected to each other.

R2 - Overview: Obtaining an overview of graph structure is challenging especially for large graphs. These are crucial for understanding the data by the researchers working with such datasets. Overview helps researchers create a mental map of the graphs they are working with.

R3 - Layout: Finding insights requires computing the layout of a graph. It means to compute the ordering of rows and columns in order to create an appropriate matrix, which in turn helps get patterns that the user can readily consume.

R4 - Clusters: Cluster detection for the terminologies is essential to argue that incorporating graphical information contributes to better-performing tasks of definition generation compared to using tabular structure. This is because of the fact that graph neural networks use vertices relationships to generate text and do not solely rely on the terminology.

R6 - Analytical information: Along with obtaining clusters, it is also important to visualize the similarity of terminology definitions of these vertices. A matrix with similarity scores of each pair of terminology definitions can be shown alongside the Graph Node Proximity Matrix.

R7 - Scalability: We would want to visualize hundreds to thousands of vertices simultaneously. Hence, we want to create a scalable solution to visualize millions of vertices relationships.

5. Visual Encoding and Interaction Design

5.1. Model Performance

This is the line chart that shows how quantitative values for performance, data size, and interpretability change with a change in model complexity. These quantitative values are plotted using joined-up lines that effectively connect consecutive points positioned along the y-axis. Performance, data size, and interpretability are displayed in the same view, each represented by a unique line. We showed performance using a green-colored line, data size with a blue-colored line, and interpretability using a yellow-colored line. Both the axes are not shown as the absolute value is not useful. The arrow just after the model complexity tells us that while moving right, the complexity of the model increases.

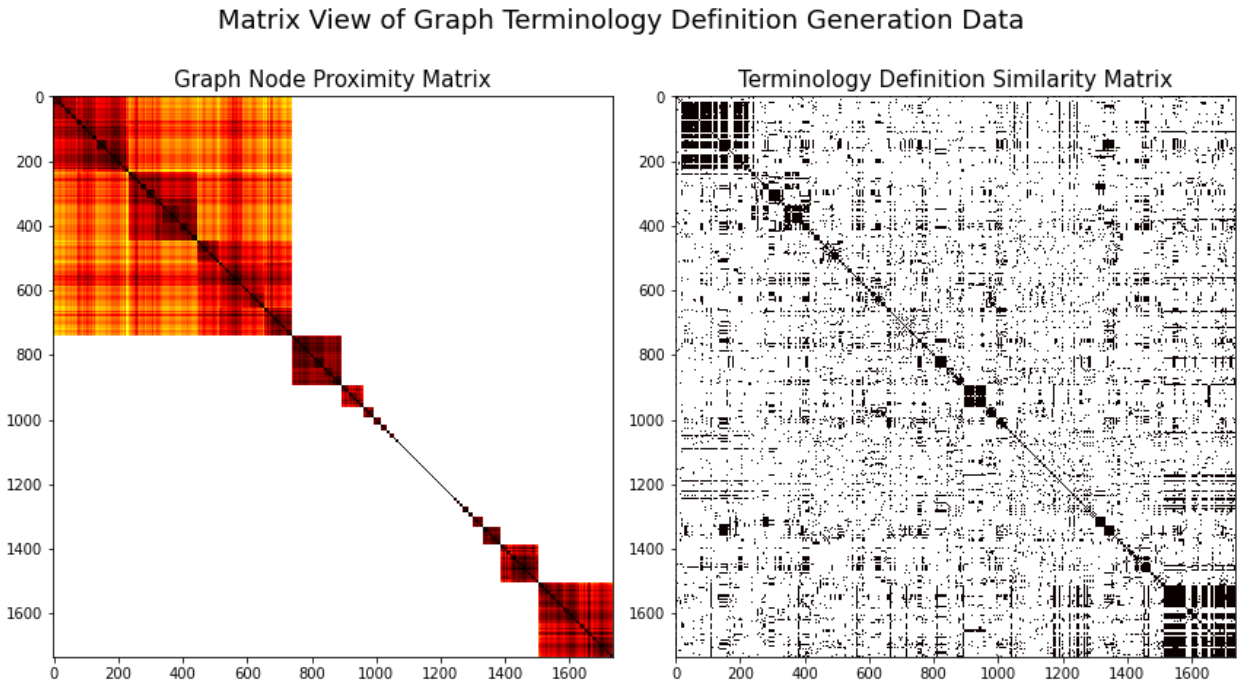
5.2. Graph Node Proximity Matrix

Graph Node Proximity Matrix is visualized using a heatmap chart with two vertices ordered ID as categorical key attributes and vertices' proximity as a quantitative value attribute. The chart shows a 2D matrix alignment of area marks. The ordered vertices ID starts from 0 to the length of vertices - 1, where the association of each ID with terminology and its definition is determined by the ordering algorithm. The ordered vertices IDs are present across the row and column header of the table layout (*R1*). Logical ordering of the vertices along the axis will help readability and obtain key relationships (*R2, R3, and R4*). Each cell is color-coded using a sequential colormap (Yellow-Orange-Red-Black colormap) to represent the vertices' proximity for each pair of vertices. The color scale of the chart mainly facilitates the high-level understanding of the magnitude of vertices' proximity (*R2*). The darker the shade closer the proximity of the nodes. This also helps to visualize the clusters in the graph (*R4*).

5.3. Terminology Definition Similarity Matrix

Terminology Definition Similarity Matrix is also visualized in a similar way as that of the above chart i.e. using a heatmap chart with vertices along the axis in the same ordering as that of the above chart. However, in this chart, we encode the chart in black and white, with black indicating at least one common word between the terminology definitions of the vertices.

The juxtaposition of chart1 with chart2 helps draw conclusions about the node proximity and definition similarity of closer proximity nodes (*R6*). We can see that the nodes closer to each other have more definition similarities. This shows that the model can use definitions of nodes that are closer in proximity for better accuracy. Both the charts are highly scalable as both the charts show thousands of vertices simultaneously (*R7*).



6. Algorithms

6.1. Automatic vertex ordering

We will use the leaf order method for ordering vertices. The method first defines the distance between two vertices and then computes the ordering by constructing hierarchical clustering on the vertices. We define the distance between two vertices as the minimum number of edges between the vertices. Hierarchical clustering considers each vertex as a cluster and then iteratively combines two clusters that are close to each other. This returns an optimal order in which the vertices are to be ordered. Since this method requires no manual effort, it is highly efficient and scalable.

6.2. Computing definition similarity

There are many ways to define similarity between terminology definitions. For our study, we define two terminology definitions as similar when the definitions contain at least one non-stop word common between them.

7. Conclusion

In this study, we show how a different input data structure can help deep learning models. In specific we show how we can effectively visualize large graph networks and how incorporating graph structures can help in different downstream tasks. We see that the clusters formed using the vertices node proximity relationship have high definition similarity. We conclude that terminology definition generation can benefit by using graph structures.

8. References

1. Z. Liu, S. Wang, Y. Gu, R. Zhang, M. Zhang, and S. Wang. Graphine: A Dataset for Graph-aware Terminology Definition Generation. EMNLP, 2021. Link
2. Z. Bar-Joseph, D. K. Gifford, and T. S. Jaakkola. Fast optimal leaf ordering for hierarchical clustering. Bioinformatics, 2001. Link
4. J. Varghese. Artificial Intelligence in Medicine: Chances and Challenges for Wide Clinical Adoption. Link