Introduction:

Objective: Predicting Myer Briggs Personality types using text posts from social media

Myer Briggs Type Indicator

Myer Briggs Type Indicator

The Myers-Briggs Type Indicator (MBTI) is a personality assessment tool that sorts people into 16 different range types of personalities, based on these four main dichotomies. Each of these are represented by a four-letter code, with each letter corresponding to one preference. The MBTI was created by Katharine Cook Briggs and her daughter Isabel Briggs Myers based on the theory of psychological types from Carl Gustav Jung. The four dichotomies and preferences are as follows:

1. Extraversion (E) Vs. Introversion (I);

2. S: Sensing vs N: Intuition

3. Thinking (T) vs. Feeling (F):

4. Perceiving (P) vs Judging (J)

Combining these choices in the experiment caused the production of 16 unique personality categories, each represented by a four-letter code. Here are the 16 MBTI personality types with a brief description of each:

1. ISTJ (The Inspector): Responsible, reliable, and detail-oriented. Prefers structure and order.

2. ISFJ (The Protector): Caring, loyal, and practical. Values harmony and cooperation.

3. INFJ (The Counselor): Idealistic, insightful, and compassionate. Seeks meaning and connection.

4. INTJ (The Mastermind): Strategic, logical, and independent. Values knowledge and competence.

5.ISTP (The Craftsman): Analytical, practical, and adventurous. Prefers hands-on problem-solving.

6.ISFP (The Composer): Artistic, sensitive, and easygoing. Values personal freedom and experiences.

7.INFP (The Healer): Idealistic, creative, and empathetic. Seeks authenticity and personal growth.

8.INTP (The Architect): Intellectual, curious, and unconventional. Values logic and theoretical exploration.

9.ESTP (The Dynamo): Energetic, pragmatic, and spontaneous. Enjoys taking risks and living in the moment.

10.ESFP (The Performer): Enthusiastic, sociable, and fun-loving. Values experiences and connection with others.

11.ENFP (The Champion): Imaginative, enthusiastic, and people-oriented. Values creativity and personal growth.

12.ENTP (The Visionary): Innovative, outspoken, and resourceful. Enjoys debating and exploring new ideas.

13.ESTJ (The Supervisor): Practical, organized, and efficient. Prefers structure and clear rules.

14.ESFJ (The Provider): Warm, cooperative, and responsible. Values relationships and community.

15.ENFJ (The Teacher): Charismatic, empathetic, and supportive. Enjoys helping others and fostering growth.

16.ENTJ (The Commander): Assertive, strategic, and decisive. Values leadership and achievement.

The MBTI is widely used in various settings, including personal development, career counselling, team building, and organizational development. However, it is important to note that while many find the MBTI framework useful for understanding themselves and others, it has also faced criticism for its validity and reliability in the field of psychology**.**

Necessary Application:

This analysis can help with:

- Customer behavior analysis
- Efficiency of employees in the organization
- Depression Detection

Dataset Selection:

Dataset: MBTI 500 dataset

Dataset Construction & Features Mentioned in Kaggle:

Content

106K records of pre-processed posts and their authors' personality types.
Posts are equal-sized: 500 words per sample.

Acknowledgments of Contributors:

- Dylan Storey, who provided a **DATASET** of 1.7M records of posts collected from Reddit using Google big query.
- Mitchell Jolly (datasheet) provided a **DATASET** of 9K of posts collected from the PersonalityCafe forum, where each record has the last 50 posts written by the corresponding user.

Text Data Pre-processing:

Posts are pre-processed texts:

- No punctuations, stop-words, URLs
- Lemmatization
- Reconstruct samples to be equal-sized chunks (500 words per sample)
- Personality types are 16 unique values

Reasons for Choosing this Dataset:

The dataset is relatively large and rich enough for GNN models which need abundant data to learn complex patterns. 106K pre-processed posts are collected as raw texts in this research

2. Constant Post Length: All posts are 500 words long, making the creation of input features for a GNN straightforward.

3. As The Types Of Personality: It is a dataset that may be used to test your multi-class classification on all 16 labels, and therefore the ideal to do so;

4. Prior Pre-processing: Cleaning of the text data, removing punctuation, stop words, and URLs from text as well as lemmatizing it to get a better form, has done to get it into better shape.

5. The data sources are real-world: These textual data come from different and actual places such as Reddit and PersonalityCafe which means - unlike synthetic corpus creation, this diverse range of linguistic styles helps fix the overfitting of language-specific patterns thereby enhancing its generalization potential.

6. Reformed Equal-sized Chunks: The reforming creates the dataset as equal-shaped chunks, making it a kind of standard input to have consistency in constructing each graph for GNN.

This high-quality and well-prepared data from multiple sources will be useful to learn GNNs as a baseline for text classification tasks because the datasets follow strict guidelines in order to make model development ad evaluation robust.

Defining the Problem as Graph Dataset:

The text classification task uses Graph Neural Networks (GNNs).

• Nodes show text posts from the dataset. Each post is a node in the graph.

•Node features are the traits or qualities linked to every node. In this case, the node features come from:

1. TF-IDF Features: These show how important words are in each post capturing the text content. The system also figures out cosine similarities between post texts.

2. Average Number of Words: This feature shows the average word count of each post. These features combine to create a feature vector for each node.

Edges

- Edges show the links or connections between nodes. Here, the system makes edges based on how similar the TF-IDF vectors of the posts are.
- The system creates an edge between two nodes if their cosine similarity is more than 0.5. This means that nodes (posts) with similar text connect.

Data Object

- Data(x, edge_index y): A PyTorch Geometric Data object holds the built graph.
- x: Node feature matrix where each row shows a node's features.
- edge_index: Edge index matrix showing the connections between nodes. Edges
- Edges represent the relationships between nodes. Here, edges are created based on the cosine similarity between the TF-IDF vectors of the posts.
- An edge is created between two nodes if their cosine similarity exceeds a certain threshold 0.5. This means that nodes (posts) that are textually similar are connected.
- y: Labels for each node, representing MBTI types.

Graph Problem Definition

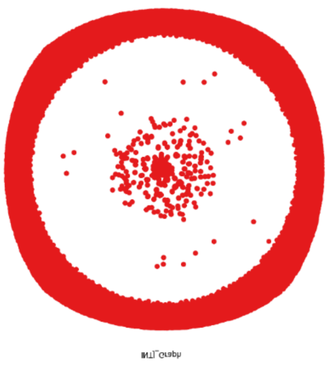
- Nodes: Each node indicating the post contains 500 words of text.
- Edges: Connection between two nodes based on cosine similarity between TF-IDF matrixes in posts.
- Node Features: Combined vector of TF-IDF features and the average number of words.
- Labels: MBTI personality type for each post.

This is beneficial for tasks like classification, where understanding both individual posts and their relationships can improve model performance.

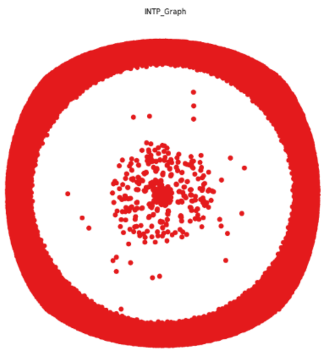
Graphs for each MBTI type:

The visualizations follow :

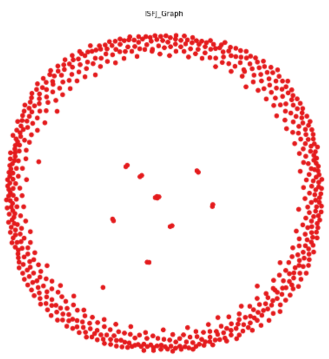
- Circular layout
- Connection based on threshold of cosine similarity among posts
- Higher the cosine similarity, closer the connection between nodes reporesenting each post



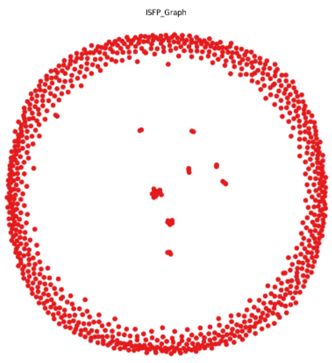
Type: INTJ



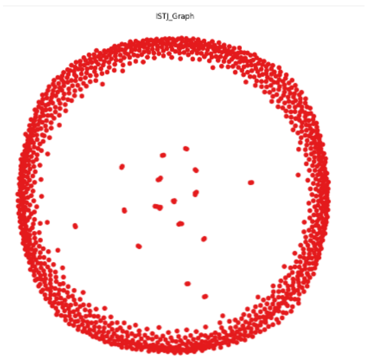
Type: INTP



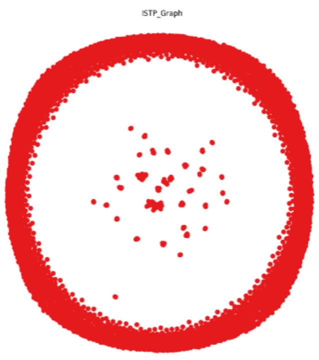
Type: ISFJ



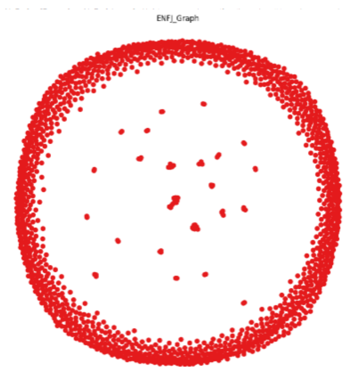
Type: ISFP



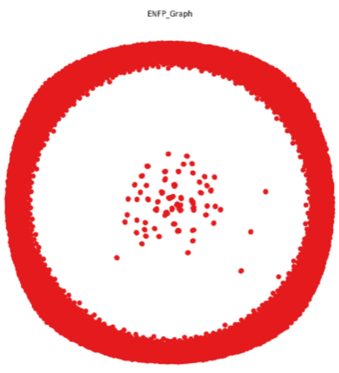
Type: ISTJ



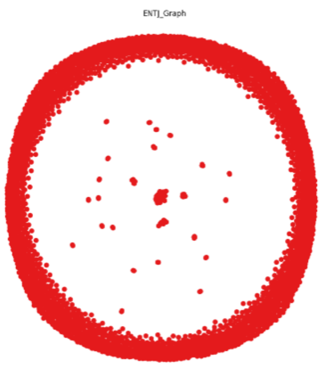
Type: ISTP



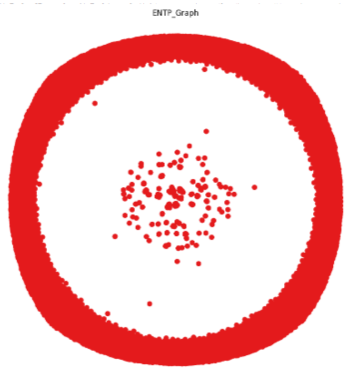
Type: ENFJ



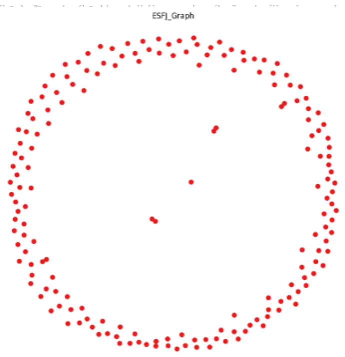
Type:ENFP



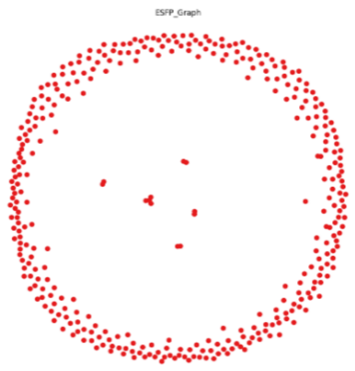
Type: ENTJ



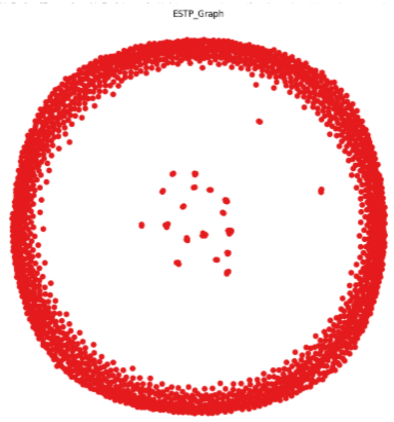
Type:ENTP



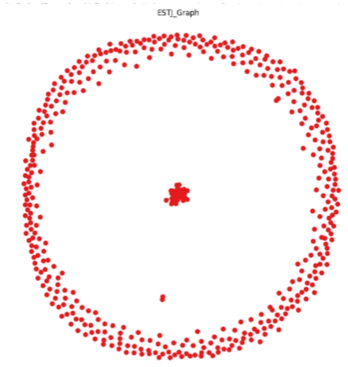
Type:ESFJ



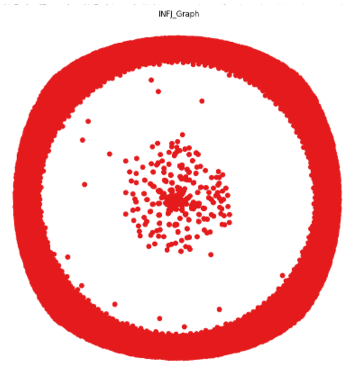
Type: ESFP



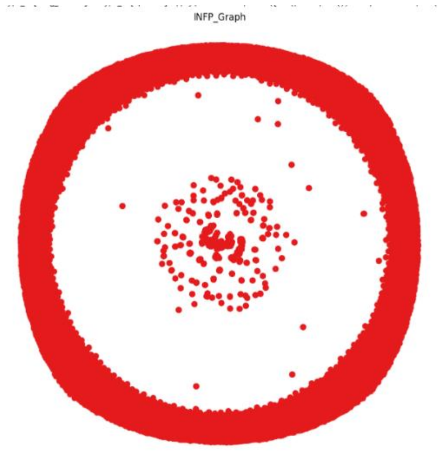
Type: ESTP



Type: ESTJ



Type: INFJ



Type: INFP

Observing the centralized and peripheral clusters in the graph, the following insights can be derived:

- INTJ, INTP, INFJ .. these categories have more similar content centralized.
- INFPs also have a fair share of similar content.
- Comparatively ESFJ, ESFP, ESTJ … these categories have a lack of similarity in posts.

Pre-processing the Graph Dataset:

- **Graph Combination and Shuffling**
  1. **Combining Graphs**: Individual subgraphs for each MBTI type are getting connected together into a single graph. Node indices and edge indices are adjusted accordingly.
  2. **Shuffling:** The nodes of connected graphs are shuffled to ensure permutation of distribution. Edge indices remain fixed to save graph structure.

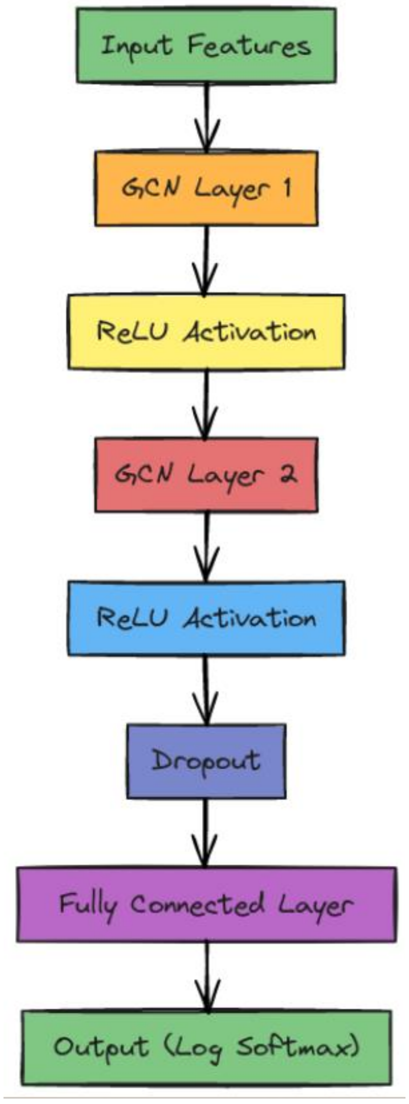
- **Data Splitting**
  1. **Training, Validation, and Test Sets:** The shuffled graph is split into training, validation, and test sets based on specified ratios (60% training, 20% validation, and 20% test). Boolean masks are created to indicate the membership of nodes in each set.

Section 2: Appropriateness & Explanation of Model

I have experimented with two types of models after studying carefully. The models are:

GCPModel Using GCN Layers [1]

Architecture



The model uses Graph Convolutional Network (GCN) layers to understand the connections and traits of nodes in a graph.

1. GCNConv Layers:

• First GCN Layer (self.conv1):

1) Input: Node traits.

2) Operation: Applies a graph convolution to create hidden node representations by collecting info from nearby nodes.

3) Output: Changed node traits into a set size (given dimension)

•Second GCN Layer (self.conv2):

1) Input: What comes out of the first GCN layer.

2) Operation: Makes node representations even better by gathering info from nodes' neighbors in given dimension

3) Output: Node traits changed again to a set size

2. Connected Layer (self.fc):

• Input: What comes out of the second GCN layer.

•Operation: Changes the node traits into the final output size, which matches the number of groups (16 MBTI types).

•Output: Scores for each group for every node.

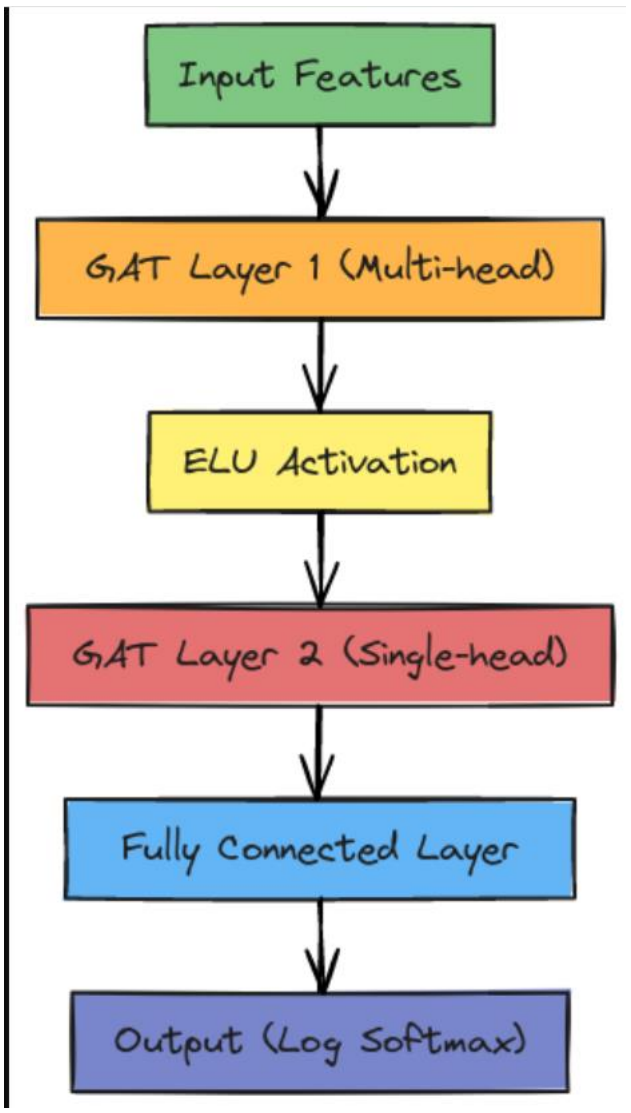
Activation and Dropout:

•ReLU Activation: The network applies this after each GCN layer to add non-linearity and boost its ability to learn complex patterns.

•Dropout: This technique helps to avoid overfitting. It works by cutting off some connections at random during the training process.

2. GATClassifier Using GAT Layers [2]

Architecture:



The GATClassifier leverages Graph Attention Networks (GAT) to dynamically learn the importance of neighbouring nodes. Here’s a detailed breakdown:

GATConv Layers:

- First GAT Layer (self.conv1):
  1)Input: Node features.
  2)Operation: Uses several attention heads to figure out how important each nearby node is and combine their features. The dropout setting helps to make the model more robust.
  3)Output: All attention heads' outputs joined together, changed node features of a specific size (hidden_dim)

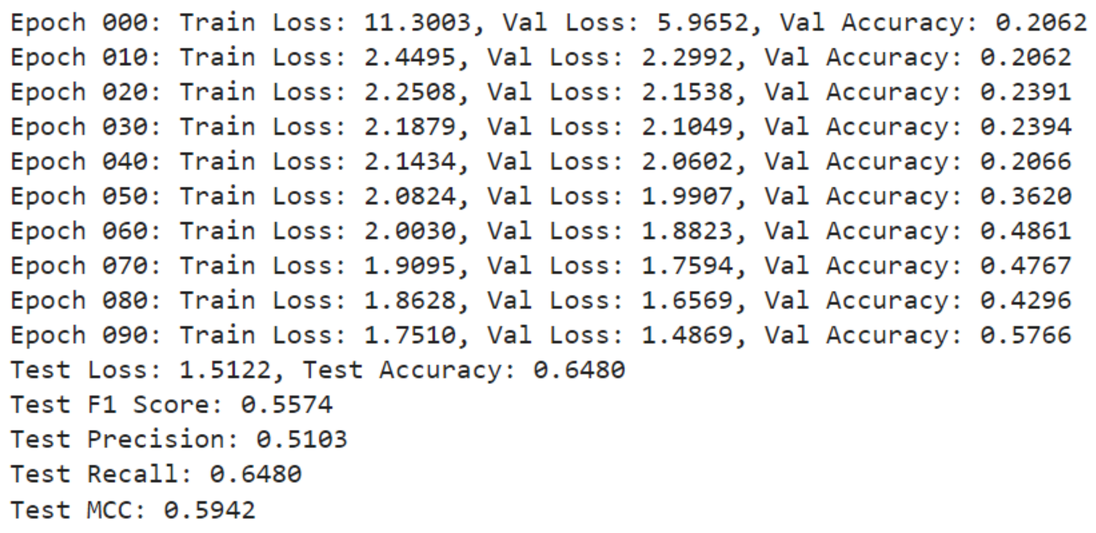
- Second GAT Layer (self.conv2):
  1)Input: Joined outputs from the first GAT layer.
  2)Operation: Uses one attention head to improve the node representations more while keeping the good points of attention methods.
  3)Output: Changed node features of size hidden_dim.

2. Connected Layer (self.fc):

•Input: Output from the second GAT layer.
•Operation: Changes the improved node features into the final output size for sorting.
•Output: Scores for each class for every node.
Activation and Dropout:
•ELU Activation: This step happens after the first GAT layer. It adds non-linearity and helps solve the vanishing gradient issue. As a result, it speeds up the learning process.
•Dropout: This technique is used inside the GAT layers. It serves to regulate the model and stop it from overfitting.

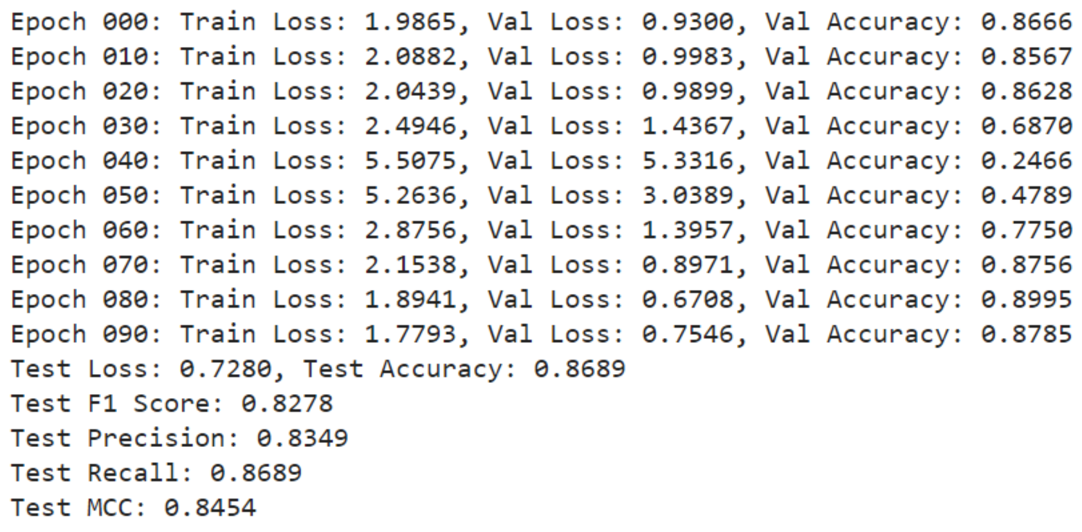
The performance metric of Both Models:

Performance of GCPModel:



```
Epoch 000: Train Loss: 11.3003, Val Loss: 5.9652, Val Accuracy: 0.2062
Epoch 010: Train Loss: 2.4495, Val Loss: 2.2992, Val Accuracy: 0.2062
Epoch 020: Train Loss: 2.2508, Val Loss: 2.1538, Val Accuracy: 0.2391
Epoch 030: Train Loss: 2.1879, Val Loss: 2.1049, Val Accuracy: 0.2394
Epoch 040: Train Loss: 2.1434, Val Loss: 2.0602, Val Accuracy: 0.2066
Epoch 050: Train Loss: 2.0824, Val Loss: 1.9907, Val Accuracy: 0.3620
Epoch 060: Train Loss: 2.0030, Val Loss: 1.8823, Val Accuracy: 0.4861
Epoch 070: Train Loss: 1.9095, Val Loss: 1.7594, Val Accuracy: 0.4767
Epoch 080: Train Loss: 1.8628, Val Loss: 1.6569, Val Accuracy: 0.4296
Epoch 090: Train Loss: 1.7510, Val Loss: 1.4869, Val Accuracy: 0.5766
Test Loss: 1.5122, Test Accuracy: 0.6480
Test F1 Score: 0.5574
Test Precision: 0.5103
Test Recall: 0.6480
Test MCC: 0.5942
```

Performance of GATClassifier:



```
Epoch 000: Train Loss: 1.9865, Val Loss: 0.9300, Val Accuracy: 0.8666
Epoch 010: Train Loss: 2.0882, Val Loss: 0.9983, Val Accuracy: 0.8567
Epoch 020: Train Loss: 2.0439, Val Loss: 0.9899, Val Accuracy: 0.8628
Epoch 030: Train Loss: 2.4946, Val Loss: 1.4367, Val Accuracy: 0.6870
Epoch 040: Train Loss: 5.5075, Val Loss: 5.3316, Val Accuracy: 0.2466
Epoch 050: Train Loss: 5.2636, Val Loss: 3.0389, Val Accuracy: 0.4789
Epoch 060: Train Loss: 2.8756, Val Loss: 1.3957, Val Accuracy: 0.7750
Epoch 070: Train Loss: 2.1538, Val Loss: 0.8971, Val Accuracy: 0.8756
Epoch 080: Train Loss: 1.8941, Val Loss: 0.6708, Val Accuracy: 0.8995
Epoch 090: Train Loss: 1.7793, Val Loss: 0.7546, Val Accuracy: 0.8785
Test Loss: 0.7280, Test Accuracy: 0.8689
Test F1 Score: 0.8278
Test Precision: 0.8349
Test Recall: 0.8689
Test MCC: 0.8454
```

Significance of the metrics:

The significance of evaluation metrics like F1 score, recall, precision, and Matthew's Correlation Coefficient (MCC) is crucial for assessing the performance of classification models, specially for imbalanced datasets.

Precision

Significance: Precision tells us how many of the predicted positive instances were actually positive. High precision means that a model has a low false positive rate.

- Precision score of GCN model: 0.51
- Precision score of GATClassifier model: 0.83

Recall

Significance: Recall tells us how many of the actual positive instances were correctly predicted by the model. High recall is important in cases where missing a positive instance is costly

- Recall score of GCN model: 0.65
- Recall score of GATClassifier model: 0.86

F1 Score

Significance: The F1 score balances the trade-off between precision and recall. It is a useful metric when balance between precision and recall is necessary, and especially in situations where the class distribution is imbalanced.

- F1 score of GCN model: 0.56
- F1 score of GATClassifier model: 0.84

Matthew's Correlation Coefficient (MCC)

Significance: MCC takes into account all four categories of the confusion matrix (true positives, true negatives, false positives, and false negatives) and is generally regarded as a balanced measure even if the classes are of very different sizes. An MCC of +1 indicates a perfect prediction, 0 indicates a random prediction, and -1 indicates total disagreement between prediction and observation.

- MCC of GCN model: 0.59
- MCC of GATClassifier model: 0.82

Each of these metrics provides different insights, and choosing the right one depends on the specific context and goals of the classification task.

Possible reasons why the GATClassifier Architecture is performing well:

- Attention Mechanism:

  Graph Attention Transformer (GAT) utilizes attention mechanism to learn the importance of each neighbor node dynamically. As the label of some nodes can be more useful than others when training GNNs, M-GAT advises the model to pay attention only to relevant input parts instead of aggregating irrelevant features during layer processing; it maintains a concision over appropriate information and an elimination over superfluous content which in turn enhances the quality level where node embeddings are learned.

- Multiple Attention Heads:

  Given attention heads in the first GAT layer, The use of multiple attention heads stabilizes learning and it allows for modeling different signals from the neighborhood. This duplication helps to reduce the influence of data that is noisy or missing.

- Dynamic Aggregation:

  The proposed method extends neural Word Embeddings to characterize structural aspects of neighborhood aggregation. GCN aggregates a fixed number of neighbours, GAT can dynamically aggregate neighbours.

- Improved Non-linearity:

  The ELU activation function injects non-linearity to remove the vanishing gradient problem and speeds up learning. Better convergence and improved performance are led.

Code Link & Instruction:

- Code file link: google drive link
- Platform suitable: Google colab
- Environment: TPU(More Disk Storage for quick execution)

-
  ```
  # the MBTI 500 dataset path in my drive folder
  df = pd.read_csv('/content/drive/MyDrive/Dataset_DSAPP/MBTI 500.csv')
  ```

  Please replace this path with the exact path from where the file is saved in google drive or any local folder.
- Please run all the cells one by one.

Reference:

- GRAPH ATTENTION NETWORKS By Petar Velickovi ˇ c´ ∗, Department of Computer Science and Technology ,University of Cambridge , Guillem Cucurull∗ Centre de Visio per Computador, UAB , Arantxa Casanova∗ Centre de Visio per Computador, UAB ´ ,Adriana Romero Montreal Institute for Learning Algorithms ´ Pietro Lio` Department of Computer Science and Technology University of Cambridge ,Yoshua Bengio Montreal Institute for Learning Algorithms ´

- SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS By Thomas N. Kipf ,University of Amsterdam ; Max Welling ,University of Amsterdam Canadian Institute for Advanced Research (CIFAR)

- kaggle link