HW3

313312753 208602557 209722537

July 11, 2024

Link to GIT

Introduction

In the field of academic research, the volume of published articles has been increasing rapidly in recent years. This growth has created a need for efficient tools that can sort and categorize these articles, making them easily accessible to researchers. To address this need, our task is to predict the categories of academic articles.

The dataset provided for this task is represented as a directed graph, which serves as a network of academic citations. The graph consists of approximately 100,000 vertices, with each vertex representing an article. The directed edges between vertices indicate citation relationships: when one article (vertex A) cites another article (vertex B) we have edge (A,B).

To assist in predicting the categories of the articles, each article is accompanied by a feature vector. This feature vector is created by averaging all the word representations found in the abstract and title of the article, which are generated using the skip-gram model. By leveraging these feature vectors, you can infer relationships and patterns within the dataset that contribute to the categorization of articles.

In addition to the feature vectors, each article in the dataset is associated with the year of its publication and a numerical representation of its category. These category numbers indicate the specific field or subject area to which each article belongs. By utilizing the provided data, we can train a model to predict the appropriate category for new, unseen articles.

By accurately predicting the categories of articles, we can contribute to the development of automatic tools that facilitate the organization and accessibility of academic publications. Such tools will greatly benefit researchers in their pursuit of relevant and up-to-date information in their respective fields of study.

This form of problem was new to us and brought along with it types of data and challenges that we had not yet faced. We needed to find ways to incorporate the features as well as the inherent structure of the data. Utilizing the edge index was of particular importance. As with most projects, we knew to not try and re-invent the wheel; turning to solutions that have been tried and tested proved, as always, to be a strong tactic.

1 Models

We considered 3 different models that we believed can be useful for our task, ChebConv, GCNConv, and GATConv.
ChebConv, GCNConv, and GATConv are all popular graph convolutional neural network (GCN) architectures used for graph-based learning tasks. They each have their own unique characteristics and strengths:

1.1 ChebConv (Chebyshev Convolution)

- ChebConv utilizes Chebyshev polynomials to perform graph convolutions.
- It approximates spectral graph convolutions by truncating the expansion of the spectral filters.
- ChebConv is computationally efficient and can handle large graphs.
- It can capture localized patterns in graph structures.

- ChebConv is limited in terms of capturing global graph information and can suffer from over-smoothing issues.

1.2 GCNConv (Graph Convolutional Network Convolution)

- GCNConv is a simple and widely-used graph convolutional operation.
- It aggregates node features by considering the features of neighboring nodes.
- It applies a linear transformation to the node features followed by a non-linear activation function (usually ReLU).
- GCNConv captures local information effectively, making it suitable for tasks where local connections are important.
- GCNConv does not consider edge weights and can be sensitive to the ordering of nodes.

1.3 GATConv (Graph Attention Convolution)

- GATConv introduces the concept of attention mechanisms to graph convolutions.
- It assigns attention weights to the neighboring nodes, allowing the model to focus on more relevant nodes during aggregation.
- GATConv captures both local and global information, as it attends to different nodes adaptively.
- It can handle graphs with varying sizes and structures effectively.
- GATConv provides state-of-the-art performance on various graph-based learning tasks.
- GATConv is computationally more expensive compared to other methods due to the attention mechanism.

In short, ChebConv is efficient and captures localized patterns, GCNConv is simple and effective for local connections, and GATConv incorporates attention mechanisms for adaptive aggregation and better global information capture. The choice of the method depends on the specific requirements of the graph-based learning task and the trade-off between computational complexity and performance.

2 Attention

Attention convolution is a technique that can be employed to enhance predictive accuracy. It seemed to be a useful mechanism to employ in our task. "Attention convolution" refers to the integration of attention mechanisms within convolutional neural networks (CNNs). While attention in CNNs is usually used to understand how sections of images interact, especially in mixed tasks such as Image based question answering. In this approach, the model dynamically assigns weights to different parts of the input data, allowing the network to focus on the most relevant features. By applying attention convolution to the feature vectors derived from the article abstracts and titles, the model can effectively identify and prioritize important information for predicting the article categories. This technique helps capture the inherent relationships and dependencies between words and features, enabling more accurate predictions.

Data Exploration

Class Distributions

We were interested to understand the distribution of the class labels. In total we were given data with 40 possible fields of research, each represented by a number between 0 and 39. We found that the distribution of research fields amongst the data was far from uniform: 2. A handful of classes made up a large majority of the classes - this class imbalance introduced yet another challenge to our prediction task. We would need techniques that would learn to classify the smaller classes quickly - as the model would not see a large amount of these samples.

Self-Citations

Having seen the imbalance in the labels we set out to see if this imbalance was represented in a homogeneous citation: Did common fields tend to cite within their own field? See 3. We defined a "self-citation" as an edge (i, j) where i and j are from the same field of research (share the same label). We see that the two largest classes (16 and 28) have a very homogeneous citation pool (with over 80% of their citations being self-citations). While the rest of the classes did not show as strong of a homogeneous tendency - more than half of the classes still had a self-citation proportion of over 50%. This could prove to be an important feature in a task. We hypothesize that this contributed to the success of the GAT model that makes use of global information such as dense sub-graphs.

Neighbor Counts

In order to account for the class imbalance we analyzed the log of the citation counts. See 4. Here too we see a strong tendency for self-citation. While this information was not immediately useful as is. We hoped that the model might be able to pick up on dense sections of the test-data graph and utilize the information for making predictions. Future work could be aimed at exploring the utility of dense sub-graphs. For our project, we relied on the abilities of our GATConv model.

Table 1: Graph Information

Number of features	128
Number of classes	40
Number of nodes	100,000
Number of edges	444,288
Average node degree	4.44
Number of training nodes	80,000
Training node label rate	0.80
Has isolated nodes	True
Has self-loops	False
Is undirected	False

Experiments and hyperparameter tuning

Training different models

We trained all three models mentioned above, with different parameters and measure their performances.

2.0.1 ChebConv

- Epochs: We have tested the performence up to 500 epochs.
- Hidden layers: We checked 3-5 layers, in sizes of 512-64.
- Activation function: We examine both relu and tanh functions.
- Learning rate: We tried range of 0.005-0.0003.
- K: The The order of Chebyshev polynomial.checked orders in range 2-5.
- Normalization: Symmetric normalization.
- Dropout: Range of $p \in 0.2, 0.6$, while in the first layers the dropout is higher.

2.0.2 GCNConv

- Epochs: We tested the performance up to 1000 epochs.
- Hidden layers: We checked 3-5 layers, in sizes of 512-64.
- Activation function: We examine both relu and tanh functions. relu performed better.

- Learning rate: We tried range of 0.005-0.0003.
- Normalization: Symmetric normalization.
- Dropout: Range of $p \in [0.2, 0.6]$, while in the first layers the dropout is higher.

2.0.3 GATConv

- Epochs: We have tested the performence up to 1000 epochs.
- Hidden layers: We checked 2-4 layers, in sizes of 512-64.2 layers were better than more.
- Heads: Number of multi-head-attentions, 4-8.
- Activation function: We examine both relu and tanh functions. tanh performed better.
- Learning rate: We tried range of 0.005-0.0003.
- Normalization: Symmetric normalization.
- Dropout: Range of $p \in [0.2, 0.6]$, while in the first layers the dropout is higher.

We also implemented a learning rate schedule that gradually reduces the learning rate over time. This can help the model converge more effectively by allowing smaller steps towards the end of training. unfortunately, it did not improve the results.

2.1 The Final Model

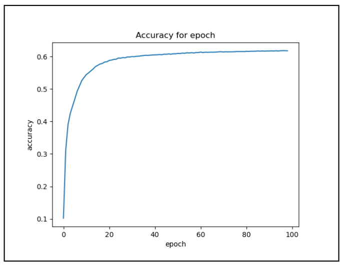
We chose to use GATconv model, which showed the best performance on the validation set.

- Epochs: 500
- Hidden layers: 2 layers, hidden channels 256
- Heads: 4
- Activation function: relu
- Learning rate: 0.0003
- Normalization: symmetric
- Dropout: 0.4

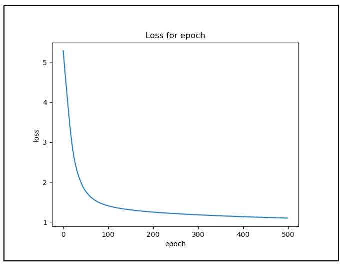
We normalized each entry by the standard - normal normalization.

3 Results

All three model reached easily to 0.6 accuracy on the validation set, while the losses were in range of $1.0 - 1.5$ after a few hundreds of epochs. The GATconv model converged the fastest and with a significantly lower loss value. The final accuracies were a bit higher than the other two models, the results can be found in fig. 1. This model uses attention - affirming our beliefe that the attention mechanism might be the best way to capture the graph complexity. It can be seen that both



(a) Accuracy over epochs



(b) Loss over epochs

Figure 1: Performance results GATconv model

accuracy and loss graphs are monotonically increasing and decreasing respectively, both measures starts with big steps and reach to convergence.

4 Post-Hoc Analysis

Based on the classification report 2, we can perform a post hoc analysis to gain insights into the performance of your model. Here are a few observations:

1. Precision: Precision measures the accuracy of positive predictions made by the model. Classes such as 4, 16, 28, and 30 have relatively high precision scores, indicating that the model performs well in correctly identifying instances belonging to these classes.
2. Recall: Recall measures the ability of the model to correctly identify positive instances from the dataset. Classes like 2, 8, 16, and 28 have high recall scores, suggesting that the model effectively captures most instances from these classes.
3. F1-Score: The F1-score is the harmonic mean of precision and recall, providing a balanced measure of the model's performance. Classes with high F1-scores, such as 2, 16, and 28, indicate that the model achieves a good balance between precision and recall for these classes.
4. Support: Support represents the number of instances belonging to each class. Some classes, like 16 and 28, have a high support value, indicating a significant presence of instances from these classes in the dataset.
5. Accuracy: The overall accuracy of the model is 0.6181, which means it correctly classifies approximately 61.81
6. Macro Avg: The macro average provides the average performance across all classes, giving equal weight to each class. The macro average precision, recall, and F1-score are 0.4801, 0.4121, and 0.4280, respectively.
7. Weighted Avg: The weighted average takes into account the class imbalance in the dataset by weighting each class's performance based on its support. The weighted average precision, recall, and F1-score are 0.6057, 0.6181, and 0.6078, respectively.

Based on this analysis, it appears that your model performs well on certain classes with high precision, recall, and F1-scores. However, there are also classes where the model's performance is relatively low. It might be worth investigating and potentially improving the model's performance on these specific classes. Additionally, the overall accuracy suggests that there is room for improvement in the model's classification performance. Further analysis, feature engineering, or model fine-tuning could help enhance the model's accuracy and overall performance.

5 Summary

We faced a task of predicting the categories of academic articles using a graph-based dataset. The dataset consisted of a directed graph representing academic citations, with each vertex representing an article and the edges indicating citation relationships. Each article was accompanied by a feature vector derived from the abstract and title of the article.

We explored three different graph convolutional neural network (GCN) architectures: ChebConv, GCNConv, and GATConv. These models offered unique characteristics and strengths for graph-based learning tasks. After training and evaluating these models, we found that the GATConv model, which incorporated attention mechanisms, outperformed the other models in terms of accuracy and convergence speed.

We also conducted data exploration to gain insights into the dataset. We observed an imbalance in the class distributions, with a few classes representing the majority of the articles. Additionally, we investigated self-citations and found that articles often cited others within the same research field.

The experimental results showed that the GATConv model achieved an accuracy of approximately 61.81% on the validation set. Although the accuracy was moderate, the model demonstrated the potential to predict article categories effectively. The precision, recall, and F1-scores varied across different classes, indicating variations in the model's performance for different research fields.

In conclusion, our study highlights the importance of incorporating attention mechanisms in graph convolutional neural networks for predicting article categories. The attention mechanism allowed the model to focus on relevant features and capture intricate relationships within the graph structure. While the model's performance can be further improved, our findings contribute to the development of automatic tools that organize and facilitate access to academic publications. Such tools can greatly benefit researchers in their pursuit of relevant and up-to-date information in their respective fields of study.

Overall, our work demonstrates the potential of graph-based learning approaches and attention mechanisms in the field of academic research categorization. Future research can focus on refining the models, addressing class imbalance issues, and exploring additional features or techniques to enhance prediction accuracy and generalization.

Appendix

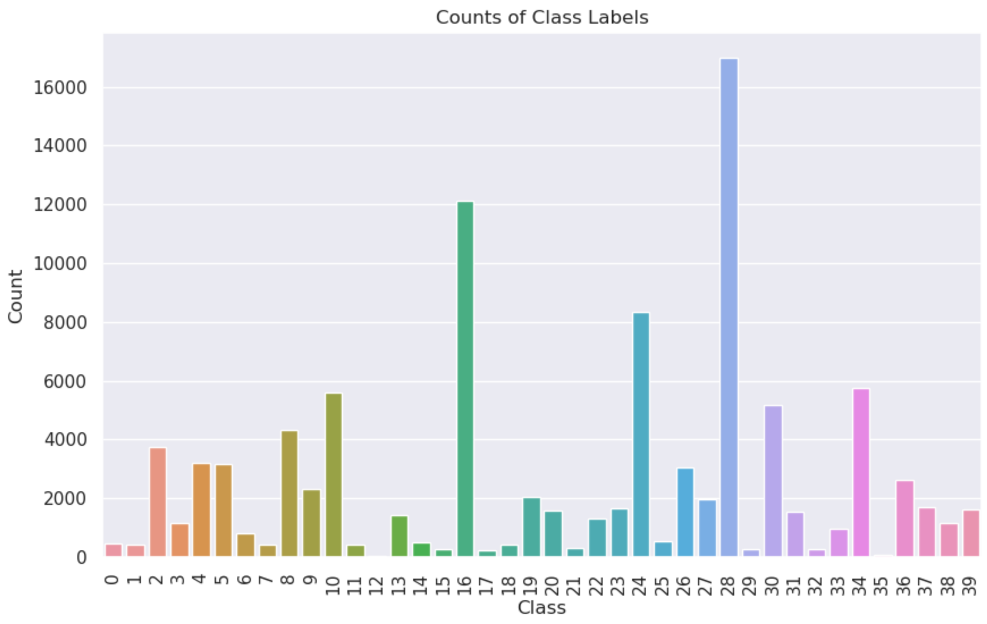


Figure 2: Counts of the class labels in the data

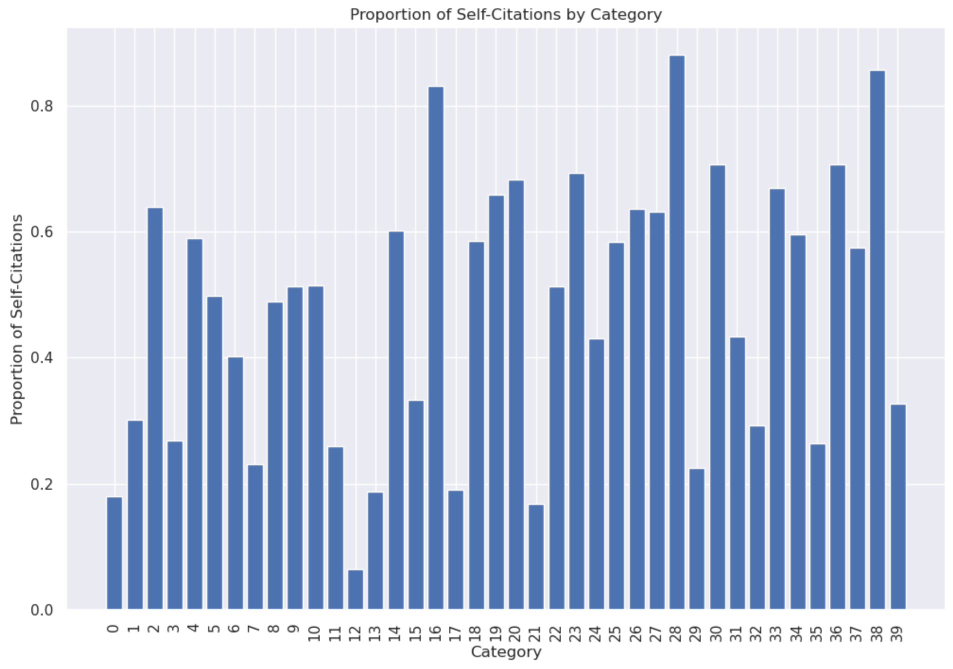


Figure 3: The proportion of citations to articles coming from the same class as the cited article

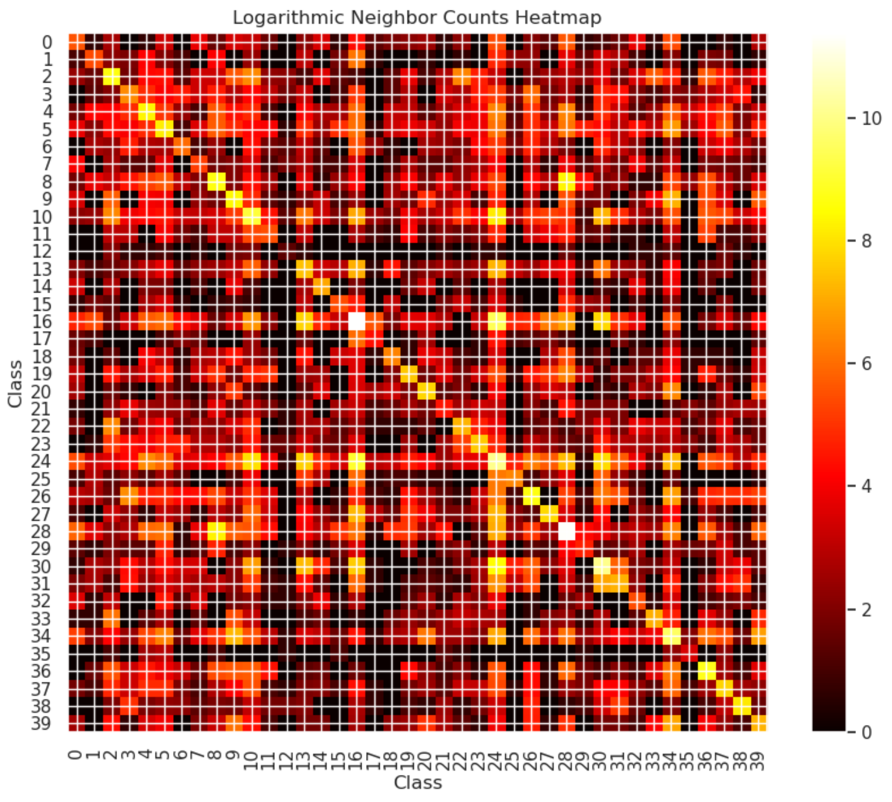


Figure 4: Heatmap of the citations between research fields

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.3500	0.2360	0.2819	89
1	0.5263	0.2500	0.3390	80
2	0.5562	0.6570	0.6024	618
3	0.3548	0.2200	0.2716	250
4	0.5834	0.6285	0.6052	673
5	0.4975	0.5185	0.5078	567
6	0.3462	0.2356	0.2804	191
7	0.5357	0.1899	0.2804	79
8	0.5388	0.6055	0.5702	768
9	0.5132	0.4146	0.4587	328
10	0.4403	0.4045	0.4217	1058
11	0.2609	0.0674	0.1071	89
12	0.0000	0.0000	0.0000	5
13	0.3801	0.3182	0.3464	264
14	0.4426	0.5000	0.4696	54
15	0.4118	0.3889	0.4000	54
16	0.7683	0.8054	0.7864	3355
17	0.2222	0.0465	0.0769	43
18	0.5652	0.3980	0.4671	98
19	0.5528	0.5366	0.5446	410
20	0.5598	0.5369	0.5481	244
21	0.0000	0.0000	0.0000	53
22	0.4759	0.3255	0.3866	212
23	0.5359	0.6276	0.5781	333
24	0.5442	0.5492	0.5467	2176
25	0.6106	0.4631	0.5267	149
26	0.5148	0.6289	0.5661	582
27	0.5658	0.5599	0.5628	484
28	0.7867	0.8449	0.8148	2663
29	0.5000	0.0455	0.0833	44
30	0.6906	0.7502	0.7192	1333
31	0.4619	0.3079	0.3695	354
32	0.5833	0.2333	0.3333	60
33	0.4912	0.4242	0.4553	132
34	0.5521	0.6426	0.5939	957
35	0.2500	0.0833	0.1250	12
36	0.6199	0.6674	0.6428	430
37	0.5494	0.4571	0.4990	280
38	0.6488	0.7112	0.6786	187
39	0.4153	0.2025	0.2722	242
Accuracy			0.6181	20000
Macro Avg	0.4801	0.4121	0.4280	20000
Weighted Avg	0.6057	0.6181	0.6078	20000

Table 2: Classification Report