Deep Learning Assignment
2AMM10
Technische Universiteit Eindhoven

Panagiotis Banos (1622773)

1 Task 1-Graph-based article retrieval

1.1 Wiki-CS

Wiki-CS is a dataset that has been constructed using data from several thousand Wikipedia articles related to Computer Science. This dataset is a graph representation of articles(nodes) and the hyperlinks(edges) that connect them to other nodes. Each article belongs to one of 10 different classes, representing 10 areas of Computer Science like Computer Security, Operating Systems and more.

1.1.1 Dataset Properties

The dataset contains 10701 nodes and 251927 edges. Every node has a 300 feature embedding that is the weighted average of the most important words in each article. The average outdegree of the nodes is 23.54. Moreover only 522 nodes are actually labeled for training and 5348 are labeled for testing.

1.2 Problem formulation and proposed approach

We are given a graph $\mathbb{G}$ along with an edge index $\mathbb{E}$ with $<t,s>$ pairs of target and source nodes $\forall\ \mathbb{V}\ \in \mathbb{G}$. As we also mentioned previously each node has an embedding e which contains all the information we need in order to train a model. Whilst we could very well do this task using an MLP based architecture, in our case we will take advantage of the graph structure and build an neural network that takes the complex relations between nodes into account.

There are many GNN architectures that have been created to tackle this problem. We have decided to build an attention based network as suggested by Veličković et al[1]. The thought behind this design choice is that a Wikipedia article has several other articles as neighbours of $1,2..n$ degree, because they share a similar topic/context. These neighbours(u) are connected to the article(v) via the hyperlinks in the body of the article. The further away a node u is from a starting node v, the smaller is the probability that these two nodes are associated. But we cannot say with a very high certainty that each and every hyperlink inside a wikipedia article is actually connected with the original because of the somewhat unregulated way that hyperlinks are inserted in each article. Thus we must pay higher attention to some of the neighbours than others.

As the only information we have about nodes are the characteristic words in each node's embedding, we can intuitively think of similar neighbours those that share similar characteristic words. Focusing more on these and fading the others we hope to train a model that generalizes well and understands the semantic context of the articles.

In traditional GNNs like GCN and GraphSAGE, the embedding of each node $e_v^{(l)}$ at a layer l is the result of some kind of aggregation, concatenation,pooling,LSTM(last two introduced from GraphSAGE) from the messages of all its neighbours plus the node's own pre-existing information from a previous update in layer l-1. Let's consider a simple example. The embedding can be expressed mathematically as following:

$$e_v^{(l)} = \sigma(\sum_{u\in N(v)} W^{(l)} \frac{e_u^{(l-1)}}{|N(v)|}) \tag{1}$$

where σ is an activation function(typically a ReLU or a sigmoid) and Σ denotes the aggregation of the messages of all neighbours u, multiplied by a weight matrix $W^{(l)}$ and normalized by the indegree of node v. To this the pre-existing information of v is concatenated thus giving us the final value of the embedding:

$$e_v^{(l)} = \sigma(CONCAT(e_v^{(l-1)}, (e \sum_{u\in N(v)} W^{(l)} \frac{e_u^{(l-1)}}{|N(v)|})) \tag{2}$$

In GAT-style networks we are learning an attention coefficient for every pair of neighbour nodes v and u, such that we are not weighting the message from each node the same(by normalizing with the indegree) but we value some of them more, or contrary less. The attention coefficients enable us to focus on local relationships calculating the coefficient only for the immediate neighbourhood N for every $u \in N_v$ The importance of a message from node u to v is calculated as such:

$$i_{v,u}^{(l)} = \alpha(W^{(l)}e_u^{(l-1)}, W^{(l)}e_v^{(l-1)}) \tag{3}$$

with α being the attention mechanism, which according to Veličković et al[1] implementation is a single-layer feedforward neural network. Once the message importances are learnt, the attention coefficient is normalized for all immediate neighbours k of node v:

$$a_{v,u} = \frac{exp(i_{v,u})}{\sum_{k \in N(v)} \exp(i_{v,k})} \tag{4}$$

Finally the embedding that is learned using the importances of the neighbours is:

$$e_v^{(l)} = \sigma(\sum_{u \in N(v)} a_{v,u} W^{(l)} e_u^{(l-1)}) \tag{5}$$

1.2.1 Article retrieval

At this point, we make the assumption that, if a model has learned some parameters that allow it to classify some input x correctly for a set of labels $y \in [1,2..10]$, then it has also learned to create embeddings at the final layers before the softmax function, that correctly compress and represent the information of various kinds of inputs x. Therefore, for the article retrieval task, we constructed a neural network and verified its abilities by performing a node classification task and then used the embeddings from the final layer to perform a kNN-type of retrieval. This works as follows:

Given a query article $q(x)$:

- Extract the candidate embedding of every node v from dataset $\mathbb{D}$ such that embedding(v_i) = e_i^d with d dimensions
- Extract the embedding of the query article s.t embedding($q(x)$) = e_q^d.
- Using cosine similarity, calculate the distance in the embedding space for every pair of query article and candidate: $\cos(e_q^d, e_i^d)\ \forall i \in \mathbb{D}$
- Insert the results in a sorted list
- Keep only the top-k most similar embeddings

The architecture of the model we built is very simple. We propagate through the network both the node embeddings and the adjacency data so that we make use of the structured nature of the graph. After two GAT layers which we consider to be enough to learn the neighbourhood relationships with respect to the node adjacencies, we pass the embeddings to a fully connected layer so that the network can get a higher level of understanding as shown by Babenko et al[2] in Neural codes for image retrieval. After that we forward the embedding into a softmax activation function and finally using CrossEntropy Loss we calculate the loss compared to the input. Since we only have a limited number of labeled nodes we can only calculate the loss on them. All in all, we have built a model that has learned some parameters $\hat{\theta}$ s.t. the loss is minimized:

$$\hat{\theta} = argmin - \sum_{i=1}^{10} \log p(x_i|\theta) \tag{6}$$

1.2.2 Network architecture

As it can be seen in Figure 1, the network contains two Graph Attention layers, with each one of them followed by a module of Batch Normalization to stabilize it, a LeakyReLU that adds expressiveness to the model and a Dropout with a rather high rate of 0.7 to combat over-fitting which is quite intense due to low number of training nodes. Finally a fully connected layer compresses the information.

Since GAT layers can sometimes be quite unstable, we are using multi-headed attention for GAT Layer 1. More specifically, we create multiple attention scores and train with them at the same time. At the end we aggregate them with a concatenation operation.

$$e_v^{(l)}[1] = \sigma(\sum_{u \in N(v)} a_{v,u}^1 W^{(l)} e_u^{(l-1)}) \tag{7}$$

$$e_v^{(l)}[2] = \sigma(\sum_{u \in N(v)} a_{v,u}^2 W^{(l)} e_u^{(l-1)}) \tag{8}$$

$$e_v^{(l)}[3] = \sigma(\sum_{u \in N(v)} a_{v,u}^3 W^{(l)} e_u^{(l-1)}) \tag{9}$$

$$e_v^{(l)} = AGG(e_v^{(l)}[1], e_v^{(l)}[2], e_v^{(l)}[3]) \tag{10}$$

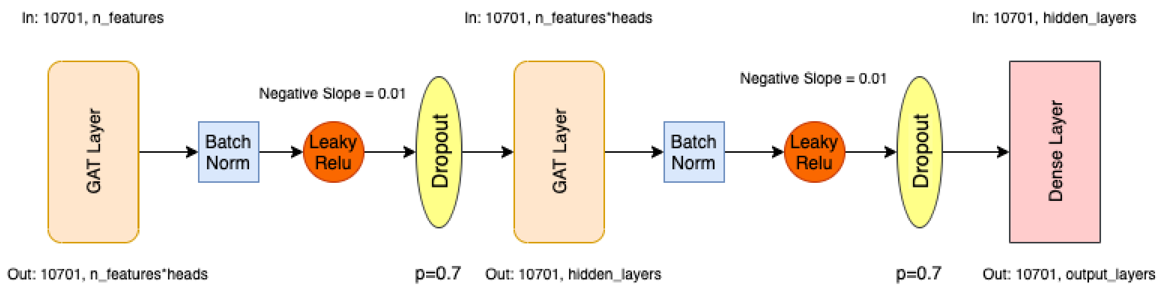


Figure 1: Model architecture for task 1

1.3 Results

First, lets see how the model performs for the node classification task. We trained for

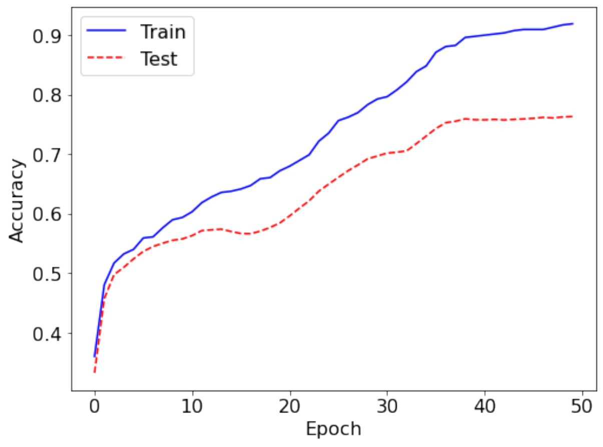


Figure 2: Training and test scores for 50 epochs, with 3 attention heads

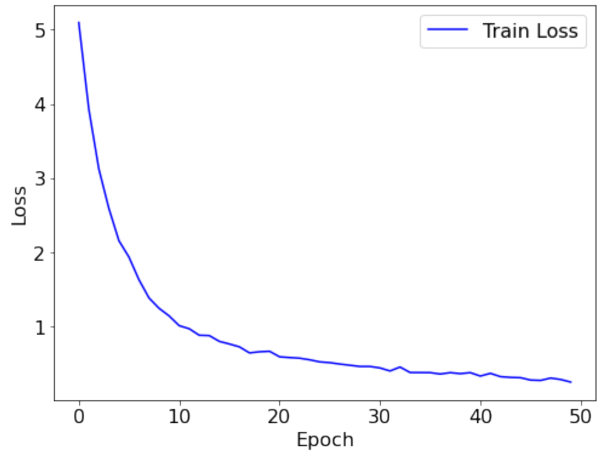


Figure 3: Training loss for 50 epochs, 3 attention heads

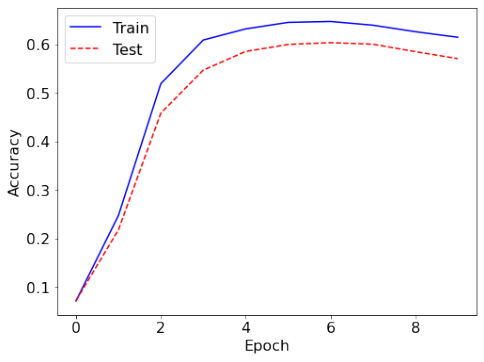


Figure 4: Training and test scores for 10 epochs,3 attention heads

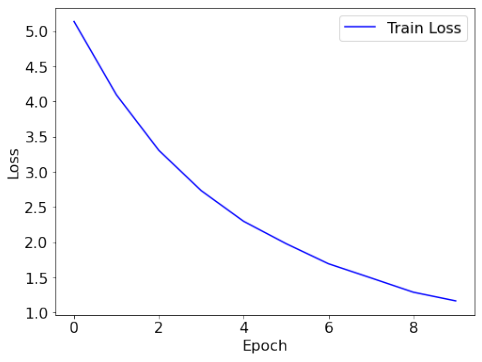


Figure 5: Training loss for 10 epochs, 3 attention heads

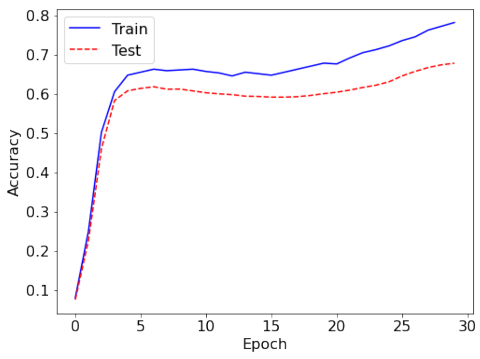


Figure 6: Training and test scores for 30 epochs,3 attention heads

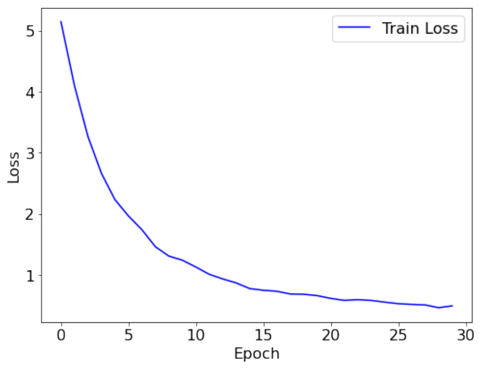


Figure 7: Training loss for 30 epochs, 3 attention heads

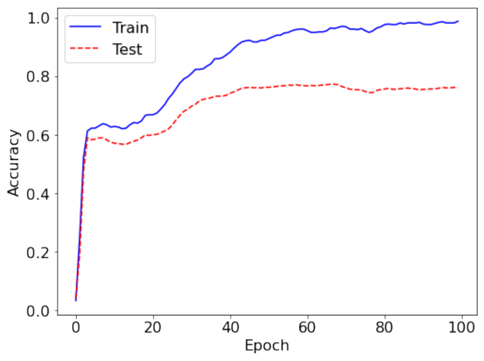


Figure 8: Training and test scores for 100 epochs,3 attention heads

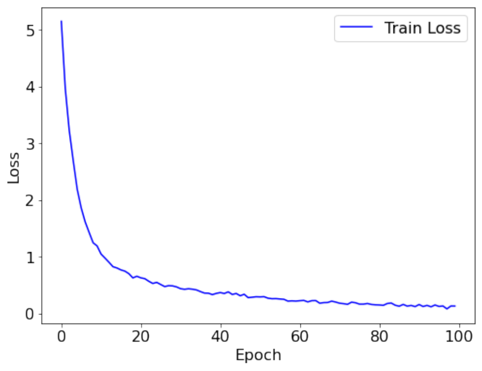


Figure 9: Training loss for 100 epochs, 3 attention heads

Training Accuracy	Test Accuracy	Loss	Heads	Epochs
61.4%	57.7%	116.7%	3	10
78.1%	67.8%	49.6%	3	30
93.6%	76.9%	27.1%	3	50
97.7%	75.9%	16.4%	3	100
90.8%	76.5%	39.2%	2	50
89.4%	76.7%	46.2%	1	50

Table 1: Accuracy and loss for various configurations of Heads and Epochs

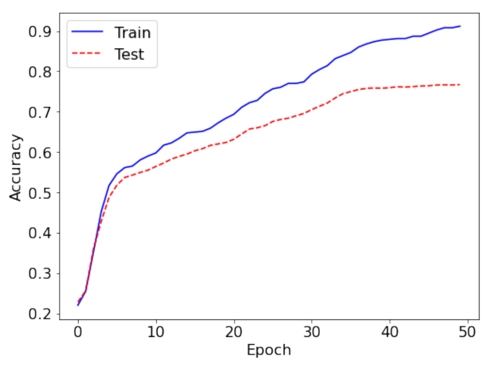


Figure 10: Training and test scores for 50 epochs,2 attention heads

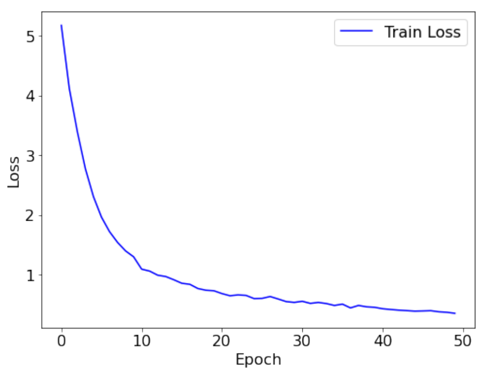


Figure 11: Training loss for 50 epochs, 2 attention heads

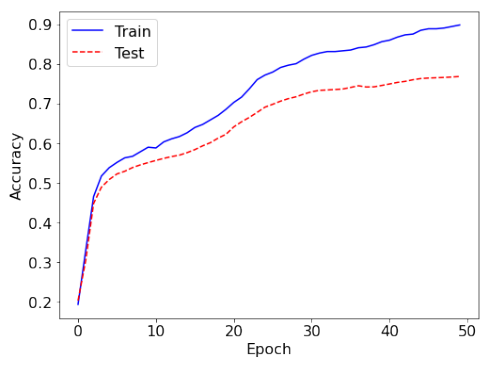


Figure 12: Training and test scores for 50 epochs,1 attention head

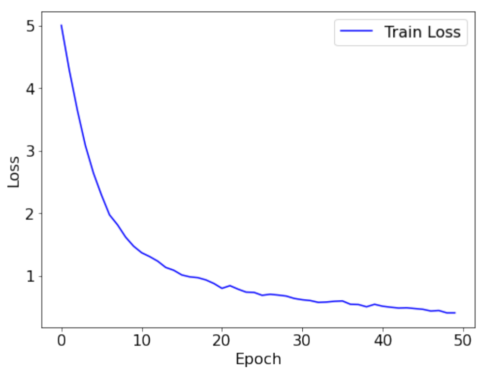
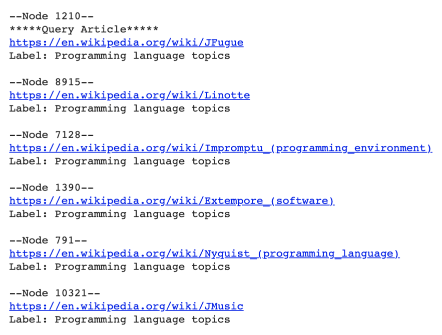


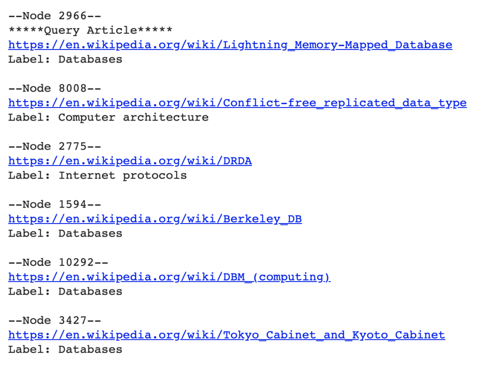
Figure 13: Training loss for 50 epochs, 1 attention head

Now lets see some example article retrievals:



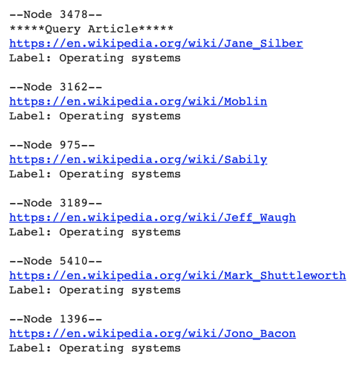
```
--Node 1210--
*****Query Article*****
https://en.wikipedia.org/wiki/JFugue
Label: Programming language topics

--Node 8915--
https://en.wikipedia.org/wiki/Linotte
Label: Programming language topics

--Node 7128--
https://en.wikipedia.org/wiki/Impromptu_(programming_environment)
Label: Programming language topics

--Node 1390--
https://en.wikipedia.org/wiki/Extempore_(software)
Label: Programming language topics

--Node 791--
https://en.wikipedia.org/wiki/Nyquist_(programming_language)
Label: Programming language topics

--Node 10321--
https://en.wikipedia.org/wiki/JMusic
Label: Programming language topics
```

Figure 14: Query about a java programming music library



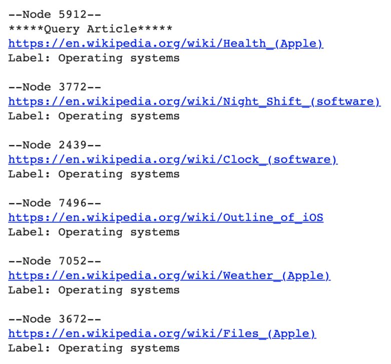
```
--Node 2966--
*****Query Article*****
https://en.wikipedia.org/wiki/Lightning_Memory-Mapped_Database
Label: Databases

--Node 8008--
https://en.wikipedia.org/wiki/Conflict-free_replicated_data_type
Label: Computer architecture

--Node 2775--
https://en.wikipedia.org/wiki/DRDA
Label: Internet protocols

--Node 1594--
https://en.wikipedia.org/wiki/Berkeley_DB
Label: Databases

--Node 10292--
https://en.wikipedia.org/wiki/DBM_(computing)
Label: Databases

--Node 3427--
https://en.wikipedia.org/wiki/Tokyo_Cabinet_and_Kyoto_Cabinet
Label: Databases
```

Figure 15: Query about a high-performance embedded transactional database that uses a key-value storage



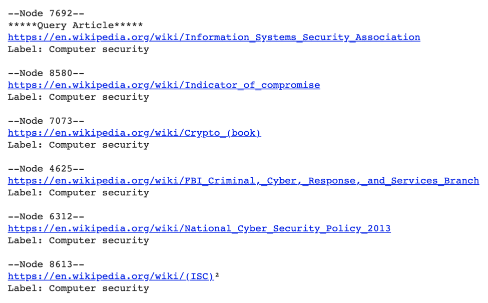
```
--Node 3478--
*****Query Article*****
https://en.wikipedia.org/wiki/Jane_Silber
Label: Operating systems

--Node 3162--
https://en.wikipedia.org/wiki/Moblin
Label: Operating systems

--Node 975--
https://en.wikipedia.org/wiki/Sabily
Label: Operating systems

--Node 3189--
https://en.wikipedia.org/wiki/Jeff_Waugh
Label: Operating systems

--Node 5410--
https://en.wikipedia.org/wiki/Mark_Shuttleworth
Label: Operating systems

--Node 1396--
https://en.wikipedia.org/wiki/Jono_Bacon
Label: Operating systems
```

Figure 16: Query about Jane Silber, a former CEO of Canonical



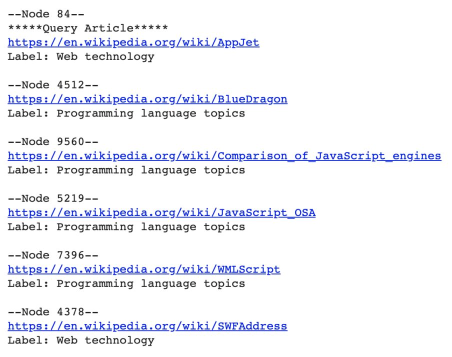
```
--Node 5912--
*****Query Article*****
https://en.wikipedia.org/wiki/Health_(Apple)
Label: Operating systems

--Node 3772--
https://en.wikipedia.org/wiki/Night_Shift_(software)
Label: Operating systems

--Node 2439--
https://en.wikipedia.org/wiki/Clock_(software)
Label: Operating systems

--Node 7496--
https://en.wikipedia.org/wiki/Outline_of_iOS
Label: Operating systems

--Node 7052--
https://en.wikipedia.org/wiki/Weather_(Apple)
Label: Operating systems

--Node 3672--
https://en.wikipedia.org/wiki/Files_(Apple)
Label: Operating systems
```

Figure 17: Query about an iOS health app



```
--Node 7692--
*****Query Article*****
https://en.wikipedia.org/wiki/Information_Systems_Security_Association
Label: Computer security

--Node 8580--
https://en.wikipedia.org/wiki/Indicator_of_compromise
Label: Computer security

--Node 7073--
https://en.wikipedia.org/wiki/Crypto_(book)
Label: Computer security

--Node 4625--
https://en.wikipedia.org/wiki/FBI_Criminal,_Cyber,_Response,_and_Services_Branch
Label: Computer security

--Node 6312--
https://en.wikipedia.org/wiki/National_Cyber_Security_Policy_2013
Label: Computer security

--Node 8613--
https://en.wikipedia.org/wiki/(ISC)²
Label: Computer security
```

Figure 18: Query about ISSA, the international professional organization of information security professionals and practitioners.



```
--Node 84--
*****Query Article*****
https://en.wikipedia.org/wiki/AppJet
Label: Web technology

--Node 4512--
https://en.wikipedia.org/wiki/BlueDragon
Label: Programming language topics

--Node 9560--
https://en.wikipedia.org/wiki/Comparison_of_JavaScript_engines
Label: Programming language topics

--Node 5219--
https://en.wikipedia.org/wiki/JavaScript_OSA
Label: Programming language topics

--Node 7396--
https://en.wikipedia.org/wiki/WMLScript
Label: Programming language topics

--Node 4378--
https://en.wikipedia.org/wiki/SWFAddress
Label: Web technology
```

Figure 19: Query about AppJet, a website that let users create web-based applications in a client web browser.

1.4 Analysis and conclusions

Starting with the classification task performance(Table 1), it is clear that an upper bound of test accuracy is hit with the configurations presented. Very similar upper boundaries are found even when testing with other architectures such as GraphSAGE and GCNs, though due to time and space constraints we are not presenting them in this report.

With respect to the combinations of heads and epochs tested, we understand that the model overfits strongly as the number of epochs increases over 50. Increasing the number of heads gives us minimal gains. We opted to have 3 heads in our default configuration, due to the amount of loss we observed. It should also be noted that there is a direct relationship with the time required to train the network and the number of heads, but due to the small size of our graph this is hardly noticeable.

Moving on to the article retrieval results, we have presented some sample query answers in Figures 14 to 19. What is apparent in these, is that the model manages to understand the underlying context of each query, at most occasions, and retrieve similar articles. For example, in Figure 16, we have a query about a former CEO of Canonical, the company behind Ubuntu, a popular Linux distribution. The article retrieval returned to us several other people that were involved with Ubuntu in the past and two other Ubuntu based Linux distribution. Similarly, we can also see that the model understands the complex relationships of each article by looking at Figure 19 and Figure 14. The former is a query about a AppJet a website that let users write javascript script in a web client. Seeing the results we notice that only one of them has the same label as the query article, but all of them contain information related to Javascript. Looking at Figure 14, the retrieval for a query about a java music programming plugin are other programming languages and frameworks dedicated to music programming.

1.5 Implementation

The implementation can be found in **gnn-and-vae.ipynb**

References

[1] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P.,& Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.

[2] Babenko, A., Slesarev, A., Chigorin, A., & Lempitsky, V. (2014, September). Neural codes for image retrieval. In European conference on computer vision (pp. 584-599). Springer, Cham.