ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

REPORT

NETWORK MACHINE LEARNING

Bohan Wang, Nearchos Potamitis

9th June 2022

Contents

CHAPTER 1

INTRODUCTION

In this report we will present our study into the fictional communities created by the Marvel and DC comics created during the 2000-2002 time period. We will start by focusing on exploring and analyzing the data with an ultimate goal of creating different graphs and using them for different classification tasks.

To begin with, we will examine the data in their current form but we will also create graphs associating different characters based on their common attributes. In particular, we will deal with common affiliations, relatives and appearances in the same comics. We will inspect different aspects of these graphs such as their global properties, the type of the graph and the properties of the nodes. To be more specific we will look into things like connected components, sparsity, diameter but also clustering coefficients and degree distribution. Additionally, we will use spectral graph theory to review the attributes of the graph. The result of this analysis will help us extract the appropriate features for each node which will be used in the next part of our study.

For the later part of this report, we will concentrate on two different classification tasks performed on the created networks. Firstly, we will try to distinguish good, bad and evil characters and then apply the same principle over different affiliations instead. For example, for the character classification we will label Spiderman as a good character and then for the affiliation classification we will label the Avengers as a good unit. To make this happen we will use different classical machine learning algorithms, regularization methods and finally we will define our own architecture for a graph convolutional neural network.

> *“I don’t have inspiration. I only have ideas. Ideas and deadlines.”*
>
> Stan Lee
> Primary creative leader of Marvel Comics

CHAPTER 2

METHODS

2.1 DATA ACQUISITION

Our data consist of Marvel and DC characters' information from 2000 to 2001. The information is scraped from the [DC Comics Database] `https://dc.fandom.com/wiki/DC_Comics_Database`) and the [Marvel Database](`https://marvel.fandom.com/wiki/Marvel_Database`) at [Fandom](`https://www.fandom.com/`) by William Cappelletti . There are totally four datasets, each of which covers one year of comics, with all appearing characters and the story they appeared in. Characters are labeled as good, neutral or evil.

The dataset columns are presented in the following table:

URL	Real Name	Current Alias	Relatives	Affiliation	Comics	Good	nb_appearences

TABLE 2.1
Names of dataset columns

There are four important attributes with which we are going to concern ourselves during this report. The **Relative, Affiliation** and **Comics** columns in Table 2.1 are three types of interactions or relations between two characters. First, each character's Relative feature is the list of known relatives, such as parents or spouses. Second, the Affiliation feature is the list of groups the character has ever been part of (not restricted to the dataframe span). Thirdly, Comics feature contains the list of comics titles the character has appeared in. Finally, **Good** feature is an integer value representing whether the character is good (+1), neutral (0), or evil (-1). We will use `Good` feature as the labels for the classification task.

2.2 DATA PRE-PROCESSING

After loading the data, there are totally 1787 rows in the Marvel 2000 dataset, 1672 rows in the Marvel 2001 dataset, 1018 rows in the DC 2000 dataset and 910 rows in the DC 2001 dataset. Each row in the four datasets consists of a character's information. We will refer to the characters as the nodes of our graphs, so we assign the node id for each row.

We notice there are no missing values in any dataset. However, in the Relatives and Affiliation features/columns, there were instead some empty lists, which means some characters are not associated with other characters. In addition, there are not any empty lists in the Comics features, because each character in the datasets has appeared in at least one comic. In Table 2.2, we present the times there is an empty

list in either of these features. We just leave the empty lists, because they are useful during the network creation process.

Dataset	Times of empty lists		
	Relatives	Affiliation	Comics
Marvel 2000	947	476	0
Marvel 2001	852	447	0
DC 2000	531	295	0
DC 2001	460	253	0

TABLE 2.2
Number of empty lists in Relatives, Affiliation and Comics features/columns

2.3 EXPLORATION

2.3.1 NETWORK CREATION

Network creation is the most important part to analyze graph structured data. After observing our datasets, we decided to create undirected multigraph for each dataset. Multigraph consists of a set of nodes, and a collection of relations that specify how pairs of nodes are related to each other. Social network and collaboration networks are the typical multigraphs. We plan to use multigraph to fully understand the patterning and intertwining of relations between two characters in the Marvel and DC universes.

For each dataset, we create multigraphs based on different relations including Relatives, Affiliation and Comics interactions. Thus we have three multigraphs for one dataset. The process of creating multigraph is described as following. First, we get the corresponding relations of each character. Second, we find the common relations which two characters have. The common relation means there is an edge between the two characters/nodes. Finally, we count the number of edges between two characters. The total number of the edges will be the edge weights between two nodes in the adjacency matrix.

After creating the adjacency matrices of the multigraphs, we notice the edge weights in Comics adjacency matrix are much larger than the other two adjacency matrices, because different characters always appear in the same comic. Therefore, we normalize each adjacency matrix to range $[1, 2]$. This can keep the same importance of different adjacency matrices. We firstly use Min-max normalization. Then, we add 1 to the normalized adjacency matrix to make sure the smallest edge weight in original adjacency matrix is at least 1 in the normalized matrices. The formula of this process is as following:

$$A_{\text{norm}} = \frac{A - A_{\min}}{A_{\max} - A_{\min}} + 1 \tag{2.1}$$

where A_{norm} is the normalized adjacency matrix, A is the original adjacency matrix, $A_{\min}$ is the smallest value in A and $A_{\max}$ is the largest value in A.

Finally, for each dataset, we add its three matrices to get the final adjacency matrix for a dataset. We set the weights of Relatives adjacency matrix as 0.5 and Comics adjacency matrix as 0.1, because the Affiliation adjacency matrix can provide the most meaningful information. Good characters always have common affiliations. Also, bad characters have common affiliations. Relatives adjacency matrix can provide meaningful information about the relationships between characters. Comics provide the least information to determine whether a character is good, neutral or bad. The formula of this process is as following:

$$A_{\text{all}} = A_{\text{affiliation}} + 0.5 * A_{\text{relatives}} + 0.1 * A_{\text{comics}} \tag{2.2}$$

2.3.2 VISUALIZATION

In order to visualise our graphs we downloaded the corresponding gexf files and imported them into Gephi, an open network visualization and exploration software. From there, we set the URL of each character as the label of the node which we then coloured blue for good, black for neutral and red for evil characters. The size of each node is proportional to its degree and its colour is according to the community that it was clustered to by the Louvain community detection algorithm (modularity). Finally, we used a force-directed layout (Force Atlas 2) to spatialize the graph and display the different communities in a more concise way. We show an example of our stationary graphs in Fig 2.1 while we include the rest of such graphs in the appendix (Fig 4.1). In case you are interested you can check the interactive version of the same graphs in the following links : - Marvel 2000, - Marvel 2001, - DC 2000, - DC 2001.

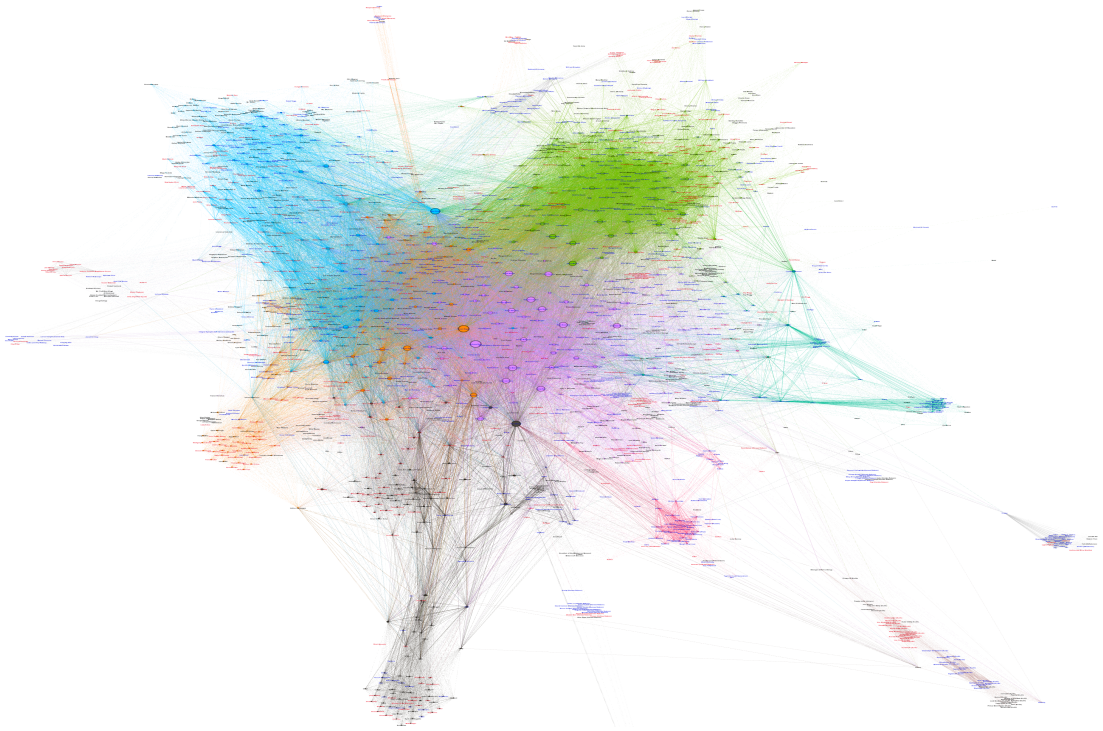


FIGURE 2.1
Network visualization for Marvel 2000 dataset

2.3.3 PROPERTIES OF NETWORKS

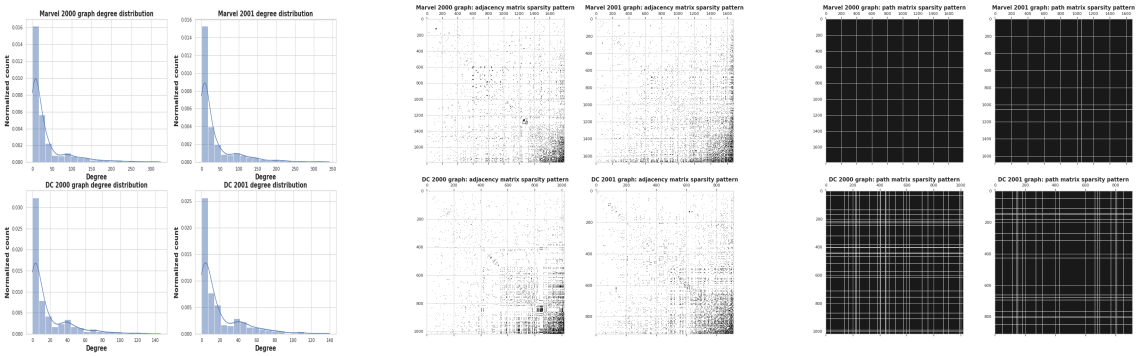
GRAPHS PROPERTIES

First of all, we explored the graph properties including number of nodes, number of edges, degree distribution, sparsity, connected components and giant components. These graph properties can reveal the type of the graphs and give us insights about the latent structures behind the network. The global properties are presented in Table 2.3. We also indicate the degree distributions and sparsity pattern of the adjacency matrices in Fig 2.2.

As shown in Table 2.3, for each network, the giant component is composed of most nodes (above 90%) in the network, which is also indicated by the total edge weights of the giant component. In addition, the

Networks	#nodes	#edges	#connected components	First moment $\langle k \rangle$	Second moment $\langle k^2 \rangle$	Total degree	Giant component		
							#nodes	#edges	diameter
Marvel 2000	1787	30011.8	3	33.59	3674.66	60023.6	1784	30010.7	6
Marvel 2001	1672	30850.7	2	36.90	4269.34	61701.4	1671	30850.7	5
DC 2000	1018	8592.3	19	16.88	840.62	17184.6	964	8535.0	8
DC 2001	910	8504.2	14	18.69	936.82	17008.4	874	8468.7	6

TABLE 2.3
Global properties of networks. Note #nodes refers to the total number of nodes. #edges represents the total edge weights. #connected components is the total number of connected components.



(A) Degree distributions **(B)** Sparsity patterns **(C)** Path matrix $(N = 10)$

FIGURE 2.2
Degree distributions and sparsity patterns of different networks

diameters of the giant components in different networks are a little large, so no characters can interact with most of other characters. We notice that $\langle k^2 \rangle$ is much larger compared to $\langle k \rangle$ revealing the scale-free behaviour of networks. By comparing the Marvel and DC networks, DC network consists of fewer nodes, fewer edges weights but much more connected components than Marvel networks. Therefore, DC universe does not connect all characters. DC universe probably would like to make stories about some characters not all characters. From Fig. 2.2 (A), the degree distributions of all networks are similar to fat tailed distributions as they follow the power law $P(k) \sim k^{-\gamma}$. Therefore, the networks are possibly scale-free networks. In Fig. 2.2 (B), we can see the networks are very sparse. Not all characters are connected. Also, Fig. 2.2 (C) indicates that not all characters interact in 10 steps path.

As summary, the networks of Marvel and DC are similar to scale-free networks. There are hubs (popular/important characters) in the networks. Also, the networks are very sparse. The giant components follow the small-world phenomenon, but hubs do not affect the small-world property significantly. Therefore, the networks' hubs are not connected to each other.

NODES PROPERTIES

The node properties such as clustering coefficient and betweenness centrality were investigated to reveal the structure of the networks further and help us design handcrafted features. To verify our network creation is meaningful, we visulaize the top 10 important nodes based on Degree and Betweenness Centrality. The details of the two global properties are shown in the following:

Degree: In this project, this is simply the normalized number of connections the node has in the network. In the Marvel and DC universe, this corresponds to the total number of common affiliations, relatives and comics which the two characters have. For example, imagine if you are Iron man, you will be very important because you will have a lot of common affiliations and relatives with other characters. You also

Graphs	Avg clustering coeffiecients
Marvel 2000	0.758
Marvel 2001	0.747
DC 2000	0.753
DC 2001	0.732

TABLE 2.4
Average clustering coefficients of different networks

appear in a lot of comics.

Betweenness Centrality: this corresponds to how many shortest paths in the network lead through the node. For example, imagine you are Iron man and you want to send a message to Wolverine. The shortest path how to send it is via Spider man, because he interacted both with Iron man and Wolverine. On the other side, if you want to send a message to Captain America, you don't have to go through Spider man because Iron man knows Captain America directly. The betweenness centrality for Spider man is computed using the number of shortest paths between all other characters that pass through him.

From Fig. 2.3 and Fig. 2.4, we can see all important characters meet our expectations. For example, Iron man and Spider man are very popular in Marvel universe. Also, Superman and Batman are very popular in DC universe.

Finally, we present the average clustering coefficients in Table 2.4. The clustering coefficient is an indication about the connectivity pattern of a node to its neighbors and gives us a hint about the network model. Generally, it tells how well connected the neighborhood of the node is. The larger the clustering coefficient, the more the node's neighbors are connected. As shown in Table 2.4, the average average clustering coefficients of Marvel and DC are similar and relatively large, so the networks have more regular connectivity patterns. Also, they are as large as to show that the networks are not random networks.

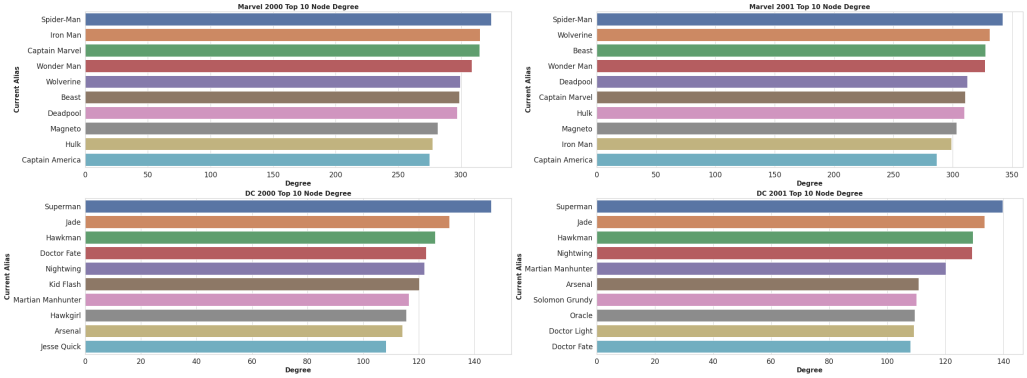


FIGURE 2.3
Top 10 degree important nodes

2.4 EXPLOITATION

2.4.1 FEATURE EXTRACTION

For feature extraction, we firstly consider the following five types of hand-crafted features :

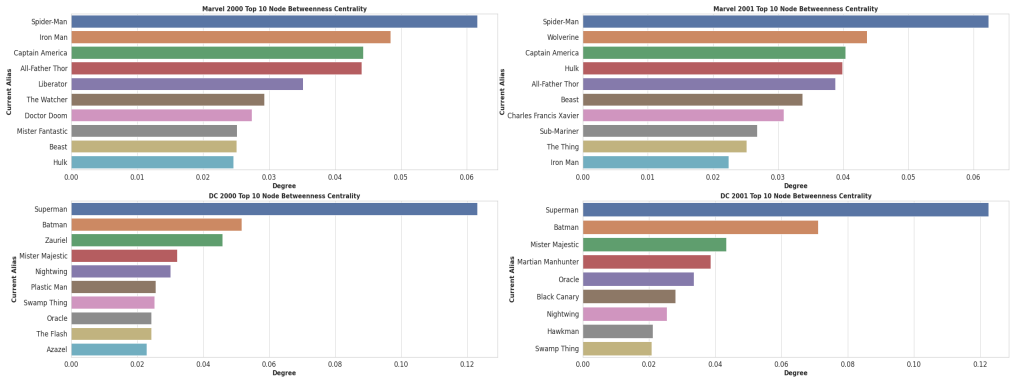


FIGURE 2.4
Top 10 betweenness centrality important nodes

- Degree
- Betweenness centrality
- Closeness centrality
- Eigenvector centrality
- Clustering coefficients

To begin with, one of the reasons we chose these features is that from Fig. 2.3 and Fig. 2.4 node degree and betweenness centrality provide meaningful information about the important nodes. Moreover, we consider closeness centrality and eigenvector centrality as they measure the importance of a node based on the amount of shortest path lengths to all other nodes and how many important nodes surround it. Finally, we use clustering coefficients to provide information about the subgraph containing the neighbors of a node, and all edges between nodes in its neighborhood.

In order to better prepare for the classification tasks we will also compute a thirty dimensional feature vector for each node. To achieve that we use Node2vec with $p = 1.0$ and $q = 10.0$ aiming to capture structural nodes e.g. hubs, outliers. The reason we selected such parameters is the fact that for social/behavioural networks the communities of nodes usually are not very clear.

2.4.2 FEATURE ENGINEERING

Since evaluating our models on an independent set is essential for classification tasks we first need to split our dataset in a training set (used for training) and a test set (used just to assess our the performance of our model). In our case, we will perform a node-level classification and therefore each node is a sample. So, to obtain our splits we use randomly selected, non-overlapping masks. A particular point where we had to pay attention at this point is the fact that the good/bad/neutral labels of our dataset are not distributed equally (shown in Fig 4.3). For this reason, we had to make sure that the training set would have the correct proportion of each of these labels.

Finally, we have everything ready for training and only two minor modifications remain. The first one, is to standardize our features by removing the mean and scaling to unit variance. This has to happen as our initial data has varying scales but also because we will use models such as logistic regression that make assumptions about our data having a Gaussian distribution. The second modification concerns the labels

of the nodes. We want to use cross-entropy to train our graphs and therefore we add one to the integer values so the labels for good are equal to two, for neutral are equal to one and for bad are equal to zero.

2.4.3 GRAPH TIKHONOV REGULARIZATION

In this section, the extracted 35 graph signals of each dataset are analyzed further. We firstly calculate the smoothness of the 35 graph signals for each network. To calculate the smoothness of our features, we calculate the inverse of smoothness along the quadratic form of each feature. Then, we calculate the average of the smoothness of all features. The process is indicated as following:

$$\text{smoothness} = \sum_{i=1}^{i=N_{\text{features}}} \frac{x_i^T L x_i}{N_{\text{features}}} \tag{2.3}$$

where $N_{\text{features}} = 35$ (number of handcrafted features), x_i is the i_{th} graph signal and L is the symmetric normalized graph laplacian. The average inverse of smoothness of each graph is demonstrated in Tabel 2.5. As indicated in Tabel 2.5, the graph signals are not smooth enough. Therefore, we decided to use a filter to only preserve the components associated with the smallest eigenvalues. The Tikhonov filter can reduce the components associated with large eigenvalues, and preserve the low frequencies. We implement it to smooth the graph signals. Each graph signal $\mathbf{x}$ is filtered as

$$\mathbf{x}_{\text{smooth}} = \mathbf{U}\hat{g}(\mathbf{\Lambda})\mathbf{U}^\top\mathbf{x} = \hat{g}\left(\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top\right)\mathbf{x} = \hat{g}(\mathbf{L})\mathbf{x}$$

where $\hat{g}(\cdot)$ is the spectral filter performed in the graph Fourier domain as a function of the eigenvalues ("frequencies"). Therefore, we firstly obtain the graph Fourier transform of the signal ($\hat{\mathbf{x}} = \mathbf{U}^\top\mathbf{x}$). Then, we get the spectral response of ideal Tikhonov filter:

$$\hat{g}(\mathbf{\Lambda}) = \frac{1}{1+\alpha\lambda},$$

In the project, we set $\alpha = \frac{0.99}{\lambda_{max}}$. After filtering the signal on the spectral domain ($\hat{g}(\mathbf{\Lambda})\hat{\mathbf{x}}$), we recover the filtered signal in the vertex domain ($\mathbf{U}\hat{g}(\mathbf{\Lambda})\hat{\mathbf{x}}$). As shown in Table 2.5, the graph signals are smoother after implementing Tikhonov regularization.

Graphs	Avg inverse of smoothness	Avg inverse of smoothness after smoothing
Marvel 2000	20378.59	10034.15
Marvel 2001	23525.87	10494.47
DC 2000	2064.98	976.74
DC 2001	1965.48	1005.67

TABLE 2.5
Average inverse of smoothness of graphs and smoothed graphs

CHAPTER 3

CLASSIFICATION

So far we have introduced our dataset, explored its attributes, extracted the most important features and prepared our data for further processing. In this section we will take advantage of our earlier preparations and use them to perform certain classification tasks. We report the test performance in Table 3.1 and Table 3.2. The F1-score is presented, because our dataset is unbalanced (good characters are more than the others shown in Fig 4.3).

3.1 CHARACTER CLASSIFICATION

As we have already mentioned, each character is classified as good, bad or neutral. Our next step is to create a model that predicts this attribute of the character based on the already discussed features of their respective nodes.

As a first step, we have tried 3 classical machine learning algorithms : Linear support vector machine classification, Support vector machine with a radial basis function (RBF) kernel and Logistic regression. Using these methods in combination with Tikhonov regularization gave us an average 60% accuracy across all different datasets and methods. We noticed that even though Tikhonov regularization increases the accuracy in most cases (sometimes by even 2%) on four different occasions the accuracy decreased instead. The regularization parameters of all classifiers are obtained by 5-fold cross validation.

In an effort to improve our results even more we decided to use Pytorch Geometric and graph neural network (GNN) to solve this task. After numerous trials, we decide to use the graph convolutional operator: A series of the chebyshev spectral graph convolutional operators from the "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering" paper followed by a fully connected two layer neural network. In the project, we use five layers 7^{th} order Chebyshev polynomials expansion to approximate the filter which can capture meaningful information from the graph signals. Also, the embedding dimensions of the layers are $8, 16, 32, 64, 128$, so finally we can get 128 dimensional vector for each node. Each layer increases the receptive field of a node by 7 hops because we use 7^{th} order Chebyshev polynomials. Due to overfitting issues we also added a dropout layer with the probability of randomly dropping as 10% after each operator. A detailed figure of the whole network can be found in the appendix (Fig. 4.4). The results for each model are portrayed in Table 3.1.

As shown in Table 3.1, GNN and Tikhonov regularised SVM with RBF kernel and can achieve good test performance.

	Marvel 2000		Marvel 20001		DC 2000		DC 2001	
Model/Metric	Accuracy	F1	Accuracy	F1	Accuracy	F1	Accuracy	F1
Linear SVM	0.54	0.51	0.50	0.48	0.61	0.43	0.64	0.55
Linear SVM (Tikhonov)	0.54	0.50	0.50	0.48	0.62	0.43	0.64	0.56
SVM with RBF	0.65	0.63	0.59	0.57	0.66	0.59	0.68	0.64
SVM with RBF (Tikhonov)	**0.67**	**0.65**	0.59	0.58	0.67	0.61	0.67	0.64
Logistic Regression	0.54	0.50	0.50	0.48	0.60	0.42	0.64	0.55
Logistic Regression (Tikhonov)	0.55	0.52	0.49	0.47	0.59	0.42	0.65	0.58
GNN (ChebConv operator)	0.64	0.62	**0.62**	**0.61**	**0.70**	**0.65**	**0.72**	**0.69**

TABLE 3.1
Character classification results

3.2 AFFILIATION CLASSIFICATION

For the affiliation classification task, the very first challenge is to assign labels to the affiliations themselves. That means to label each affiliation as good, bad or neutral. To solve this problem we calculated the mean value of the node labels within the affiliation and based on this average we set the label of the affiliation as an integer value. The method is shown in equation 3.1. We followed this method as it seemed reasonable to us that an affiliation composed mostly by a certain type of characters means that the affiliation as a whole can be characterized by the same type too. The next and final step is to pool (average pooling) the node embedding of each affiliation. The node embedding is achieved by previous graph neural networks. Hence, we are ready to perform the classification on the affiliations dataset.

$$y = \begin{cases} -1 & \text{if } -1 \leq avg < -0.33, \\ 0 & \text{if } -0.33 \leq avg < 0.34, \\ 1 & \text{if } 0.34 \leq avg \geq 1, \end{cases} \tag{3.1}$$

where y is the label of an affiliation, avg is the average of the node labels within an affiliation.

For this task we only used the classical machine learning models as the results were already much better. The regularization parameters of the classifiers are also obtained by 5-fold cross validation. Across all models and datasets we averaged an 85% classification accuracy. Even though our results are better, this might be due to the fact that in each dataset there are about three times more good affiliations than bad or neutral ones.

The results for each model can be found in Table 3.2.

	Marvel 2000		Marvel 20001		DC 2000		DC 2001	
Model/Metric	Accuracy	F1	Accuracy	F1	Accuracy	F1	Accuracy	F1
Linear SVM	0.83	0.79	0.87	0.85	0.86	0.82	0.81	0.75
SVM with RBF	0.84	0.81	0.88	0.87	0.88	0.84	0.86	0.80
Logistic Regression	0.84	0.80	0.87	0.86	0.88	0.84	0.87	0.82

TABLE 3.2
Affiliation classification results

CHAPTER 4

APPENDIX

4.1 GRAPH VISUALIZATION

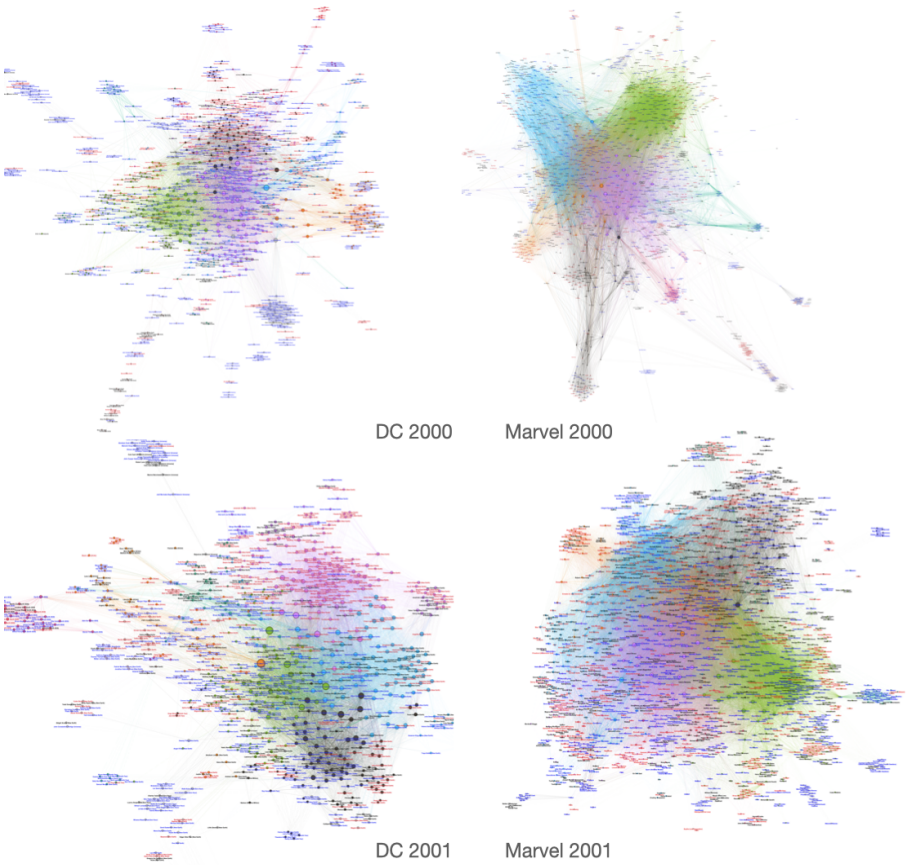


FIGURE 4.1
Final graphs visualization

4.2 FEATURE VISUALIZATION

At this moment we are dealing with high dimensional data (35 features) which means that we can't visualise them in their current form. To do that we will first have to apply some dimensionality reduction methods first. In this section we will use the following methods to visualise our data :

- Principal Component Analysis (PCA) : The principal components of the dataset are computed and then used to perform a change of basis in the data.
- t-Distributed Stochastic Neighbor Embedding (TSNE) : Statistical method for visualizing high-dimensional data by giving each datapoint a location in a two or three-dimensional map.
- Isomap : Computes a quasi-isometric, low-dimensional embedding of a set of high-dimensional data points.

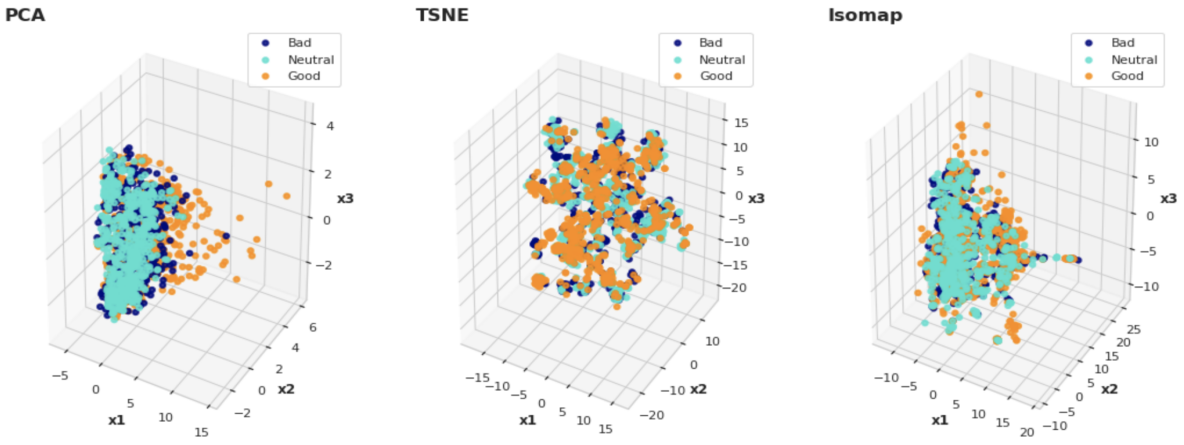


FIGURE 4.2
Feature visualization for Marvel 2001 dataset

4.3 LABEL DISTRIBUTION

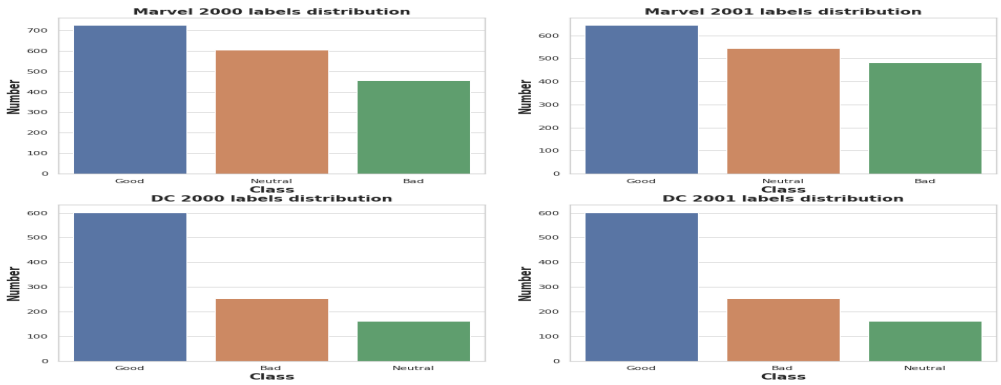


FIGURE 4.3
Nodes labels distributions

4.4 GNN ARCHITECTURE

```
NodeClassifier(
  (gnn_block): GNN(
    (gnn): ModuleList(
      (0): GNNBlock3(
        (conv1): ChebConv(35, 8, K=7, normalization=sym)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (1): GNNBlock3(
        (conv1): ChebConv(8, 16, K=7, normalization=sym)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (2): GNNBlock3(
        (conv1): ChebConv(16, 32, K=7, normalization=sym)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (3): GNNBlock3(
        (conv1): ChebConv(32, 64, K=7, normalization=sym)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (4): GNNBlock3(
        (conv1): ChebConv(64, 128, K=7, normalization=sym)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
  )
  (classifier): Sequential(
    (0): Linear(in_features=128, out_features=128, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.1, inplace=False)
    (3): Linear(in_features=128, out_features=3, bias=True)
    (4): Dropout(p=0.1, inplace=False)
  )
)
```

FIGURE 4.4
GNN architecture