DOI: https://doi.org/10.53052/9788367652124.05

Mikołaj GRYGIEL[1]

Opiekun naukowy: Stanisław ZAWIŚLAK[2]

SYSTEM DORADCZY WYBORU UTWORÓW MUZYCZNYCH Z ZASTOSOWANIEM GRAFOWYCH SIECI NEURONOWYCH

Streszczenie: W artykule omówiono własną aplikację będącą system doradczym wyboru utworów muzycznych. Dane o milionach utworów pobrano z zasobów sieciowych – z zakresu muzyki popularnej – jazz, pop, rap, country itd. Rady są generowane poprzez grafową sieć neuronową. Sieć zbudowano za pomocą narzędzi profesjonalnych typu open-access. Zastosowano uproszczone reguły uczenia uwzględniające tzw. osadzenia wierzchołków. Trenowano sieć wykorzystując różne wartości hiperparametrów, w celu uzyskania trafniejszych wyników Aplikacja dla podanego utworu podpowiada użytkownikowi 10 podobnych dzieł np. tych samych autorów, podobne zespół, podobny styl, ten sam okres powstania itp. Działanie porównano z klasycznym narzędziem SPOTIFY, uzyskano zadawalające podobieństwo wyników, przy czym wybrane sugestie się powtarzały. Przeprowadzono szeroką analizę wpływu parametrów na metryki ewaluacji. Rozważany problem jest z zakresu rozrywki, ale przetestowano zastosowanie i uczenie grafowej sieci neuronowej osiągając praktyczny, poprawnie działający system doradczy. Może być zastosowany do wszelkich problemów doradczych z zakresu big data. Można skomplikować metody uczenia poprzez modyfikację wag na krawędziach grafu.

Słowa kluczowe: osadzenia wierzchołków, zasady uczenia odnoszące się do sąsiedztwa, dane pobrane z zasobów sieciowych

ADVISORY SYSTEM RELATED TO MUSIC AND SONGS BASED ON GRAPH NEURONAL NETWORK

Summary: In the paper, own application is described. The software consists in issuing suggestions on music pieces and songs. The big data resources were downloaded from network sites. The considered styles are: jazz, pop, rap, country, big beat etc. The pieces of advice are generated by a graph neuronal network. This net was built utilizing the professional tools of open-access type. The simplified learning rules were applied i.e. only vertex embeddings had been modified in consecutive iterations. The graph neuronal network was trained using different hyperparameters configuration to get more accurate results. For a given entered song, the application gives an output i.e. a list of 10 similar songs e.g.: written by the same authors or similar band, of the same style or/and period of presentation etc. The results were compared

[1] MSc, University of Bielsko-Biala, Dept. of Informatics and Automation, mgrygiel@ubb.edu.pl

[2] PhD, D.Sc., University of Bielsko-Biala, Dept. of Informatics and Automation, szawislak@ubb.edu.pl

with a classic tool (SPOTIFY). Acceptable level of similarity was achieved, some results were identical. Wide analyses of parameter values impact on the evaluation metrics were performed. The considered problem is related to amusement (social life), however it was checked that application and training of a graph neuronal network is usable and effective. Moreover such a system can be used in any general advisory system related to big data. The training methods can be more complicated i.e. modification of edge weights can be inserted.

Keywords: vertex embeddings, neighborhood learning rules, www-based downloaded data

1. Introduction

Nowadays, the advisory systems are commonly used. The considered issue belongs to the area of Artificial Intelligence (AI). The problem of giving pieces of advice on songs and music is considered. A graph neuronal network was proposed as a modern tool for this purpose, however other approaches like matrix factorization are known [2]. The problem belongs to the social life culture, but it represents a model for a general approach. The consecutive tasks were as follows: downloading big data resources from website and simultaneous generation of graph using SNAP for Python. Then the graph neuronal network was created via a professional tool PyTorch Geometric. After that, the training was performed. The applied learning rules were based on reference descriptions. The so called vertex embeddings were utilized in the prepared application. The notion is similar to weights in classical neuronal networks, but to highlight the difference – the new name is commonly used. It is also connected with the special modification rules.

The paper is organized in the following way: underneath the reference study is given. In chapter two – chosen basic notions are introduced and explained. In chapter three, own application is described. Exemplary results are presented in the following chapter. In part number five, an analysis of chosen parameters is presented. Conclusions are drawn to finish the text.

The paper in based upon the master thesis of Mikołaj Grygiel, MSc, when Stanisław Zawiślak, university professor was a supervisor [4].

The idea of graph neuronal networks is relatively new one therefore the cited publications were edited just in eight recent years.

The numerous learning challenges involve working with graph data that contains rich relationships between its elements. Whether it's simulating physical systems, understanding molecular structures, forecasting protein interfaces, or categorizing diseases, these tasks demands models capable of extracting knowledge from graph-based inputs. For this reason, the proposal of the general design pipeline for graph neural networks models has been made. [11]

In paper “Computing Graph Neural Networks: A Survey from Algorithms to Accelerators“ authors are presenting a review of the different approaches to graph neural network in perspective of computing. The Paper also provides an analysis of both software and hardware acceleration schemes. Additionally the extensions to improve the support for GNNs by means of the popular libraries such as TensorFlow and PyTorch are described. [1]

Another example of the usage of graph neural networks in recent years is the Spectral Temporal Graph Neural Network (StemGNN), which is a cutting edge solution for Multivariate Time-series Forecasting that utilizes the concept of graph

neural networks. To demonstrate the accuracy and effectiveness of StemGNN authors conduct extensive experiments on ten real-world datasets. [3]

The graph neural networks can also be utilized in many ways in intelligent fault diagnostics. In paper "Multiscale Deep Graph Convolutional Networks for Intelligent Fault Diagnosis of Rotor-Bearing System Under Fluctuating Working Conditions" authors proposes a new algorithm called multiscale deep graph convolutional networks (MS-DGCNs), aimed at addressing this issue. [10] Authors of another paper "The emerging graph neural networks for intelligent fault diagnostics and prognostics: A guideline and a benchmark study" provide us with practical instructions on utilizing graph neural networks for intelligent fault diagnostics and prognostics. Framework created by the authors can be divided into two different branches. First for diagnosing faults at the node level and the second for diagnosing faults or regression at the graph level. In node-level fault diagnosis, each individual node within the graph is treated as a sample, while in the graph-level fault diagnosis, the entire graph is treated as a sample. In the framework, there are three techniques for creating graphs (KnnGraph, RadiusGraph, and PathGraph), along with two distinct data input options (Frequency domain and time domain). Additionally, there are seven GNN models and four methods for pooling in the framework. [8]

2. Some general remarks on the notions and tools

Neuronal network is a world-wide known idea. Sometimes they are built of layers where nodes are displaced in organized manner. Layers are connected by means of links – to which weights are assigned. However in the discussed problem the graph neuronal networks are utilized which touch the problem in a more direct way. It is relatively new idea therefore some basic data and features are describe in what follows.

2.1. Advisory systems and discussed particular one

Artificial Intelligence (AI) is a branch of knowledge which is just now under rapid and many-sided development. Expert systems can issue information, proposal of diagnosis for medicine doctors, engineers, bank clerks etc. In the paper, we reduce the purpose of such a system to an everyday life problem i.e. issuing suggestions on music compositions and songs. Nevertheless, (in every case), data base is needed which - in our problem - consists in the website based resources. They just exist and are downloadable form Aicorwd website. The problem is how to convert them from JSON format into the formal needed/readable graph for our software. It was solved by preprocessing data using two tools: PyTorch Geometric combined with SNAP for Python - prepared in the Stanford University, USA. [7] The graph created form Aicorwd's dataset called Spotify Million Playlist Dataset has 3 262 292 vertices and 65 464 776 nodes. Then the 'engine' is needed which processes the data, finally the system create a text of the advice. In our case, based upon the song (entered by a user) system gives an output i.e. a list of 10 similar songs.

The knowledge is stored in a graph which edges represent relations between the songs and the playlists. The learning procedure makes the relation to our song - more evident, easy to extract. In the consecutive iteration these relations are

emphasized via so called embeddings which are the representation of graph nodes. These embeddings can subsequently serve as a foundation for making predictions at the node, edge, or graph level – so finally it is possible to extract the most relevant songs.

2.2. Graph neuronal networks (GNN)

Like it was written above, graph neuronal network merges ideas of a graph with some general ideas of neuronal networks e.g. connections of net nodes in our case are converted into edges linking graph vertices, correction of weights – in our case – is changed into modification of embeddings assigned (in general) to graph nodes and edges. Training of GNN is performed via consecutive iterations where embeddings are modified according to introduced formulas. For every node in the graph, the structure of computation graph is defined based on the number of layers in the GNN. Message transformation and aggregation functions are used to update the embedding of the target node with the information form the neighborhood. The information are taken form the direct neighbors of target node, and each of the direct neighbors takes information from its own direct neighbors. That message exchange architecture makes the embedding modification capable of incorporate information both locally and more globally.

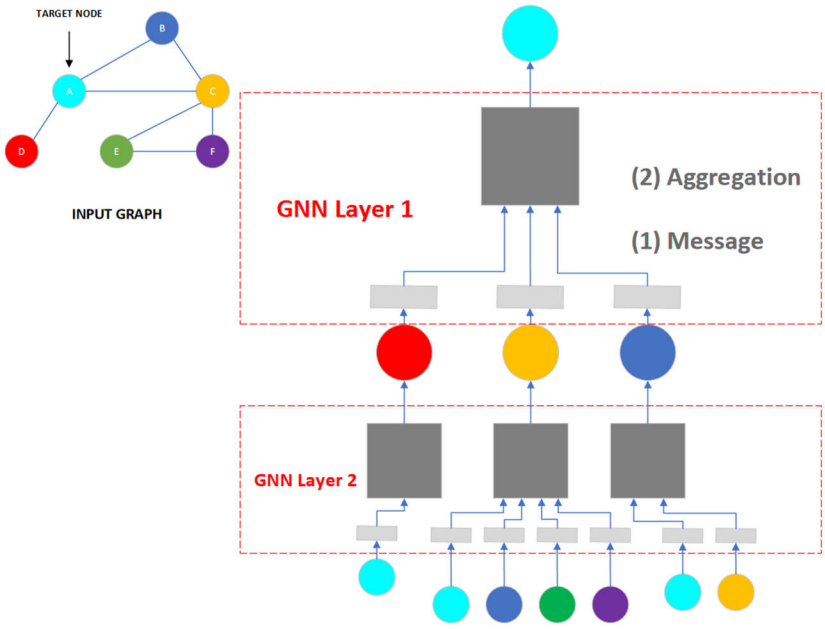

Figure 1. Target node update in graph neural networks using computation graph

2.3. Light graph convolution network (LightGCN)

The particular model of graph neural network used in the authors' application is the Light graph convolution network which is a state-of-the-art solution

for recommendations and advisory systems. The main concept of LightGCN was the complete removal of trainable weight matrices and nonlinear activation functions. This approach leads to substantial enhancements in processing speed. LightGCN is a graph neural network and like it was mentioned, it propagates the data from node embeddings across the local and global neighborhood in the graph using message and aggregation functions. As a result, the embeddings update mechanism remains relatively straightforward, as depicted below:

$$e_i^{(k+1)} = \sum_{u \in N_i} \frac{1}{\sqrt{|N_i|}\sqrt{|N_u|}} e_u^{(k)} \tag{1}$$

Where e_i^k and $e_u^{(k)}$represent the embeddings for vertices i and u after k layers of propagation, N_i and $N_u$represent the neighborhoods of vertices i and u, respectively. This equation describes the aggregation of messages transmitted between neighboring vertices. After performing k layers of propagation, the embeddings obtained at each layer are combined to create the final embedding for vertex i:

$$e_i = a_k \sum_{k=0}^{K} e_i^{(k)} \tag{2}$$

In equation (2), a_k represents the weight for layer k. However, the authors of the LightGCN suggests that a_k can be replaced with value $\frac{1}{K+1}$, which simplifies the equation to:

$$e_i = \frac{1}{K+1} \sum_{k=0}^{K} e_i^{(k)} \tag{3}$$

This change leads to improved efficiency in cases where there is no optimizer for the values of a_k. The authors of the LightGCN decided not to introduce an optimizer to keep the model simple. [5]

2.4. Neural networks and graph neural networks - differences

In case of classical ANN the structure is built based upon good practices, experience and the knowledge of the problem. Number of layers especially hidden layers, number of nodes in every layer are adjusted to the considered problem. Another issue consists is preparing a training data set as well as a teaching procedure which frequently uses an idea of back propagation (of errors). On contrary, GNN consists in generation agraph which itself represents a problem and edges represent real theoretical connections between vertices which also represent some real notions (in our case songs). The meaning of weights is also different – in ANN they are abstract values giving the aggregate effect via the whole network, in case of GNN – the notion of embeddings was introduce to highlight the direct meaning of every single embedding. The layers in ANN can be divided into three groups. Input layer which receives the input. Output layer that predicts the output. In between exist the hidden layer or multiple layers which perform most of the computation required by our network. The idea of layers in GNN is different and the main goal of single

GNN layer is to compress or aggregate a set of messages from the target node neighbors. The set of messages can be represented as a set of vectors (embeddings). Like it was shown in Figure 1, the single layer of GNN consists of two components. The message transformation and the message aggregation operation. Among the differences between various GNN architectures, they basically differ in how these two operations are being done. So, to conclude – despite the same name – the meaning and properties of the layers in two types of neuronal networks are essentially different.

2.5. Embeddings and hyperparameters

Like it was mentioned embeddings are assigned to vertices. Node embeddings are the method of mapping the vertices in the graph into d-dimensional space and represent them as the vector of d numbers. They also capture the structure of the graph in the way that higher similarity between node embedding indicates their similarity in the graph. In the described application created by author, embeddings established after graph neural network training were used to predict the most relevant songs for the given music piece.
The hyperparameters are variables that serves as a means to control the learning process. In contrast to normal parameters (typically related to node weights), which are determined through the training process. Selecting the optimal hyperparameters setup for machine learning models significantly increases the model's effectiveness. In authors' implementation of graph neural network the hyperparameters were tuned manually, but there are papers that are presenting algorithms used to tune the hyperparameters. [9]

3. Description of the application

Own application was prepared in Python programming language which is commonly used and many on-line tutorials and manuals are available.

Table 1. Graphs created as inputs for graph neural network

	Vertices	Playlists	Songs	Edges
Original graph	3 262 292	1 000 000	2 262 292	65 464 776
K-core graph, K=120	56 600	46 444	10 156	7 044 802
K-core graph, K=128	33 316	26 758	6 558	4 174 904

The idea of a code for a piece music was proposed, the collected information encloses: title, author of lyrics, author of music, band, singer, members of the band, year of first presentation etc. The advisory system consist of three parts.

Learning procedure was prepared consisting in PyTorch Geometric library. Application implemented the LightGCN graph neural network model which was described in 2.3 subsection. The original graph created form Aicrowd's dataset was too large to perform training on it, so it was necessary to create the smaller graphs form it. The smaller graphs were K-cores of the original graph for K=120 and K=128.

The two smaller input graphs have been under training for 50 and 100 epochs, depending on values of hyperparameter of batch size.
The learned embeddings were used by REST API created in Python framework Fast API to get top ten most relevant songs for entered piece. To achieve this, API endpoint use torch function from the PyTorch library. The interactive user interface that presents the predictions given by learned embeddings was prepared in React – JavaScript library.

4. Exemplary results

The exemplary results are presented underneath, showing the table of content prepared for music piece titled "Fortunate Son" composed by Creedence Clearwater Revival, based on data presented by authors' application and Spotify. More detailed description and other comparisons are given in the thesis [4].

Table 2. Songs recommended by authors' application and Spotify

Authors' application	Spotify
Led Zeppelin – Stairway To Heaven	Ram Jam – Black Betty
Lynyrd Skynyrd – Free Bird	Lynyrd Skynyrd – Free Bird
Eagles Hotel California – Remaster	Creedence Clearwater Revival – Bad Moon Rising
Kansas Carry on Wayward Son	The Rolling Stones – Paint It, Black
Boston More Than a Feeling	The Rolling Stones – Sympathy For The Devil
Aerosmith Dream On	The Jimi Hendrix Experience – All Along The Watchtower
Lynyrd Skynyrd Sweet Home Alabama	Norman Greenbaum – Spirit In The Sky
The Who Baba O'Riley	Creedence Clearwater Revival – Run Through the Jungle
Tom Petty Free Fallin'	Steppenwolf – Born to Be Wild
Guns N' Roses Sweet Child O' Mine	The Who Baba O'Riley

The results given in table above indicates that the songs recommended by authors' program could be consider as relevant. All of them are form rock genre. The 2 pieces are the same as the Spotify given suggestion. Eight out of ten recommendation were released between 1971 and 1976. "Fortunate Son" composed by Creedence Clearwater Revival was released in second half of 1969. It means that the music pieces are form similar time iterval. The other two come from 1987 and 1989 but still belong to the rock category.

5. Analysis of chosen parameters

Several parameters are usually introduced to controls such advisory systems, in our case there are so called hyperparameters of GNN. We established the values of these parameters based of the references study. Obviously, the values can be changed within some intervals/ranges. So, aiming for proper tuning of the system – some analyses of meaning and importance of particular parameters were widely performed [4].

Here, a few results are shown enabling confirmation of the formulated conclusions. For hyperparameter analysis the evaluation metric of recall at k was used, as a main factor of learned embeddings quality. As a result of that choice, a higher value of recall at k indicates better performance of the training model. Recall at k measures the percentage of relevant items discovered within the top-k recommendations. There are more evaluation metric that could be found in following paper. [6] However, this work is exclusively focused on the evaluation metric of recall at k.

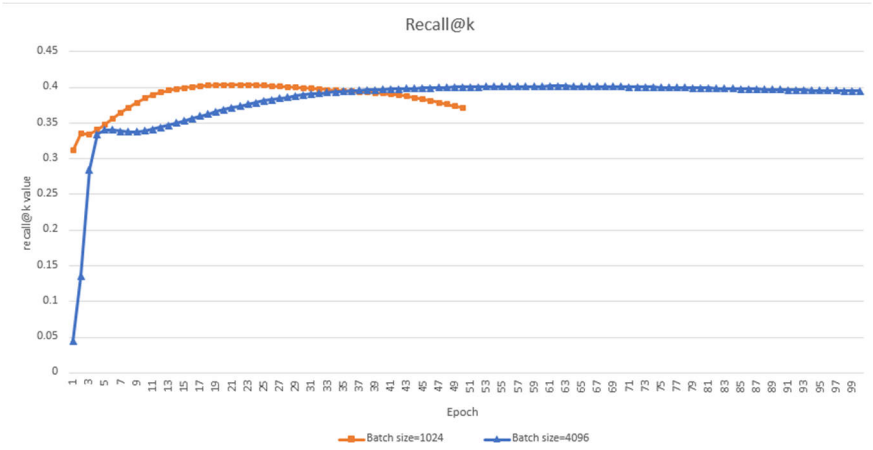

Figure 2 Change in recall@k values over 100 epochs, depending on the batch size. For a batch size of 1024, the training was stopted after 50 epochs. Keeping the value of k at 300 and the graph's K-core constant, with K equal to 120

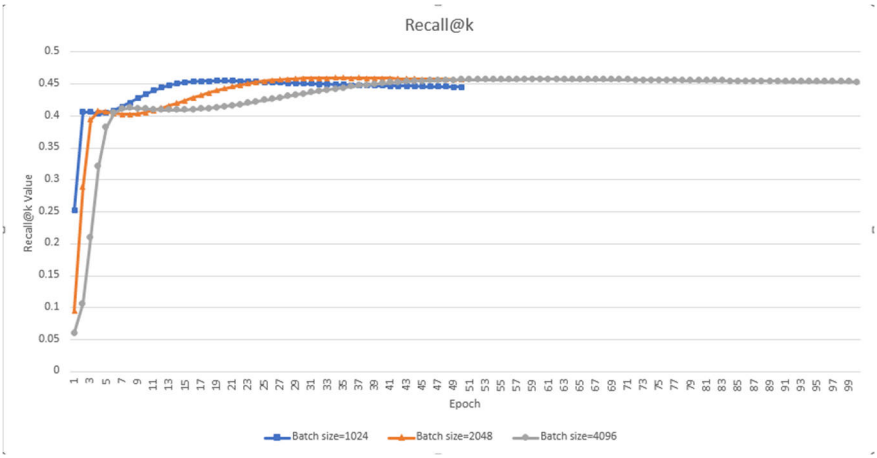

Figure 3 Change in recall@k values over 100 epochs depending on the batch size. For batch sizes of 1024 and 2056, the training was stopped after 50 epochs, respectively. Maintaining a constant value of k=300 and a constant K-core of the graph, with K=128

Conclusion form the Figure 2 and 3 is as follows. In implemented model of LightGCN the lower the batch size then less numbers of epochs is needed to achieve the highest

evaluation value - recall at k. The lowering value of recall at k can signalize the overfitting of the model. Since the time taken for each epoch varies, with a strong dependence on the batch size hyperparameter, it becomes necessary to examine the time required to achieve the highest recall at k value in order to gain a comprehensive understanding of the discussed learning process. That case and further wide analysis of model training can be found in the thesis. [4] The specific embedding that was used to obtain advice for a given set of music pieces was acquired as a result of GNN learning for a K-core graph, with K=128, a batch size of 2048, and after 37 epochs of training. It was chosen due to having the highest value of recall at k among all other embeddings.

6. Conclusions and final remarks

In the present paper, the created advisory system is described. The application – upon the entered song - generates a list of 10 similar compositions. System works effectively and the obtained results are similar to those given by a professional one (SPOTIFY). The utilized graph neuronal networks are new, contemporary developed idea which was proposed upon traditional world-widely known neuronal networks. However, GNN are different in such a way that a generated graph represents connections between considered items (songs). They are represented by graph vertices. Weights are assigned to vertices and edges, but in the presented application, only weights on vertices are considered. They are called embeddings to highlight the difference between classical-NN and graph-NN. In general, just graph structure represents (in essential way) dependences between considered items – in our case the songs. In our case, another aspect is big data type of resources. As we know, every year millions of compositions are produced in every country all over the world. But due to the www, social media and TV - all the people on earth are connected and a song from every country could be a world-wide famous hit. The considered problem represents a class of problems in which ideas, clips, adverts, photos etc. are available for everyone in her/his web-device laptop or mobile phone in every particular place. So, the problem related to big data issue was effectively solved just for music fans. Acquired knowledge, understanding of the utilized ideas and practical expertise enables future update of the described software as well as - even more important - solving similar problems.

REFERENCES

1. ABADAL S., et al.: Computing graph neural networks: A survey from algorithms to accelerators. ACM Computing Surveys (CSUR), 2021, 54.9: 1-38.
2. BENZI K., et al.: Song recommendation with non-negative matrix factorization and graph total variation. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, China, 2016, pp. 2439-2443.
3. CAO Defu, et al.: Spectral temporal graph neural network for multivariate time-series forecasting. Advances in neural information processing systems, 2020, 33: 17766-17778.

4. GRYGIEL M.: System of recommendation of music pieces built utilizing a graph neuronal network (in Polish), Master Thesis, University of Bielsko-Biala, 2023.
5. HE Xiangnan, et al.: LightGCN: Simplifying and powering graph convolution network for recommendation. Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, 2020.
6. HOSSIN M., et al.: A review on evaluation metrics for data classification evaluations. International journal of data mining & knowledge management process, 2015, 5.2: 1.
7. LESKOVEC J., et al.: SNAP: A General Purpose Network Analysis and Graph Mining Library. CoRR, 2016, tom abs/1606.07550.
8. LI Tianfu, et al.: The emerging graph neural networks for intelligent fault diagnostics and prognostics: A guideline and a benchmark study. Mechanical Systems and Signal Processing, 2022, 168: 108653.
9. YANG Li, et al.: On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing, 2020, 415, 295–316.
10. ZHAO Xiaoli, et al.: Multiscale deep graph convolutional networks for intelligent fault diagnosis of rotor-bearing system under fluctuating working conditions. IEEE Transactions on Industrial Informatics, 2022, 19.1: 166-176.
11. ZHOU Jie, et al.: Graph neural networks: A review of methods and applications, 2020.