Graph recommender systems in Automatic Playlist Continuation

Marko Panushkovski
Faculty of Computer Science and Engineering
Ss. Cyril and Methodius University
Skopje
marko.panushkovski@students.finki.ukim.mk

Sonja Gievska
Faculty of Computer Science and Engineering
Ss. Cyril and Methodius University
Skopje
sonja.gievska@finki.ukim.mk

ABSTRACT

Recommender systems are key point of the procedure for offering new and previously unknown content to certain user. As a product of rapid market growth, the recommender systems need to be aware of the current trends in order to give precise recommendations. That means they have to be fed with new data regularly. Beside the part of having a consistent data that flows into a model, the importance of data representation cannot be neglected. According to the current trends for data representation in this field, which focus towards mapping data in multi-dimensional space, this research examines the feasibility of using a graph-based recommender system for next song prediction with two state-of-art representation models. In fact, the language representation model BERT and GraphSage based representations of nodes and edges were investigated. Due to homogenous nature of GraphSage representation model, additionally a HinSage algorithm was investigated. Link prediction was used to identify the songs that follow in a given playlist. The results of our research are encouraging and open a new directions for further investigations.

CCS CONCEPTS

▸Design and analysis of algorithms ~ Graph-based recommender systems ~ Feature extraction ~ Natural language processing ~ Neural networks ~ ▸Automatic playlist continuation ~ Spotify million playlist dataset

KEYWORDS

graph-based recommender systems, link prediction, vector embeddings

1 INTRODUCTION

The general approaches for building recommender systems which include content-based and collaborative filtering tend to show their disadvantages at some point in time. The problem of data sparsity is present at the very beginning in collaborative filtering method, meanwhile a lack of diversity in recommendations is a problem that emerges in later phase of content-based filtering. A lot of research efforts that try to improve the efficacy of these systems were investigated, whether that is making a hybrid system by combining the content-based and collaborative filtering methods or additionally integrating latent factor matrices of preferences in the process **[2]**.

Being aware of these problems in recommender systems along with the current trends for applying deep learning in the field of natural language processing was motivation to make examination of how accurate a graph-based recommender system for next song prediction in playlist will be by integrating language representation model BERT with the GraphSage based representations of nodes and edges. The automatic feature extraction offered by BERT and the inductive conclusion process which is present with GraphSage were factors that tend to give promising results in this research. No audio or image feature were used.

Following a short citation of the related research in this field in Section 2, the complete methodology was explained in Section 3 of this research. The comparison between GraphSage and its variant for heterogenous graphs – HinSage **[4]** were discussed in Section 4, while Section 5 concludes the research.

2 REALTED WORK

This paper is product of observation and examination of similar tasks in this field that contributed in gaining best practices of applying certain algorithms for the problem of designing

recommender system. With the RecSys Challenges 2018 a number of models were purposed that used ensemble methods for prediction of next song in a playlist. Even though the purposed models of the challenge varied from traditional machine learning models to few deep learning models, in this study a design for solution is seen through Graph Convolutional Networks (GCN). With the help of paper published under Creative Commons **[1]** in 2020 a possible solution for automatic playlist continuation was purposed in this study. The key point in using GCN algorithm is inductive conclusion which is not present in Word2Vec or Node2Vec algorithms (transductive conclusion). The technique for fusing diverse in nature features was used in the following purposed models in order to improve and diversify recommendations.

3 METHODOLOGY

The process of creating a model is based on construction of a bipartite graph which represents playlists and tracks as nodes connected with an edge if a track belongs to a certain playlist. Information about playlists, tracks and linkage between them were extracted from the Million Playlist Dataset from Spotify. Vector representation of the data i.e. embeddings are used when textual information about tracks and playlists is being encoded with language representation model BERT. Result embeddings from BERT model are used in later phase of generating node and edge embeddings with GraphSage algorithm.

BERT is state-of-the-art algorithm which is used in processing textual information. It is considered as revolutionary advancement since it has the capability to read an entire sequence of words, differentiating from Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). This advancement of model BERT is accomplished with use of Transformers, an architecture that relies on attention mechanism which takes into account words according to their importance in a given context of sentence **[5]**.

GraphSage is a framework that allows inductive representation learning. It is used to generate low-dimensional vector representations for nodes and is useful in graphs containing nodes with large number of attributes. The algorithm of GraphSage is done by sampling neighborhood of a given node, from which feature information is being aggregated in order to predict graph context and label **[3]**.

HinSage is an algorithm that extends GraphSage in order to support heterogeneity of nodes and edges in a graph. The algorithm uses separate neighborhood weight matrices for every unique ordered tuple of (node1-edge-node2), and separate self-feature matrices for every node type. Given this if we have 2 node types there will be 4 different edge types between the nodes.

A recommendation model based on link prediction for assigning tracks to playlist is purposed in this study. The model is fed with pairs of positive samples (nodes having edge between them) and negative samples (nodes not having edge between them), which performs the task of binary classification.

3.1 Dataset

Million Playlist Dataset from Spotify contains 1 million playlists including playlist titles and track titles, created by users on the Spotify platform between January 2010 and October 2017 **[6]**.

A subset containing 998,409 connections between 15,000 unique playlists and 218,129 unique tracks has been selected for evaluating the proposed graph-based recommender model in this paper.

Initial data that has been collected into groups of thousand playlist samples was transformed from JSON format into CSV format. Having properties of relational data, the connection between a playlist and a track was modeled as M:N relation, represented in three tables: two separated tables for playlists and tracks with their features, and third table that keeps references of playlists and tracks belonging to them.

3.2 Method

3.2.1 Graph construction

An undirected, bipartite graph was constructed based on the reformatted data from the Million Playlist dataset, where tables that represented playlists and tracks with their features were taken as nodes, while the table holding references to them constructed the edges of the graph.

3.2.2 Generation of node features

Minimal preprocessing of text has been performed in order to satisfy the requirements for pretrained model BERT by converting it to lowercase. In order to generate vector representation of text a pretrained, uncased BERT model with 2 hidden layers (Transformer blocks), a hidden size of 256 and 4 attention heads were used. This model was fed with text, containing fully concatenated data from tracks, which is track name, album name, artist name as well as their unique identifiers. As a result from this model semantically meaningful word embeddings of size 256 were derived.

Since tracks are the smallest building blocks that define a playlist, their vector representation was used to generate the vector representation for playlists. By averaging the vectors of all tracks present in a given playlist, approximate vector representation of that playlist was produced. Furthermore, having equal dimensionality of both types of nodes has been advantage when using GraphSage algorithm.

3.2.3 Generation of node embeddings

Low-dimensional vector representations of nodes from previously created bipartite graph were generated with help of GraphSage and its variant HinSage model. Receiving playlist track pair of nodes as inputs, these models transform them into node embeddings which are consumed in the further layers of this process.

3.2.4 A recommender model based on link prediction

A layer of link classification was utilized to combine node embeddings from the previous step into edge embeddings. Binary classification happens with help of Activation layer which uses sigmoid as activation function. The models were trained in 10 epochs using Adam optimizer with learning rate of 1e-3, along with binary cross-entropy loss and binary accuracy.

4 DISCUSSION OF RESULTS

A test dataset is created by subtracting 10% of all tracks present in a given playlist for each playlist in the dataset. Having an ordered list for tracks belonging to a playlist, it was the last 10% of all tracks that were chosen. Since link prediction was modeled as binary classification (an edge is present or not), the models required negative samples (tracks that do not belong to playlist). After randomly sampling non-existent edges, the train and test datasets were constructed and training of models was performed.

4.1 Comparison of training accuracy between GraphSage and HinSage

Both of GraphSage and HinSage models purposed in this study were trained in 10 epochs with same hyper parameters and they gave the following results

	Loss	Binary accuracy
GraphSage	0.55	0.75
HinSage	0.52	0.79

After these models were trained, it is shown that HinSage model performed slightly better than GraphSage model on the training dataset.

4.2 Evaluation of the recommender models based on link prediction

Evaluation metrics chosen after test dataset yielded results are ROC AUC (Receiver Operating Characteristic – Area Under Curve) and AP (Average precision). ROC AUC score is significant in describing accuracy of the model, because it tells us how much a model is capable of distinguishing between classes. On the other hand Average precision will give us a total overview of how the model performed after every correct guess during its evaluation.

	AUC-ROC	AP
GraphSage	0.83	0.76
HinSage	0.81	0.74

Test dataset results gave advantage to the GraphSage model instead of HinSage. Although there is a slight difference of 0.02 between these results, both of the model's performances were satisfying with respect to data quantity they used. Further examination of their performances could be done with more data for training and testing purpose taken from the Million Playlist Dataset. Another possible direction for advancing these models is adding weight on track-playlist edges according to track's position in certain playlist.

5 CONCLUSION

This paper presents an overview of latest advancements in the field of natural language processing and graph representation through a concrete example. With this example of link prediction for assigning track to a playlist, the language representation model BERT and GraphSage algorithm show their superiority in this task like in many others.

REFERENCES

[1] Boosting recommender systems with advanced embedding models – Gjorgjina Cenikj

[2] Chapter 9 Recommendation Systems – Stanford InfoLab

[3] Snap Stanford Edu – GraphSage algorithm

[4] StellarGraph – Link prediction with heterogenous GraphSage (HinSage)

[5] TechTarget – What is BERT and how does it work?

[6] AICrowd – Spotify Million Playlist Dataset Challenge