Vector Space Representation of Movies and Music Albums Using Node Embedding Methods

Vid Keršič[1]

[1]*University of Maribor, Faculty of Electrical Engineering and Computer Science*
Koroška cesta 46, 2000 Maribor
E-mail: vid.kersic@um.si

Abstract

The performance of machine learning methods improves every year. While commonly used machine learning methods that process data in vector form can efficiently perform many tasks, non-Euclidean data structures like graphs are often present in different real-world problems. Therefore, in recent years, machine learning methods for node embedding, which transform nodes of graphs into vector space, have gained a lot of traction. In this paper, we conduct an analysis of movies and music albums to explore their usage in recommender systems. We construct graphs from Wikipedia articles by following the contained hyperlinks. Different node embedding methods are used to embed nodes of the obtained graphs in a vector space, which allows us to analyze them more efficiently, focusing on visualization, similarity (recommendations), and classification of movies and albums into genres. We achieved a classification accuracy of 88.5% for movies and 89.3% for albums.

1 Introduction

Machine learning has made great strides in recent years, successfully performing various tasks like classification, regression, and clustering [1]. Although entirely different machine learning methods are used for the mentioned tasks, they have a similar input data structure, which is a one-dimensional, two-dimensional, or high-dimensional vector [2]. But nowadays, many scientific fields like social networks deal with non-Euclidean data structures like graphs [2]. For that reason, machine learning methods for node embedding, which transform nodes of graphs into vector space, are becoming very popular.

Node embedding methods have been successfully used to increase the effectiveness of recommender systems. Embedding nodes of graphs constructed from Wikipedia articles improved the ability to find similar historical figures, perform a recommendation of articles to editors of Wikipedia, and classify books into genres [3, 4, 5]. In this paper, we use node embedding methods to analyze movies and music albums and explore their usability in recommender systems, e.g. movie and music album recommendation. We construct a graph from Wikipedia articles by following the available hyperlinks and then using machine learning methods for node embedding on the obtained graphs analyze movies and albums, focusing on visualization, similarity (recommendations), and classification of movies and albums into genres.

In the next sections, we review different machine learning methods for node embedding, describe the data collection and graph construction process, and present the results of the analysis of movies and music albums.

2 Node Embedding

A graph is a data structure defined as $G = (V, E)$, where V is a set of nodes, and E is a set of edges. Node embedding is defined as:

$$f : V \to \mathbb{R}^d, \tag{1}$$

where d is the dimension of vector space. Node $v_i \in V$ is transformed into vector $\mathbf{z}_i \in \mathbb{R}^d$. The objective of node embedding is to preserve the relations between the nodes and the structure of the graph. Over the past few years, many machine learning methods for the vector embedding of graph nodes have been introduced. In this section, we review the methods that were used in this analysis.

The first widely used node embedding methods were shallow embedding methods, where a single matrix $\mathbf{Z} \in \mathbb{R}^{d\times|V|}$ represents the embedding of all nodes (i-th column is vector $\mathbf{z}_i$) [6]. One of the most popular shallow embedding methods is Deepwalk [7]. DeepWalk is a random-walk based method that uses the Skip-gram model to learn node embedding. Skip-gram is trained on the generated random walks in an unsupervised manner. Its objective is to embed a pair of nodes that are close to each other (inside the sliding window) in random walks to also be close in the vector space, where the similarity between vectors is measured using the inner product. The probability of node v_j appearing close to node v_i is defined by the softmax function:

$$p(v_j|v_i) \approx \frac{e^{\mathbf{z}_i{}^T\mathbf{z}_j}}{\sum_{v_k\in V} e^{\mathbf{z}_i{}^T\mathbf{z}_k}} \tag{2}$$

During training, Skip-gram attempts to minimize the following cost function:

$$\mathcal{L} = \sum_{(v_i,v_j)\in\mathcal{D}} -\log\left(p(v_j|v_i)\right), \tag{3}$$

where $\mathcal{D}$ is the training set of pairs of nodes from the generated random walks. Since the normalization factor in softmax is expensive to calculate, DeepWalk uses hierarchical softmax [7]. Another popular random-walk based method is node2vec, which differs slightly from DeepWalk in the process of generating random walks and the calculation of the normalizing factor [8]. Random walks are generated in a biased way, using the hyperparameters p and q. While p determines the likelihood of the walk returning to the previous node, q determines the likelihood of going further away from the last node. Therefore, random walks can be biased towards a breadth-first search (BFS) and depth-first search (DFS) of a graph. Instead of hierarchical softmax, node2vec employs negative sampling [8].

Shallow embedding methods have some limitations, like the inability to capture non-linearity in a graph structure [6]. Methods based on deep autoencoders like SDNE (Structural Deep Network Embedding) were introduced to solve the non-linearity problem [9]. Deep autoencoders consist of two deep neural networks - an encoder and a decoder. The encoder transforms input data into the embedding, while the decoder tries to reconstruct the input data from the embedding. SDNE is a semi-supervised deep model that attempts to preserve the graph's local and global structure using first-order and second-order proximities. First-order proximity describes the local structure and aims to produce similar embeddings for directly connected nodes. Second-order proximity describes the global structure and seeks to embed nodes with similar neighborhood structures closer to one another in the vector space. The loss functions of proximities and regularization define the loss function of SDNE:

$$\begin{aligned}\mathcal{L}_{1st} &= \alpha \sum_{i,j=1}^{n} s_{i,j} \|\mathbf{z}_i - \mathbf{z}_j\|_2^2 \\ \mathcal{L}_{2nd} &= \sum_{i=1}^{n} \|(\hat{\mathbf{s}}_i - \mathbf{s}_i) \odot \mathbf{b}_i\|_2^2 \\ \mathcal{L}_{mix} &= \mathcal{L}_{1st} + \mathcal{L}_{2nd} + \gamma \mathcal{L}_{reg},\end{aligned} \tag{4}$$

where $\mathbf{s}_i$ is the neighborhood vector of node v_i ($s_{i,j} \in \mathbf{s}_i > 0$ if edge between nodes v_i and v_j exists) and $\hat{\mathbf{s}}_i$ is the reconstructed neighborhood vector. The values α, $\mathbf{b}_i$, and γ are hyperparameters.

3 Data Collection and Graph Construction

The analysis of movies and music albums is done on the graphs constructed from Wikipedia articles and hyperlinks. Because we did not find any publicly available dataset containing movies and albums with hyperlinks to their Wikipedia articles, we created the dataset ourselves using web scraping. We extracted the list of movies from the website *TheNumbers*, where we chose five genres and 100 movies per genre: superhero, kids fiction, horror, musical, and fantasy [10]. A similar procedure was repeated for albums. We extracted the list from the website *DigitalDreamDoor* and also chose five genres (100 albums per genre): country, rap, metal, rock, and electronic [11]. To obtain Wikipedia articles of movies and albums, we wrote a custom script. Because some movies and albums do not have Wikipedia articles, we had to check the lists manually. Table 1 shows the number of movies and albums per genre after data cleansing.

Table 1: The number of movies and albums per genre after data cleansing.

Genre	Movies	Albums
Genre 1	100	93
Genre 2	100	100
Genre 3	100	100
Genre 4	93	86
Genre 5	92	99
Total	485	478

From the lists of movies and albums, we constructed the graphs by following hyperlinks in the articles. We created two graphs: one for movies and one for albums. Nodes in graphs are Wikipedia articles, and edges are hyperlinks between articles. Both graphs are undirected and unweighted. We only included hyperlinks from sections of the articles, which in our opinion best characterize movies and albums. The hyperlinks selection is one of the most critical steps because wrongly selected hyperlinks can connect completely different movies and albums (e.g. hyperlinks to film studios). We constructed graphs by running DFS with depth limited to 2 from every movie and album from the list. All of the discovered Wikipedia articles, not only those describing movies and albums, were added to the graphs. The constructed graph for movies consists of 3,216 nodes and 7,520 edges, while the graph for albums consists of 3,601 nodes and 9,289 edges. Figure 1 shows the graph for movies.

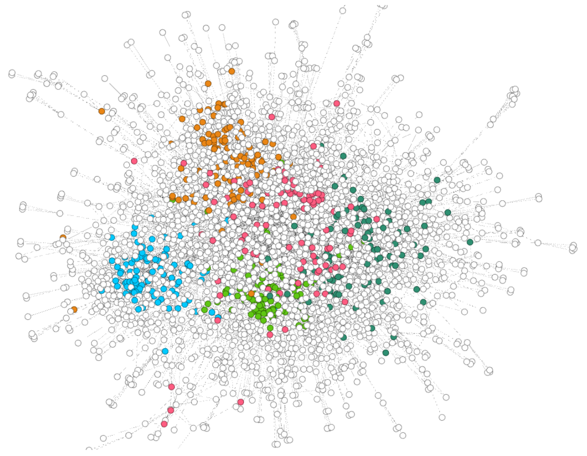
Figure 1: The constructed graph for movies. The colors of the nodes represent genres of movies: superhero (blue), kids fiction (green), horror (brown), musical (turquoise), and fantasy (pink). White nodes represent other Wikipedia articles.

4 Results

We compared the embeddings produced by four methods: DeepWalk, node2vec-BFS, node2vec-DFS, and SDNE.

The dimension of the node embedding was set to 128. The hyperparameters of the methods were tuned using a random and grid search. We generated 25 random walks from every node in random walk-based methods. For DeepWalk, we used *window size* 5 for movies and 15 for albums, and *walk length* 80 for movies and 60 for albums. For node2vec-BFS, we used p=0.5, q=2.0, *window size* 10 for movies and 15 for albums, and *walk length* 80 for both movies and albums. For node2vec-DFS, we used p=2.0, q=0.5, *window size* 15 for both movies and albums, and *walk length* 80 for both movies and albums. For SDNE, we used α=10^{-6} and β=150 for both movies and albums. The deep neural networks of SDNE consisted of two layers (3216-128 and 128-3216) for movies and three layers (3601-512-128 and 128-512-3601) for albums.

In this analysis, we only kept the embeddings of Wikipedia articles for movies and albums from the lists. We reduced the dimensionality of embeddings from 128 to 2 using principal component analysis (PCA) for visualization. Figure 2 shows the closest vector embeddings to the superhero movie The Dark Knight. Figure 3 shows the embeddings of all movies and albums colored by their genre.

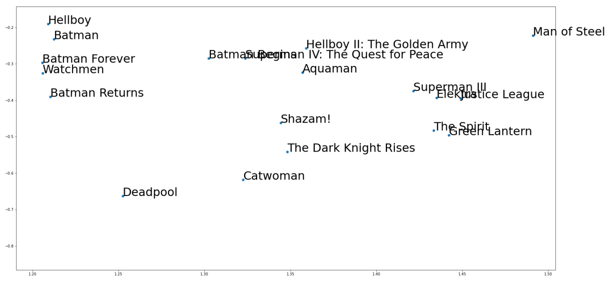

Figure 2: The closest vector embeddings to the superhero movie The Dark Knight produced by the method node2vec-DFS. All visualized movies are superhero movies, and many are from the Batman film series.

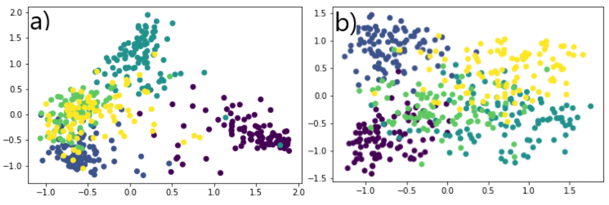

Figure 3: The embeddings of movies (a) and albums (b) produced by the method node2vec-DFS. Colors of the visualized vector embeddings represent genres (movies: superhero (violet), kids fiction (blue), horror (turquoise), musical (green), fantasy (yellow); albums: country (violet), rap (blue), metal (turquoise), rock (green), electronic (yellow)). Almost all movies and albums within the same genre have similar vector embeddings. Many musicals, fantasy, and kids fiction movies overlap. The reason is that many kids fiction movies are also fantasy movies, and they contain similar hyperlinks as fantasy movies. Some electronic, rock, and metal albums also overlap.

Cosine similarity was used to calculate the similarity between vector embeddings:

$$cos(\theta) = \frac{\mathbf{z}_i{}^T \mathbf{z}_j}{|\mathbf{z}_i||\mathbf{z}_j|} \tag{5}$$

The similarity between vectors is used for recommending movies and music albums. Table 2 shows the five most similar movies to the superhero movie X-Men: Dark Phoenix, and Table 3 shows the five most similar albums to the rap album Escape. SDNE embeds similar movies and albums much closer in a vector space than node2vec-DFS. However, because the graphs contain all articles from Wikipedia, the movie (album) articles can be far apart from each other. Because SDNE uses first and second-order proximities as similarity measurements, it fails to embed well articles that do not have a lot of similarity with others and are further away. On the other hand, random walk-based methods can explore a deeper part of the graphs.

Table 2: The top five most similar movies to the superhero movie X-Men: Dark Phoenix using cosine similarity as a similarity measurement between embeddings. For SDNE, all movies are from the same film series, while for node2vec-DFS the last one is not, but is still a superhero movie from the same film studio.

Method	Movie	Cos. sim.
node2vec-DFS	X-Men: Days of Future Past	0.852
	X-Men: The Last Stand	0.815
	X-Men 2	0.787
	X-Men: First Class	0.785
	Avengers: Endgame	0.759
SDNE	X-Men: Days of Future Past	0.929
	X-Men: First Class	0.922
	X-Men 2	0.904
	X-Men: The Last Stand	0.904
	X-Men: Apocalypse	0.878

Table 3: The top five most similar albums to the rap album Escape using cosine similarity as a similarity measurement between embeddings. In this case, SDNE generates similar embeddings for entirely different albums, while in the case of node2vec-DFS, all albums are from the same genre.

Method	Album	Cos. sim.
node2vec-DFS	Raising Hell	0.621
	How Ya Like Me Now	0.587
	King of Rock	0.583
	Run-D.M.C.	0.566
	The Low End Theory	0.501
SDNE	Ride the Lightning	0.728
	Mantronix: The Album	0.723
	Demon Days	0.722
	Black Sabbath	0.712
	Post	0.709

Multiclass logistic regression (one-vs-all) with 10-fold cross-validation was used for classification into genres. We evaluated classification performance using accuracy, precision, recall, and F1 score. Table 4 shows the classification accuracy of different methods. All methods achieve high classification accuracy for movies, while SDNE fails for albums. Because node2vec-BFS and SDNE achieve lower classification accuracy for albums, we concluded that album articles are more distant from each other than movie articles. Tables 5 and 6 show detailed classification results for movies and albums using node2vec-DFS. We achieved F1 scores higher than 0.75 for all genres.

Table 4: Classification accuracy for movies and albums. The method node2vec-DFS achieved the highest classification accuracy - 88.5% for movies and 89.3% for albums.

Method	Movies	Albums
DeepWalk	0.885	0.887
node2vec-BFS	0.882	0.867
node2vec-DFS	0.885	0.893
SDNE	0.854	0.662

Table 5: Detailed classification results for movies using the method node2vec-DFS. The worst classified movie genres were kids fiction and fantasy, which can also be seen on the node embedding visualization.

Genre	Precision	Recall	F1
superhero	0.973	0.954	0.964
kids fiction	0.811	0.856	0.833
horror	0.942	0.948	0.945
musical	0.890	0.884	0.887
fantasy	0.806	0.774	0.789

Table 6: Detailed classification results for albums using the method node2vec-DFS. The worst classified album genres were rock and electronic, while the other three genres had an F1 score higher than 0.92.

Genre	Precision	Recall	F1
country	0.903	0.942	0.922
rap	0.937	0.950	0.943
metal	0.957	0.928	0.942
rock	0.796	0.781	0.788
electronic	0.861	0.853	0.857

5 Conclusion

In this paper, we used node embedding methods on the graphs constructed from Wikipedia articles to analyze movies and music albums, and explored their usability in recommender systems, focusing on visualization, similarity (recommendations), and classification into genres. We compared embeddings produced by four node embedding methods. The node2vec-DFS method achieved the best classification accuracy for both movies and albums: 88.5% for movies and 89.3% for albums. While all methods achieve high classification accuracy for movies, SDNE fails for albums. We have shown that calculating similarities between vectors produced by node embedding methods can be used for recommending movies and albums.

In the future, the analysis can be expanded in several ways. The nodes of the graphs can be extended with features, and different neighborhood aggregation methods (e.g. a graph convolutional network) can be used to produce node embeddings. An analysis of less extensive Wikipedia articles, which do not contain as many hyperlinks as movie and album articles, can also be performed.

6 Acknowledgements

This paper is closely related to my bachelor's thesis. I want to thank associate professor Damjan Strnad and assistant Štefan Kohek for their guidance and assistance.

References

[1] G. Rebala, A. Ravi, and S. Churiwala: An Introduction to Machine Learning. Cham: Springer International Publishing, 2019.

[2] M. M. Bronstein, J. Bruna, Y. Lecun, A. Szlam, and P. Vandergheynst: Geometric Deep Learning: Going beyond Euclidean data, IEEE Signal Processing Magazine, vol. 34, no. 4. Institute of Electrical and Electronics Engineers Inc., pp. 18–42, 01-Jul-2017, doi: 10.1109/MSP.2017.2693418.

[3] Y. Chen, B. Perozzi, and S. Skiena: Vector-based similarity measurements for historical figures, Inf. Syst., vol. 64, pp. 163–174, Mar. 2017, doi: 10.1016/j.is.2016.07.001.

[4] O. Moskalenko: Convolutional Graph Embeddings for article recommendation in Wikipedia, 2019.

[5] Building a Recommendation System Using Neural Network Embeddings. [Online]. Available: https://towardsdatascience.com/building-a-recommendation-system-using-neural-network-embeddings-1ef92e5c80c9. [Accessed: 19-Jun-2020].

[6] W. L. Hamilton, R. Ying, and J. Leskovec: Representation Learning on Graphs: Methods and Applications, Sep. 2017, arXiv: 1709.05584.

[7] B. Perozzi, R. Al-Rfou, and S. Skiena: Deepwalk: Online learning of social representations, in Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 701–710.

[8] A. Grover and J. Leskovec: node2vec: Scalable Feature Learning for Networks, Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., vol. 13-17-Augu, pp. 855–864, Jul. 2016.

[9] D. Wang, P. Cui, and W. Zhu: Structural Deep Network Embedding, in Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, vol. 13-17-Augu, pp. 1225–1234, doi: 10.1145/2939672.2939753.

[10] The Numbers - Where Data and the Movie Business Meet. [Online]. Available: https://www.the-numbers.com/. [Accessed: 19-Jun-2020].

[11] DigitalDreamDoor.com - Greatest Music, Movie, and Book lists. [Online]. Available: https://digitaldreamdoor.com/. [Accessed: 19-Jun-2020].