Graph Neural Networks for Recommender Systems

Heidi C. Martinsaari, Maris Häusler
Institute of Computer Science, University of Tartu

***Abstract*—"Look, you may know these people!" "Other users were also looking for." "Customers also bought these products." Recommendation systems have become more and more popular on the internet platforms suggesting you friends, products, topics, etc. Recommendations can be beneficial for both - customers and companies. Such suggestions are generated using recommender systems. This paper is about building the recommender system from start to end, including introducing different systems from the near past. All the work is done for the Network Science course project and the recommender system we tried to create is based on the graph data. For this purpose we used Amazon product co-purchasing network metadata. The main focus of the project is the graph convolutional network. This is the heart of our recommender system which creates embeddings for simple Nearest Neighbours' algorithm to find top K similar products. In this paper we present the variations we used and the results that exceed the results of baseline models. We explain some issues which can appear when building the recommendation system.**

***Index Terms*—Graph, Graph Embedding, Graph Convolutional Network, Recommender System.**

I. Introduction

The problem nowadays is that it gets increasingly harder to orientate in the amount of information and products available on the web. It is easy for customers to leave the page when non-interesting products show up. For keeping customers in e-shop or media platforms it is necessary to show them items they are interested in. Recommender systems make the life easier for customers finding appropriate information or items. This increases the probability of making a purchase, thus increasing the profit of providers.

Recommender systems are very popular in the Industry. Many large data-driven companies like Amazon [9] and e-Bay use their AI based systems, and even social media platforms like Facebook and LinkedIn implement the recommendation systems for suggesting similar users or people that might be the acquaintances or friends of the user. Pinterest - a visual discovery engine - is using for their recommendation system a model named PinSAGE [6].

Our motivation behind this project was to understand the recommender systems in general. We decided to implement one recommendation system with some variants from the start to understand all the parts of the system. What data is needed? What data is useful? How to prepare the data for systems? We were interested in the different models and algorithms that have been used for recommendation systems. Nowadays, when there is much varying and sometimes complex data, complex models are also used to make the data digestible. Before deep neural networks many white box models were used. So, our goal was to dig into the evolution of the recommendation systems and have better understanding of them.

For the start, we needed a dataset with some product-customer information that is freely available. In this project we used dataset of Amazon product co-purchasing network metadata. This dataset contains metadata and review information about 548,552 products and 1,788,725 co-purchased products collected from year 1995 till 2006. We extracted different relationships between products and customers, analysing all of them and then choosing product-product relationship based on co-purchased products as the basis for our recommendation system.

Our plan was to build a recommender system that uses the embeddings created by the graph neural network and which uses some other machine learning algorithm to predict the recommendations based on the embeddings. For calculating the embeddings we used one shallow network and one deep network for comparison. Product features, such as product group, Amazon's salesrank and average rating, were used as the inputs for the neural networks. Our plan was to test different models and vary the input sets of the models. For the baseline we used similarity metrics of the products in the graph.

In addition to understanding the data, it is necessary to understand the architecture of the recommendation system's base model to determine which model is appropriate and fits with our graph. For this we needed to answer some questions. What are the measures of the graph and what is the graph type? Are we using homogeneous or heterogeneous graph? What features we know about the entities and edges? It is also important to think about what information we want to gain. Do we want to know the similar products of a product? Or maybe we are predicting the products for certain type of users? How big area of the graph structure we want to put into embeddings? These are all questions that are trade offs for the computational capability we have. It is obvious that for large datasets and complex models we need much resources for storage and computational power.

The tools for practical work were chosen based on their possibilities. The analysis of the graphs were done using Networkx module's property and visualization methods, but for neural networks the graphs were turned into Stellargraph objects. For simplicity we chose the models also from Stellargraph library. As we had product-product homogeneous graph, we decided to choose the Attri2Vec for shallow network, and GraphSAGE for deep network. We also were looking for a neural network for our bipartite graph with customer and product entities, like PinSAGE, but unluckily we did not find any suitable and working model for our data.

The recommender itself is based on Nearest Neighbour algorithm from Scikit-learn library. The algorithm takes in embeddings of products and gives out the top K similar products as for the recommendations in cross-selling task. If the customer is looking for some certain product, then other products are suggested to him or her next to it.

The result variations of our system are highly dependent on the embedding model used, as the recommendations are different between the models. On the other hand, different distance metrics play a smaller role here, having smaller differences in the order of the recommendations, while mostly recommending the same products. Compared to the baseline for which we used the similarity metrics calculated straight from the graph of Amazon similar products, the result differs greatly.

In the next section we cover the related works we were interested in and what we took as motivations. In the third section we describe the dataset and its preprocessing details. We show some statistical properties there. It is followed by the section of methodology where we describe our way of work in more detail. The fifth section covers the most important results section. Lastly, we conclude our work with some future thoughts.

II. Related work

The Amazon co-puchasing network dataset we decided to use has already been analysed from the recommendation systems perspective. Article [1] analyses the Amazon co-purchase network from the dynamic aspect. It is shown that patterns present in time t-1 are also present in time t with high probability. The authors suggest using Markov chain model on this type of data for implementing generalized recommendation system.

Totally graph-based and also white box model has been introduced in paper [2] which focuses on overlapping community detection algorithm based on cliques and finding top N nodes in those communities. Maximal cliques and then overlapping communities among them are identified. Top N nodes are either ranked by overlapping frequencies or by occurrence frequencies in the adjacent sub cliques that combine the discovered communities. Named techniques have also been applied on Amazon co-purchase network data.

The shallow network model named entity2rec [3][4] that is based on the node2vec [5], is tailored for Knowledge Graphs and measures user-item relatedness for top-N item recommendation. It works by creating property-specific subgraphs and finding their corresponding embeddings using node2vec. From there, similarity measures can be calculated. The advantage of entity2rec is that features can be clearly interpreted, and it works well on sparse datasets. In our implementation we also use a similar shallow network approach named attri2vec which name betrayds that it uses node attributes to create their embeddings.

Based on our customer-product network dataset would have been most applicable to PinSAGE like algorithm. PinSage [6] is a graph convolutional network which combines random walks and graph convolutions to generate embeddings of nodes involving the graph structure and node feature information. Using the neighbour information and many convolutional layers makes the model very powerful. Moreover, the PinSAGE algorithm is made for large scale graphs, introducing new training techniques and algorithmic innovations (random walks, importance pooling). PinSAGE similarly outputs embeddings which are then used in neareast neighbour algorithm like we implemented in our project.

Little bit same flavored algorithm is descibed in paper [7] but this graph convolutional network is built for knowledge graphs. Knowledge graphs are heterogeneous graphs which have different types of entities and edges. KGCNs are even more powerful, being able to capture both - the high-order structure and semantic information of the Knowledge Graph. Having ability to act easily in complex graph dataset means highly intelligent algorithm and techniques which are very difficult to copy. In addition, lightweightness is not the only criteria.

Another important aspect is explainability, and also the fact that data given is often sparse. The paper [8] is precisely pointing out that key function of GCNs is neighbourhood aggregation mechanism while the real-world user-item graphs are often incomplete and noisy. That causes low performance for graph embeddings as the aggregations are used recurrently. Authors' goal was to alleviate this problem using already existing models and implementing clever techniques on them.

III. Dataset

As the purpose of the project was to create a **recommender system**, we needed some customer and product data. We found the Amazon product co-purchasing network metadata a good initial dataset. This open data was downloaded from the page of Stanford University https://snap.stanford.edu/data/amazon-meta.html. This dataset provides 548,552 products with 1,788,725 similar product-product connections, 7,781,990 customer reviews and ratings for products by date and much more. The products are divided into product groups from which we selected the biggest groups: books, music, video and DVD. This dataset is from year 1995 till 2006. So, the data is quite old, but very big and good enough for a toy dataset. See the field descriptions in the table I.

The initial dataset was given as a text file which we read into Pandas dataframe with specific function written by ourselves. It was then cleaned more and divided into different dataframes. Several analysis were done on the dataset and based on that we have applied some filters for selecting the most interesting part of the data. First, we selected only products from four product groups. Secondly, we selected only those product-product connections (similar products by Amazon) where both products have additional data like title, product group, salesrank, etc. Named filters give us the undirected homogenous graph of product-product relationships (A links) (figure 1) which properties can be seen in the second column of the table II.

Customer reviews on the products gave the relationships between customers and products (B links) (figure 1). These links were used to create the bipartite graph with nodes of

TABLE I
DESCRIPTION OF PROVIDED DATA

Field	Description
Id	Product ID (numbers from 0 to 548551)
ASIN	Amazon Standard Identification Number
Title	Name or title of the product
Group	Product group (Book, DVD, Video, Music, etc)
Salesrank	Amazon Salesrank
Similar	ASIN of co-purchased products
Categories	Location in product category hierarchy to which the product belongs
Reviews	Product review information: date, user id, rating, total number of votes on the review, total number of helpfulness votes

TABLE II
MEASURES OF PRODUCT-PRODUCT GRAPH AND CUSTOMER-PRODUCT BIPARTITE GRAPH

Measure	Product-Product	Customer-Product
No of nodes	296,458	639,793
No of edges	236,429	410480
Transitivity	0.0071	0.0
Average clustering	0.0072	0.0
Edge density	5.3803e-06	2.0056e-06
Average degree	1.5950	1.2832
Total triangles	3372	0
No of connected components	61,292	229,313

customers and nodes of products. As a result of filters we are left with fewer nodes and edges also in bipartite graph as seen in the last column of table II.

There was also one more product-product graph made from the bipartite graph (C links) (figure 1). The link connects the products of the same customer. This graph became too large for our computational capability. But it's worth to consider using it in the future as such connections include much information about the users and their preferences. In the end, we used the product-product graph with Amazon similarities (A links).

So, we used the Amazon's given similarity (same market

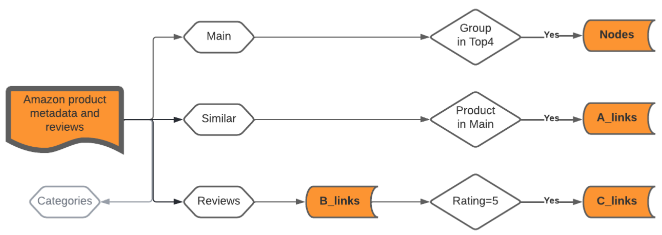


Fig. 1. Dataset Preparation

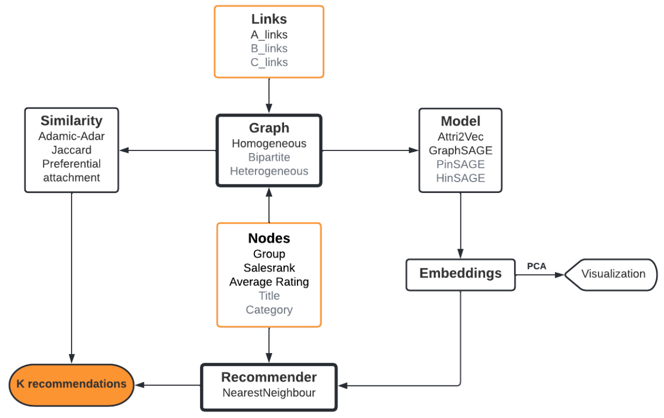


Fig. 2. Methodology flowchart

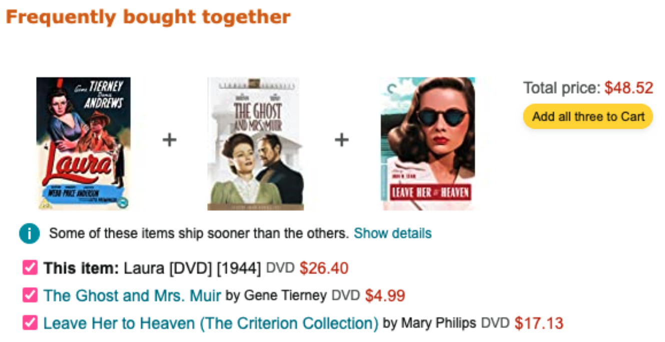
basket products) links (A links), and additionally some node data. We limited ourselves with numerical and categorical data. We used the product group which we encoded, Amazon's salesrank and the average rating. There was also a possibility to use text processing for product titles or embed the categories. The categories were given as hierarchical tree and one product could belong into multiple category branches.

The code of fetching the data from source file and data cleansing is provided in DataPreparation.ipynb with some additional python files with necessary methods. Strategic intermediate results are saved into csv files. So, it is not mandatory to execute all the code for the final steps. As usual, preprocessing the data is the most time consuming part for data scientists. All needed files can be found in our Github repository. Readme file guides the user through the necessary steps.

IV. METHODOLOGY

The main goal was to create a recommender system based on a combination of a graph neural network and the classical machine learning algorithm. Our methodology for achieving it is as in figures 1 and 2. Getting Nodes and Links from the dataset is decribed in details in previous section of Dataset. In figure 2 we may see again that for nodes (products) we have the attributes group, salesrank, average rating, title and category. And there are three different possible links we could use.

All the links were used for creating three graphs and made the standard graph analysis using Networkx module. Basic measures, such as number of nodes and edges, transivity, density, number of connected components, etc were calculated (table II). The most popular products were calculated using centrality measures. Knowing the popular products can help to understand why some products appear more frequently in recommendations than others. The analysis is shown in the notebook GraphAnalysis.ipynb. As a result of the graph property calculations, we discovered that our graphs are huge, but still sparse, which proves the complexity of the graphs.


Frequently bought together

Total price: $48.52

Add all three to Cart

Some of these items ship sooner than the others. Show details

☑ **This item:** Laura [DVD] [1944] DVD $26.40

☑ The Ghost and Mrs. Muir by Gene Tierney DVD $4.99

☑ Leave Her to Heaven (The Criterion Collection) by Mary Philips DVD $17.13

Fig. 3. Frequently bought together in Amazon

In this paper we created recommendation system using graph of similar products (A links) directly given in the original Amazon dataset. From node features we selected only the product group, Amazon's salesrank and average rating for simplicity. The created undirected unweighted homogeneous graph was used to create embeddings using Stellagraph library. We implemented two different models for embeddings: Attri2Vec and GraphSAGE to get comparable results. The embeddings calculated based on the node features and/or neighbourhood information were the input to our ”recommender” that used the Nearest Neighbour algorithm and different distance metrics to find similar products. Results of the recommender were also compared to the similar products found directly from the graph using different similarity measures. The result of our project is a recommender system that recommends top K products based on an example product.

V. Results

Our problem set-up was totally unsupervised cross-selling task. We found the recommendations for the products: a customer is interested in certain product and the system recommends him or her K other products. We were provided some similar products (frequently bought together), but these connections gave us quite a sparse graph. Based on this product-product graph we calculated the similarities between products, but these results were much different from our results.

Our recommender system used six different approaches: two different networks for creating embeddings and three different metrics for finding nearest neighbours of embeddings. In the first two columns of figure 4 we can see the K=5 recommendations for each approach for the DVD “Laura”. In the first column are shown the recommendations based on the GraphSAGE embeddings and in the second column we can see the recommendations calculated from Attri2vec embeddings.

For both embeddings we used the Nearest Neighbour algorithm using 3 different distance metrics. In upper boxes there are results for Minkowski metric, in center Manhattan distance and in bottom Canberra distance, which are calculated as follows:

$$d_{manhattan} = \Sigma\big(|x-y|\big) \tag{1}$$

$$\tag{2}$$

$$d_{minkowski} = \Sigma\big(w \cdot |x-y|^{p}\big)^{\frac{1}{p}} \tag{3}$$

$$\tag{4}$$

$$d_{canberra} = \Sigma\frac{|x-y|}{|x|+|y|} \tag{5}$$

We may notice that the first recommendation for all approaches is the same: DVD “Michiganfest 2002”. Embeddings from one model provide almost the same results for all distance metrics. GraphSAGE has three common results over all metrics and Attri2vec four common for all and five common for two metrics. The product DVD “Laura” was selected from the set of popular products which had the biggest pagerank, eigenvector, Katz, closeness and degree centrality. When we selected the products randomly there was an interesting outcome. Attri2vec resulted for all distance metrics in exactly the same recommendations for many products. But GraphSAGE found the different products in different cases.

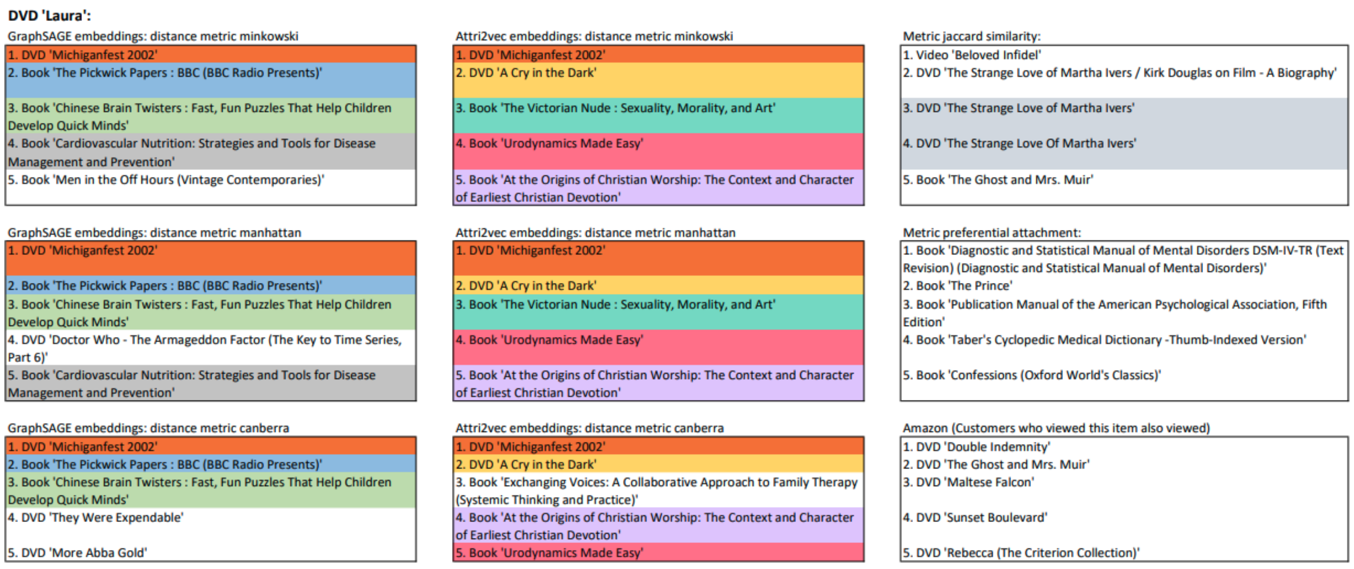
We can conclude that Attri2vec which is shallow network and used only the attributes of the products themselves does not give much information about the related products, but the GraphSAGE which uses also the neighbourhood information is much more smarter. Attri2vec would have performed better if we had given it more input attributes.

The similarity measures as Jaccard similarity and preferential attachment were used as the baselines measures for recommendations. Our system searches the smallest distance. But the similarity values are opposite to distance. The recommendations based on similarities can be seen in the third column of the figure 4. There are also the recommendations from the Amazon website. Firstly, similarity measures give totally different results that we have got with our six approaches. But we may notice that recommendation of Jaccard similarity is similar with the recommendation from Amazon website. Book ”The Ghost and Mrs. Muir” is similar to DVD ”The Ghost and Mrs. Muir”.

It may seem like a big failure at the first glance, but we must keep in mind that the similarity or difference (distance meaning) might measure different aspects. For example the Attri2vec model used the node numerical features for calculating the embeddings. GraphSAGE however took the neighbourhood into account. It is also important to note, that we selected quite lightweight neighbourhood for computational purposes, but the radius of the neighbourhood should not be too big as also discovered in papers [6] and [7]. Too big neighbourhood can lead the system to predict too different items. The best ground truth would be putting the systems into practice like PinSAGE [6].

VI. Conclusion

Recommender systems have become very popular in industry as these are found beneficial especially in cross-selling purposes. Recommender systems use different models from



DVD 'Laura':

GraphSAGE embeddings: distance metric minkowski
1. DVD 'Michiganfest 2002'
2. Book 'The Pickwick Papers : BBC (BBC Radio Presents)'
3. Book 'Chinese Brain Twisters : Fast, Fun Puzzles That Help Children Develop Quick Minds'
4. Book 'Cardiovascular Nutrition: Strategies and Tools for Disease Management and Prevention'
5. Book 'Men in the Off Hours (Vintage Contemporaries)'

Attri2vec embeddings: distance metric minkowski
1. DVD 'Michiganfest 2002'
2. DVD 'A Cry in the Dark'
3. Book 'The Victorian Nude : Sexuality, Morality, and Art'
4. Book 'Urodynamics Made Easy'
5. Book 'At the Origins of Christian Worship: The Context and Character of Earliest Christian Devotion'

Metric jaccard similarity:
1. Video 'Beloved Infidel'
2. DVD 'The Strange Love of Martha Ivers / Kirk Douglas on Film - A Biography'
3. DVD 'The Strange Love of Martha Ivers'
4. DVD 'The Strange Love Of Martha Ivers'
5. Book 'The Ghost and Mrs. Muir'

GraphSAGE embeddings: distance metric manhattan
1. DVD 'Michiganfest 2002'
2. Book 'The Pickwick Papers : BBC (BBC Radio Presents)'
3. Book 'Chinese Brain Twisters : Fast, Fun Puzzles That Help Children Develop Quick Minds'
4. DVD 'Doctor Who - The Armageddon Factor (The Key to Time Series, Part 6)'
5. Book 'Cardiovascular Nutrition: Strategies and Tools for Disease Management and Prevention'

Attri2vec embeddings: distance metric manhattan
1. DVD 'Michiganfest 2002'
2. DVD 'A Cry in the Dark'
3. Book 'The Victorian Nude : Sexuality, Morality, and Art'
4. Book 'Urodynamics Made Easy'
5. Book 'At the Origins of Christian Worship: The Context and Character of Earliest Christian Devotion'

Metric preferential attachment:
1. Book 'Diagnostic and Statistical Manual of Mental Disorders DSM-IV-TR (Text Revision) (Diagnostic and Statistical Manual of Mental Disorders)'
2. Book 'The Prince'
3. Book 'Publication Manual of the American Psychological Association, Fifth Edition'
4. Book 'Taber's Cyclopedic Medical Dictionary -Thumb-Indexed Version'
5. Book 'Confessions (Oxford World's Classics)'

GraphSAGE embeddings: distance metric canberra
1. DVD 'Michiganfest 2002'
2. Book 'The Pickwick Papers : BBC (BBC Radio Presents)'
3. Book 'Chinese Brain Twisters : Fast, Fun Puzzles That Help Children Develop Quick Minds'
4. DVD 'They Were Expendable'
5. DVD 'More Abba Gold'

Attri2vec embeddings: distance metric canberra
1. DVD 'Michiganfest 2002'
2. DVD 'A Cry in the Dark'
3. Book 'Exchanging Voices: A Collaborative Approach to Family Therapy (Systemic Thinking and Practice)'
4. Book 'At the Origins of Christian Worship: The Context and Character of Earliest Christian Devotion'
5. Book 'Urodynamics Made Easy'

Amazon (Customers who viewed this item also viewed)
1. DVD 'Double Indemnity'
2. DVD 'The Ghost and Mrs. Muir'
3. DVD 'Maltese Falcon'
4. DVD 'Sunset Boulevard'
5. DVD 'Rebecca (The Criterion Collection)'

Fig. 4. Recommendations for DVD "Laura"

white box to black box and from shallow to deep networks. These models have to be capable of handling complex and large datasets and being explainable at the same time.

We created the recommender system using shallow network and deep network followed by classical machine learning algorithm nearest neighbour. We varied the nearest neighbour with the help of three different distance metrics. For the result we landed into six approaches which we implemented on the homogeneous graph of the similar Amazon products using few attributes of the products.

The results of our system exceed the results of baseline for which we used graph similarity measures. The recommendations differ when using different embedding models but have only slight variation between the different distance metrics. We concluded, that it is very hard to decide or measure the goodness of the recommender without putting it to the practice and doing usual A/B testing as the problem of recommendations is totally unsupervised task.

There came up many ideas how to improve our work in many aspects. Firstly, we would like to add more data to the model. We did not utilize the textual information about the product title and the category tree, which could have been used to create additional embedding input for our model. Secondly, we would have wanted to build PinSAGE-like model for bipartite graph (B links). Third idea was to implement the model on product-product links (C links) gained from bipartite graph, but utilizing this we would need more computational power. Some ideas may grow into master thesis ideas.

VII. Github

You may find our GitHub Repository at the link https://github.com/h31d1/Recommender. The **Readme File** guides how to find the necessary information from the notebooks and code.

References

[1] P. Basuchowdhuri, M. K. Shekhawat and S. K. Saha, "Analysis of Product Purchase Patterns in a Co-Purchase Network," *2014 Fourth International Conference of Emerging Applications of Information Technology*, 2014, pp. 355-360, doi: 10.1109/EAIT.2014.11

[2] M. Farrag, L. Fangary, M. Nasr, and C. Salama. "Rank the Top-N Products in Co-Purchasing Network through Discovering Overlapping Communities Using (LC-BDL) Algorithm," *Journal of Multidisciplinary Engineering Science and Technology (JMEST)*, 2017, vol. 4, pp. 8153-8166

[3] E. Palumbo, D. Monti, G. Rizzo, R. Troncy, and R. Baralis, "entity2rec: Property-specific knowledge graph embeddings for item recommendation," *Expert Systems with Applications*, 2020, vol. 151, doi: 10.1016/j.eswa.2020.113235

[4] E. Palumbo, G. Rizzo, and R. Troncy, "entity2rec: Learning User-Item Relatedness from Knowledge Graphs for Top-N Item Recommendation," *In Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys '17). Association for Computing Machinery*, pp. 32–36, doi: 10.1145/3109859.3109889

[5] A. Grover, J. Leskovec, "node2vec: Scalable Feature Learning for Networks," *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016, doi: 10.48550/ARXIV.1607.00653

[6] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph Convolutional Neural Networks for Web-Scale Recommender Systems," *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, doi: 10.1145/2F3219819.3219890

[7] H. Wang, M. Zhao, X. Xie, W. Li, and M. Guo. "Knowledge Graph Convolutional Networks for Recommender Systems," *In The World Wide Web Conference (WWW '19). Association for Computing Machinery*, 2019, pp. 3307–3313, doi: 10.1145/3308558.3313417

[8] H. Chen, L. Wang, Y. Lin, C.-C. M. Yeh, F. Wang, and H. Yang. "Structured Graph Convolutional Networks with Stochastic Masks for Recommender Systems," *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. Association for Computing Machinery*, 2021, pp. 614–623, doi: 10.1145/3404835.3462868

[9] A. Krysik, "Amazon's Product Recommendation System In 2021: How Does The Algorithm Of The eCommerce Giant Work?" *https://recostream.com/blog/amazon-recommendation-system*

[10] "Github: StellarGraph Machine Learning Library," *https://github.com/stellargraph/stellargraph*