E-Commerce Product Clustering

Vanshita Garg Etiksha Jain Deepanshu Rohilla

November 14, 2023

Abstract

Recommendation systems have become the backbone of the e-commerce platforms like Myntra, Amazon etc. In this work, we work with a dataset of products and users from an e-commerce site and try different clustering techniques to see which works the best. The work focuses on the use of Graph Neural Networks to create hidden representations for products and see how these embeddings can be used in addition/in replacement of the existing raw features.

1 Introduction

Clustering e-commerce products can help us gain useful insights about product demand, which users prefer which type of products and so on. Sometimes only object features may not be sufficient to describe a product or its relations with other entities. We therefore explore the idea of using graph neural networks to create embeddings/ meaningful representations for every product which can incorporate in itself the product's interaction with different users, other products, etc. Such embeddings can be obtained by modeling the problem as either a link prediction task or a node classification task. Further on, we use clustering methods like K-means, DBScan and compare the clusters obtained by using raw features and embeddings obtained from various graph learning algorithms.

2 Dataset Analysis

Two CSV files have been provided :

1. One contains product IDs along with the price and vertical of every product
2. The second file contains the user ID, the product ID the user has clicked on, and the count of the number of times the product has been clicked upon

2.1 Dataset Visualization

We do an inner join on the two tables to make one bigger table that can be used for further computation. The dataset statistics are as follows:

Table 1: Count.csv

Product ID	Price	Cms Vertical
ACCG2DHTPQRPV4ZP	3	36ee6d3cd4e30b41e7b6aba58641e09a
ACCG57P5RGWJZARH	13	36ee6d3cd4e30b41e7b6aba58641e09a
MOBFUSBNAZGY7HQU	16	36ee6d3cd4e30b41e7b6aba58641e09a
ACCFHYJ8ZU9WDHDW	1	36ee6d3cd4e30b41e7b6aba58641e09a
ACNGFVMJ5ZKZVFDS	2	36ee6d3cd4e30b41e7b6aba58641e09a

Table 2: Products.csv

Product ID	Count	Account ID
9780000018885	690.000000	book
9780002245708	2665.000000	book
9780007154692	333.000000	book
9780007175185	192.000000	book
9780007216161	395.000000	book

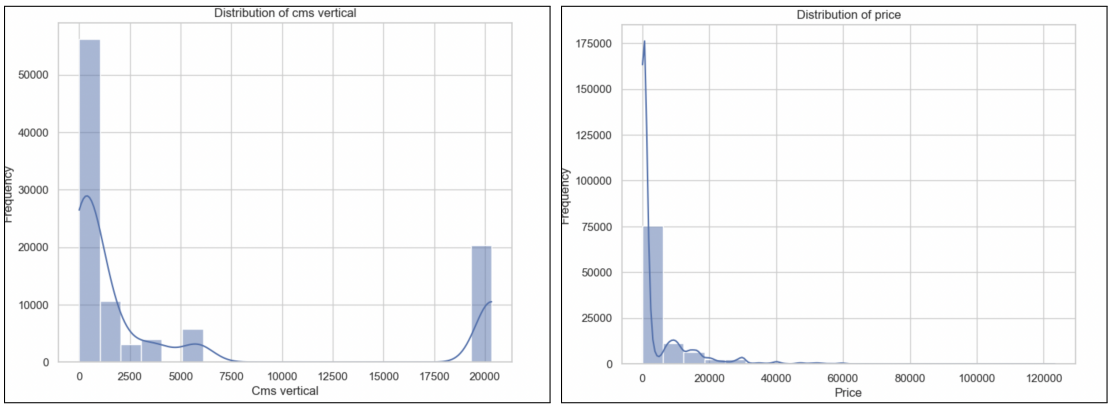
Table 3: Dataset Stats

Unique Product IDs	16 million
Unique Account IDs	2 million
No. of Verticals	2K
Mean views of users	1.5
Maximum views by user	2K

2.2 Data Cleaning

Before Cleaning

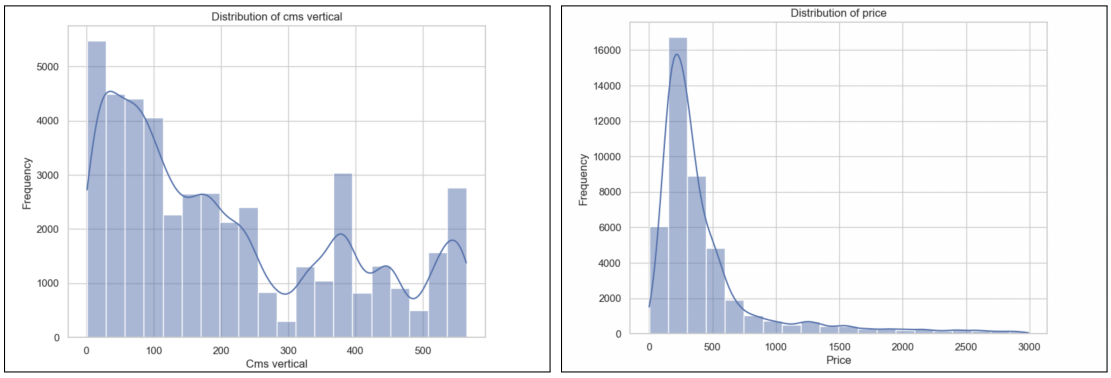
Initially, the data is highly positively skewed and distributed irregularly. We use frequency encoding for cms vertical. We can see that the distribution of frequencies and price is very sparse.



After Cleaning

We therefore perform data cleaning to remove outliers and also perform frequency encod-

ing for the categorical variable- product vertical.



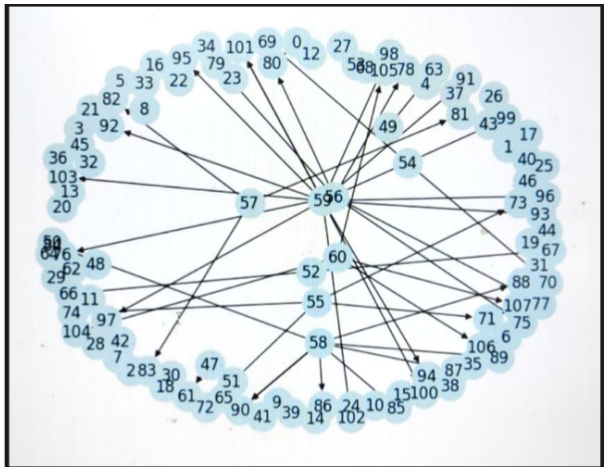
3 Problem Modeling

The problem is modeled as a graph that has two types of nodes:

1. User Nodes - Features: [Mean user views, total views]
2. Product Nodes - Features: [Price, Vertical]

We create a bipartite graph such that an edge exists between a user node and a product node if the user has clicked on the product at least once. Every node has its own features as well. The graph is created using the Deep Graph library (DGL) and visualized using networkx.

Figure 1: Graph



After the graph is created, we perform the following steps:

1. Obtain node embeddings by training a GNN. We use two neural networks:
   (a) Graph Autoencoder

(b) Graph Attention Network

2. Perform product clustering on the obtained embeddings. We use the following algorithms to obtain clusters

   (a) K-means

   (b) DB-scan

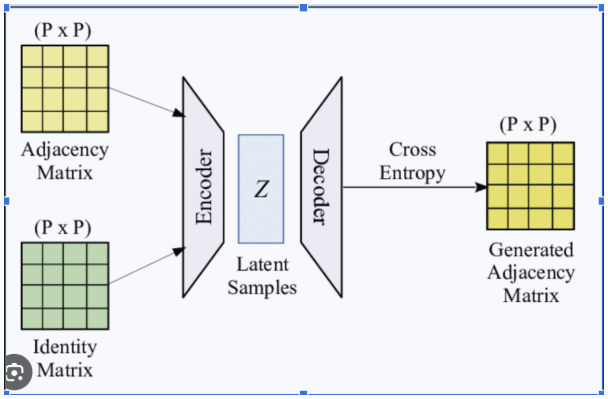
The clusters obtained from all approaches are then compared with raw features.

4 Methods

4.1 Graph AutoEncoder

The traditional autoencoder is a neural network that contains an encoder and a decoder. The encoder takes a data point X as input and converts it to a lower-dimensional representation (embedding) Z. The decoder takes the lower-dimensional representation Z and returns a reconstruction of the original input X-hat that looks like the input X. The quality of the embedding determines the quality of the output X-hat.

Figure 2: GAE Algorithm

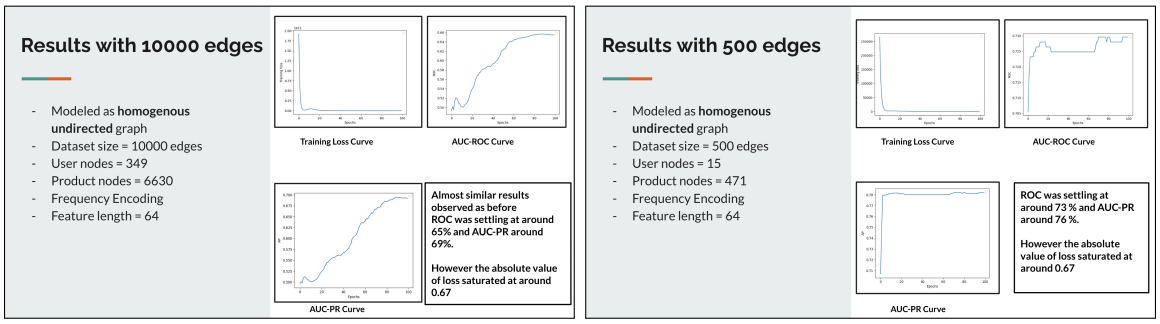


Implementation We implement the code of the reference paper[3]

- The autoencoder architecture contains two Graph Convolution layers and then an InnerProduct Decoder
- This model treated the graph as **homogeneous**
- We extended the feature-length to 64 using padding
- We computed the AUC-ROC and AUC-PR with 500 and 10K edges in the following settings:

  1. Using raw node features
  2. Using features obtained after Collaborative filtering [1]

Results
The results obtained are just satisfactory with an AUC-ROC of 73% and AUC-PR pf 76%. The possible reasons could be the small amount of dataset processed due to computational limitation, less number of original features and modeling the graph as homogenous.



Raw features used as initial node embeddings

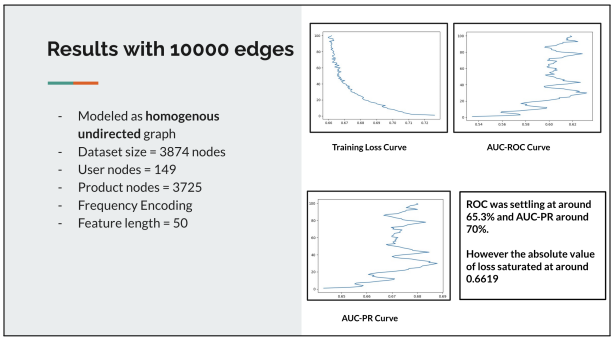


Figure 3: Collaborative filtering features used as initial node embeddings

4.2 Link Prediction using Graph Attention Network

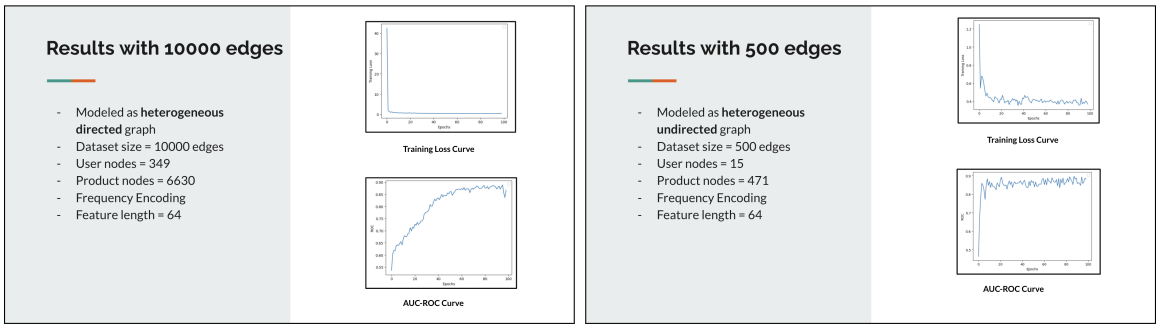
In our next approach, we model the problem as a link prediction task, where the loss function takes the original adjacency matrix for computation. We use **graph attention networks** for this purpose.

Implementation We implement the code of the reference paper[4]

- The GAT architecture contains two GAT convolution layers followed by a linear and an embedding layer
- This model treated the graph as **heterogenous**
- We extended the feature-length to 64 using padding
- We computed the AUC-ROC and AUC-PR with 500 and 10K edges using raw features

Results
We get better results as compared to our previous approach, with an AUC-ROC of around 85% since this method models the graph as heterogeneous. The results obtained are as

follows:



Raw Features used as initial node embeddings

4.3 Result Analysis

The above results show that modeling the graph as a **heterogeneous** graph is more intuitive and beneficial since we have two different types of nodes- users and products. Link Prediction is thus giving better results. The above results were computed on very little dataset and the same process was repeated after the data was cleaned. This improved our results but only marginally, taking ROC to around 78.9% in case of the autoencoder approach. At the same time, the results also suggest that taking the features obtained from the collaborative filtering approach as initial node embedding did not prove to be very useful as the Alternating Least Squares approach used to create them only takes into account the interaction between a user and product based on the count of clicks, and does not take into account any other explicitly given product/user features.

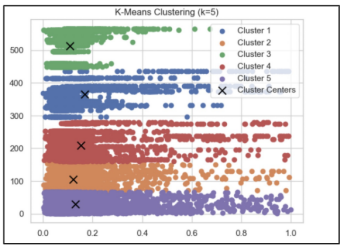
5 Clustering

Clustering is employed to find out the hidden features of the data distribution and how we can group the data based on several aspects. In our case, we employ 2 clustering techniques and compare both of them in three scenarios:

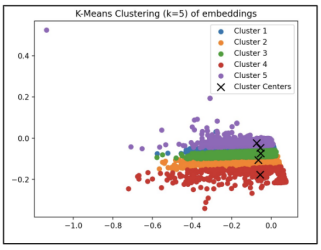
1. Using the raw features of products for clustering
2. Using the embeddings obtained from the AutoEncoder network
3. Using embeddings obtained from GAT as features of our products for clustering.

5.1 K-means Clustering

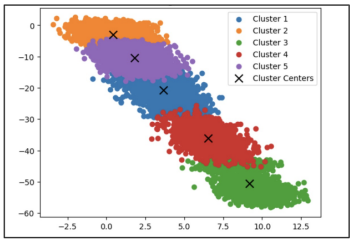
[7] K-means is a centroid-based or partition-based clustering algorithm. This algorithm partitions all the points in the sample space into K groups of similarity. The similarity is usually measured using Euclidean Distance.



Using raw node features
X-axis: Price ; Y-axis : Vertical



Using embeddings obtained from Graph Autoencoder

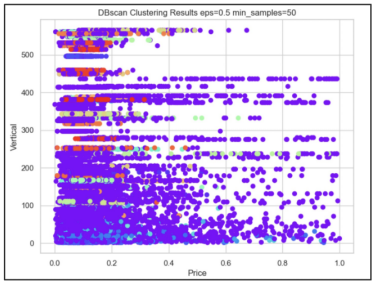


Using embeddings obtained from Graph Attention Network

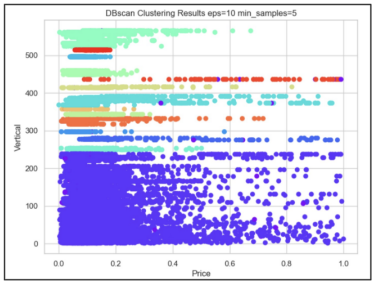
5.2 DBScan Clustering

DBScan is a density-based clustering algorithm. The key fact of this algorithm is that the neighborhood of each point in a cluster that is within a given radius (R) must have a minimum number of points (M)[6].
M is usually taken close to the dimensionality of data. For our implementation, M=64, and R is taken as the K-distance.
The results of DBScan clustering depend largely on the initialization of parameters- epsilon, min samples. We therefore experiment with different set of parameters.

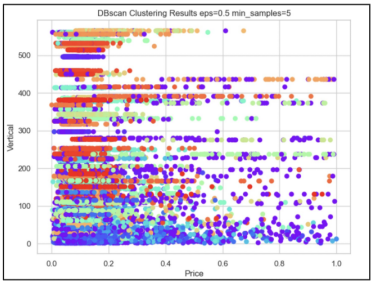
5.3 DBScan Clustering with Raw Features



Poor Clustering

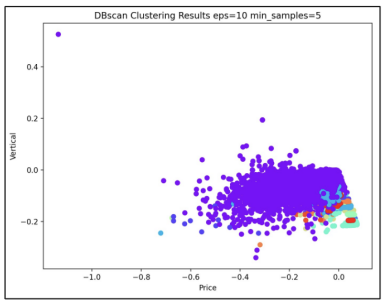


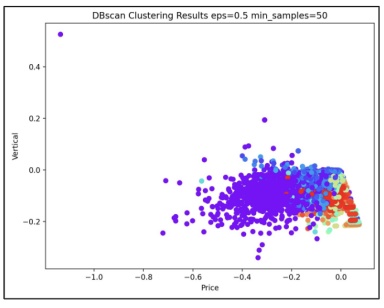
Epsilon changed; better results

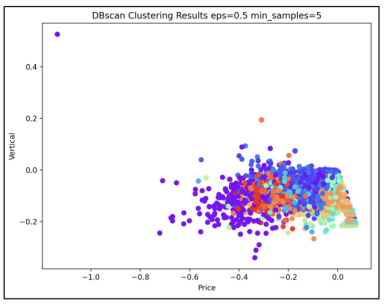


Epsilon reduced; discrete clusters

5.4 DBScan Clustering of Embeddings







5.5 Clustering Results Analysis

- Both raw features, as well as embeddings, give rise to about 5 clusters of products
- In the case of raw features, irrespective of price, products are clustered on the basis of their vertical
- In the case of embeddings, DBScan is not able to do clear clustering due to the high dimensionality of embeddings
- It is not clear whether clusters are based on price or vertical It might be the case that user features are affecting these clusters, but that is unclear here

6 Conclusion

Based on the analysis done so far, we reach the following conclusions:

- Inherently the products are clustered on the basis of the vertical to which they belong
- K-means clustering proves to be a good technique here, giving consistent results with raw features as well as embeddings
- The product embeddings obtained from link prediction modeling have similar cluster structure and centroid coordinates suggesting that the vertical of a product has a strong influence in its final embedding and is not much dependent on the user features we had considered
- DBScan doesn't work well for our data due to 'curse of high dimensionality ' along with problematic parameter tuning
- This might also mean that there might be a lack of good features in the data, pointing to the need for data enrichment

6.1 Future work

- The work can be extended to fit a larger dataset by utilizing HPC resources
- More data can be scraped/acquired along with better features to improve training accuracy
- Techniques like t-SNE visualization etc can be used further for better analysis [5]

7 Code

All the code implementations can be found here.

References

[1] Alternating Least Square for Implicit Dataset with code — by Himanshu Kriplani — Towards Data Science

[2] ALS Implicit Collaborative Filtering — by Victor — Rn Engineering — Medium

[3] [1611.07308] Variational Graph Auto-Encoders (arxiv.org)

[4] Medium - PyG: Link Prediction on Heterogeneous Graphs

[5] Automatic feature extraction with t-SNE — by Gonçalo Rodrigues — Jungle Book — Medium

[6] How to determine epsilon and MinPts parameters of DBSCAN clustering - Amir Masoud Sefidian - Sefidian Academy

[7] K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks — by Imad Dabbura — Towards Data Science