Graph Neural Network with Social Media E-Commerce*

Sonia Zeng (1006683082), Bonnie Luo (1006762905), Sherry Li (1006922332)

March 27, 2023

1 Problem Statement

Social media has become an increasingly essential tool for e-commerce brands, where they can understand relationships between different products, develop products catering to new trends, and eventually gain a competitive advantage in the market [1]. Traditionally, companies rely on the experience and intuition of industrial analysts or influencers for trend prediction. While these methods may be effective, their performance in trend identification is poorer than data-driven methods using statistical models or machine learning [2].

This project aims to take full advantage of social media posts about products, from both companies and users, to analyze potential connections between *hashtags* using Graph Neural Network (GNN), in which way to predict future product and social media trends. By analyzing the connection between hashtags used in the existing posts and other hashtags, we aim to help e-commerce businesses identify new directions for innovations, indicated by those promising new hashtags.

2 Method Overview

2.1 Data Collection

The data for this project was obtained from Instagram, a social media platform that is popular among younger users who frequently post user experiences on the platform [3]. To obtain the initial set of hashtags, we scraped the hashtags used by a selected brand account as the source of our database. We then collected other related hashtags in our domain of interest to form a directed connected graph, where nodes represent hashtags, and edges represent connections between hashtags (Fig. 3).

2.2 Link Prediction

To predict potential connections between hashtags, we implemented *negative sampling* by treating existing edges as positive examples and sampling a set of non-existing samples as negative examples (Fig. 2). We then divided the subgraphs of positive and negative examples into training, validation, and testing sets. These sets were fed into a GNN for further link prediction.

The GNN was selected as the primary link prediction algorithm due to its ability to analyze graph-structural data, which is the natural representation of data in an e-commerce social network. Unlike other ML frameworks, GNN naturally integrates node information as well as topological

*All scripts of this project can be found in this repository.

structure, making it more powerful in learning graph representations [4]. In this way, the existence of an edge between two arbitrary nodes in a graph can be predicted.

Furthermore, nodes represent Instagram hashtags, while *node features* and *edge features* can be used to indicate the occurrence frequency of each hashtag, and the co-occurrence frequency of a particular pair of hashtags. These features can thus be passed down to the model for the computation of pair-wise scores as discussed in the Section 5.

3 Prior Work

3.1 GNN

The primary prior work used in the project is GNNs, which uses graphs, a non-Euclidean data structure for machine learning [5], for node classification, link prediction, and clustering tasks.

A graph is denoted as $G = \{V, E\}$, where $|V| = N$ is the number of nodes in the graph, and $|E| = N^e$ is the total number of edges with e representing the number of edges for one node, assuming all nodes have the same degree. There are several types of graphs:

- **Directed/Undirected graphs**: In a directed graph, the connections between the nodes have a specific direction, whereas in an undirected graph, the connections are bidirectional. In this sense, the direction of connections indicates causal relationship between variables, or the order of events, while undirected connections contain less information.
- **Homogeneous/Heterogeneous graphs**: A homogeneous graph is a graph where all nodes belong to the same type or category, while the nodes in a heterogeneous graph are different. Homogeneous graphs are often used in the context of graph-based classification or regression tasks, where the goal is to predict a target variable based on the attributes of the nodes and edges. In contrast, heterogeneous graphs are often used in the context of graph-based recommendation systems, where the goal is to recommend items or products to users-based on their preferences or past behaviour.

At the interim stage of the project, we decided to experiment with directed homogeneous graph to represent the relations of all hashtags in our dataset. However, the other types of graphs can also be explored in the later stage whose performance can be compared.

The GraphSAGE convolution can also be used for our project, which is a general *inductive* framework that generates node embeddings for previously unseen data by sampling and aggregating neighbouring node's features [6]. Compared to other embedding methods using matrix factorization, GraphSAGE utilizes nodes features (e.g., node degree, node centrality, node eigenvector similarity, etc.) to generate inductive embeddings for new nodes with a set of aggregation functions which are capable of aggregating information from nodes located at various distances or depths from the given node (Fig. 1). As for the aggregation calculations, three methods have been proposed:

- Mean aggregator: take the mean value of the embeddings of the node and its neighbors
- LSTM aggregator: permutes the nodes randomly as the nodes are not in sequence
- Pooling aggregator: an element-wise pooling function on the neighboring nodes.

They have also designed an unsupervised loss function for GraphSAGE, which allows it to be trained without the requirement of task-specific supervision, following prior research on generating node embeddings.

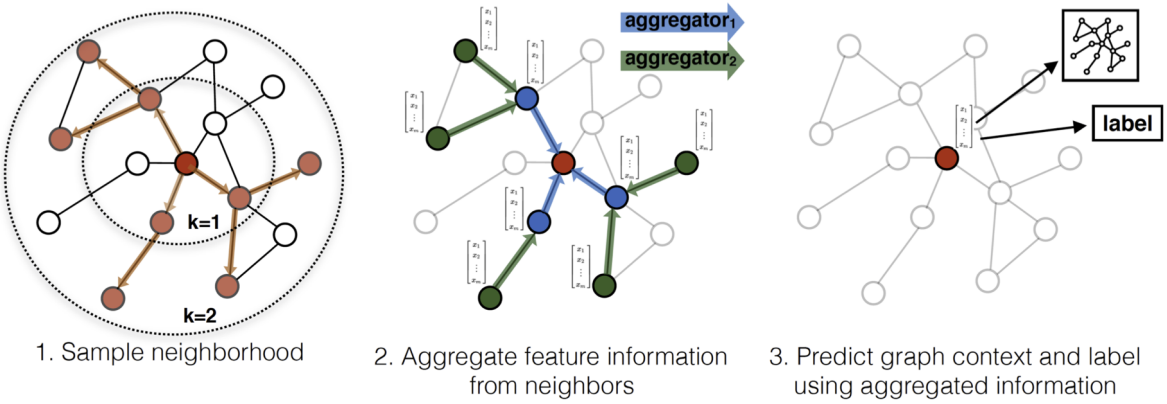


Figure 1: GraphSAGE's approach to sample and aggregate neighbor nodes' features with 3 steps

The embedding generation algorithm used by GraphSAGE is introduced in the Algorithm below as in the original paper [6]. In a graph of $\mathcal{G}(\mathcal{V}, \mathcal{E})$, all nodes have their corresponding features $\mathbf{x}_v$ as the input for the GraphSAGE algorithm. By sampling a fixed size of neighbour nodes, different but uniform samples are obtained per batch and aggregated to a single vector $\mathbf{h}^{k-1}_{\mathcal{N}(u)}$. In this step, AGGREGATE $_k, \forall k \in \{1, \ldots, K\}$ stands for the K aggregator functions which aggregate information from neighbours.

After this, the GraphSAGE algorithm concatenates the current node's vector representation $\mathbf{h}^{k-1}_v$ with that of its neighbours $\mathbf{h}^{k-1}_{\mathcal{N}(u)}$, before computing a dot product with a set of weight matrices $\mathbf{W}^k, \forall k \in \{1, \ldots, K\}$, which propagates information between different layers of the model, and passing to a nonlinear activation function σ. Consequently, nodes can collect information from further nodes and obtain a label with all aggregated features.

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}^0_v \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}^k_{\mathcal{N}(v)} \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}^{k-1}_u, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}^k_v \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}^{k-1}_v, \mathbf{h}^k_{\mathcal{N}(v)})\right)$
  end
  $\mathbf{h}^k_v \leftarrow \mathbf{h}^k_v / \|\mathbf{h}^k_v\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}^K_v, \forall v \in \mathcal{V}$

3.2 Apify API

We can construct a dataset containing hashtags relevant to our domain via Apify API, which returns a specific number of posts with the targeted hashtag in the JSON or CSV formats. The Apify API can also be accessed using python by import the `apify-client` PyPI package. With our own scripts of obtaining and adjusting the dataset, Apify can be well integrated into our pipeline, while ensuring the dataset is inclusive and general enough.

3.3 InstaCrawlR

InstaCrawlR is a GitHub repository of four scripts, including jsonReader, hashtagExtractor, graphCreator, and g2gephi. By using these scripts, we can

- download the latest posts for any designated hashtag,
- export files that shows post ID, URL, number of likes, account ID, post data and location,
- automatically extract associated hashtags and download images from the posts,
- collect related hashtags and compute frequency,
- generate and export a graph that displays the connections between primary and related hashtags.

For example, if we want to evaluate a company's connection with its customers on Instagram, the primary input for InstaCrawlR will be all hashtags that this company's brands have used. After this, a social network analysis will be run to scrape all other related hashtags and build up the hashtag database. However, the functionality of scraping posts are now no longer available, as a result of Instagram's restriction on API use to prevent spamming behaviour and data exploitation [7]. Thus, we decided to use Apify API to reach the same purpose.

4 Data Processing

4.1 Scraping Posts by Hashtags

Hashtag is a very convenient way for customers and brands to label products on Instagram, infer user preferences user and product relationships, and create customer groupings for a product. This project targets at the cosmetics domain since it is one of the industries that use Instagram heavily in propagating and advertising products. Therefore, a list of hashtags related to our domain of interest is generated as the initial query, before Apify API is used to scrape posts which contain those initial hashtags. By collecting all the hashtags that are used in the posts we scrape, we can obtain a large dataset containing hashtags in our domain of interest.

4.2 Preprocessing Data using Python Scripts

We also pre-process the dataset obtained from the last step using Python to generate a form that can be passed down to R scripts and become a graph input. As being implemented in the **json_csv.py** file, the top 3 popular hashtags are explored. We collect the information of all posts that mentioned at least one of these hashtags, including post ID, URL, number of likes, owner username, post caption and its timestamp. Note that these post information will be renamed in the section below for the R scripts. As a result, the extraction of source hashtags is finished as illustrated in step 1 of Stage 1 in Fig. 2.

4.3 Creating Graphs using R Scripts

After the previous two steps, the clean data are organized in the .csv file format with columns "ID, URL, Likes, Ownder, Text, Date". The open-source GitHub repository [7] provides some basic R scripts to help build connections between different hashtags and compute the weight of each edge (i.e., the frequency of two hashtags' co-occurrence).

hashExtractor.R scans through all the post texts and extracts related hashtags by tokenizing them. It also captures the frequency of each hashtag for later usage and organizes the hashtags with their frequency in .csv format. Note that hashtags with degree lower than 80 are filtered out for the purpose of data cleaning. Then, Stage 1 of the entire process is finished.

graphCreator.R reads from the .csv file generated from the previous script and generates an edge matrix. Then it calculates the degree of each hashtag (i.e number of connections to other hashtags). To improve the link prediction of the hashtags that have a lower frequency, the ones with a lower degree will be more focused. Graphs will then be generated using `induced.subgraph` function as shown in Fig. 3.

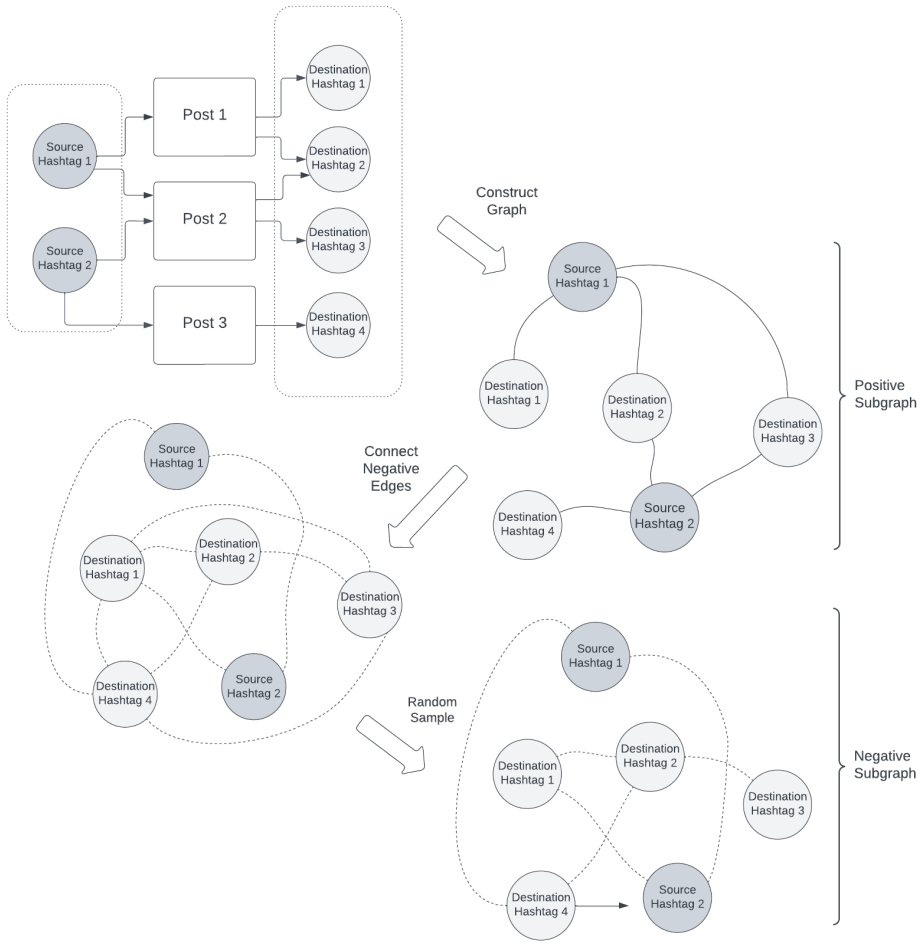


Figure 2: Data gathering and processing pipeline, including four stages which are introduced in Section 4.2 and Section 4.3. Stage 1, 2, 3 and 4 are ordered as what the arrow points to.

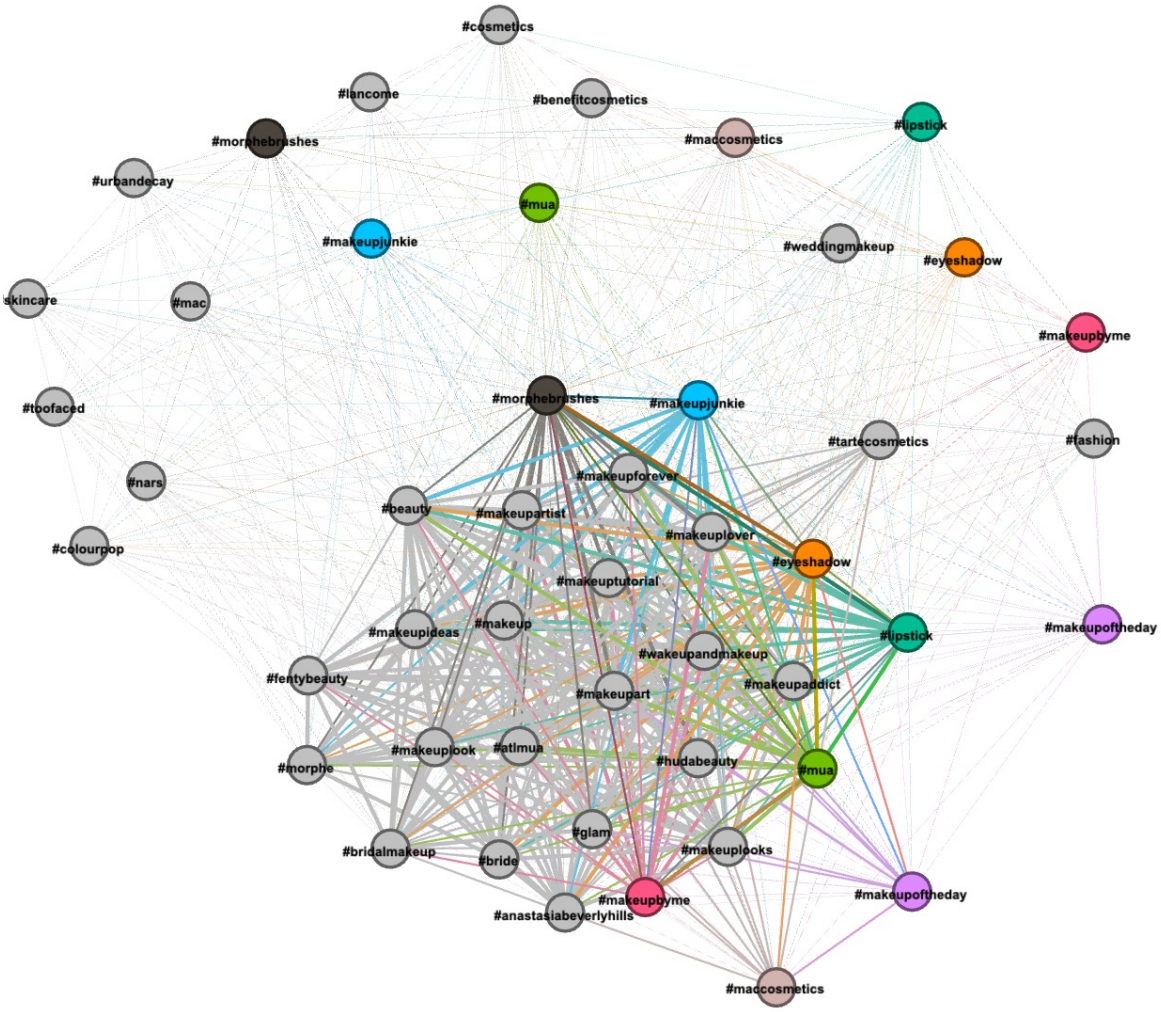


Figure 3: The generated graph with dark grey edges representing strongly related edges and light grey edges representing weakly related edges.

4.4 Subgraphs

With the edge information exported from the R scripts, an original graph of social networks is obtained, which is also called a *positive* subgraph (Stage 2 of Fig. 2) containing all existing edges. A fully connected graph can thus be generated by permuting all nodes in the graph as demonstrated in Stage 3. The non-exisiting edges are named as *negative* edges, by randomly sampling which and reaching to the same number of positive edges, a negative subgraph is acquired (Stage 4). Now, the positive and negative subgraphs can be passed to the GNN model as inputs.

5 GNN Model

According to the paper, the mean operator can be seen as a "skip connection" as it does not concatenate the node's previous layer with the current layer of the neighboring nodes, it is proved

to improve the performance of the model [8]. The project will use the mean aggregator to update the embeddings of the nodes according to the formula:

$$h_v^k \leftarrow \sigma(\mathbf{W} \cdot \text{MEAN}(h_v^{k-1} \cup h_u^{k-1}, \forall u \in \text{N}(v))) \tag{1}$$

where h is the embeddings, v is the updated node, k represents the k-th layer, $N(\cdot)$ are the connected nodes.

The **objective of the training process** is to maximize the similarity between the node embeddings learned from the model and the original node features in the dataset. After sending the positive and negative subgraphs into the model and passing through two GraphSage submodules (Fig. 4), the model will output the predicted node embeddings. To calculate the loss, the function of `DotPredictor` is used to compute the edge embeddings from the node embeddings and compared them with the ground truth label (connected edges are labeled as 1 and unconnected edges are labeled as 0). The choice of the predictor has not been finalized and will be investigated in the future.

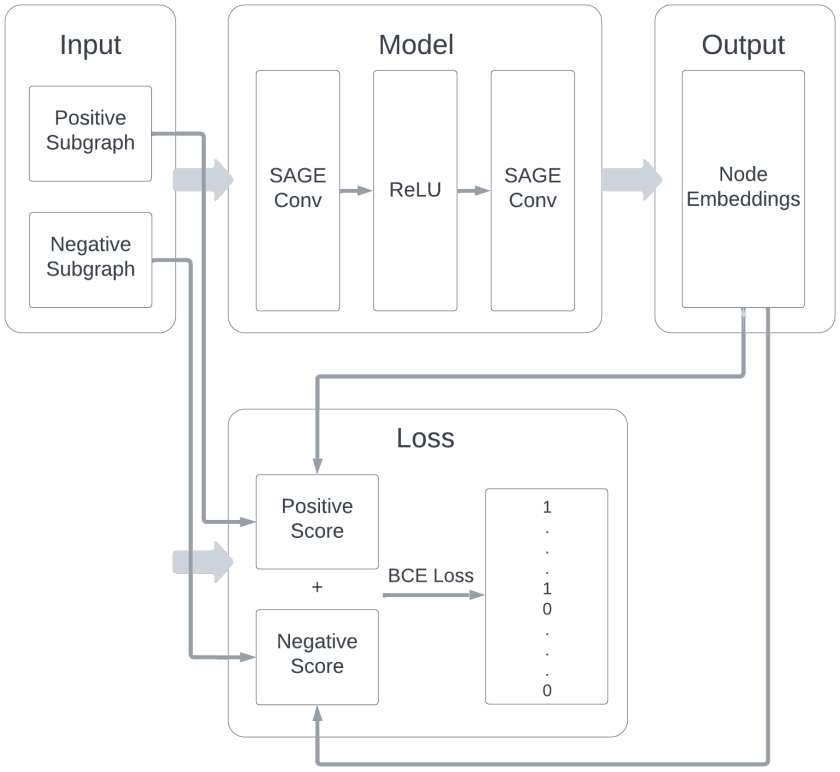


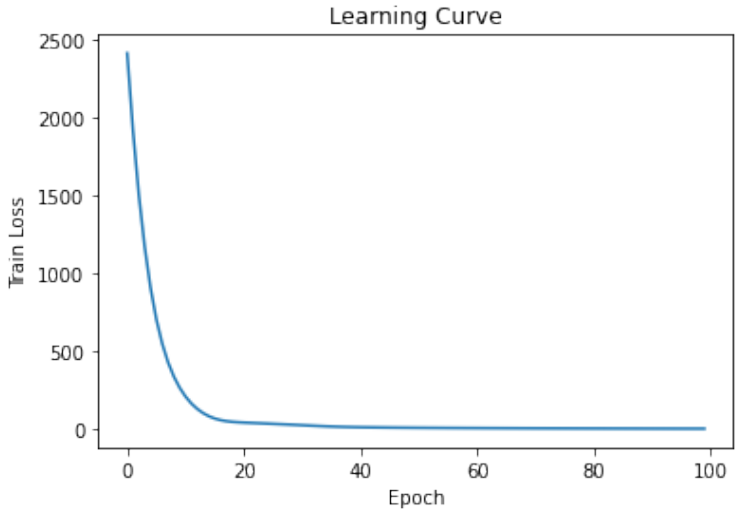
Figure 4: The model architecture used in the paper.

6 Result

We used the 25 most discussed cosmetic brands on Instagram as our initial query, then scraped 10 users for each brand, resulting in 49 total hashtags and 795 edges.

After training the model with this dataset with 100 epochs and a learning rate of 0.01, the final test accuracy is 51.2 %. Compared to the ratio in the original dataset (50% of connected edges and 50% of unconnected edges), the current model only tried to recover the original graph.

Improvements in the training dataset and the model layers will be explained in the next section, and fine-tuning the models will be conducted in the next phase of the project.




```
In epoch 0, loss: 2463.833984375
In epoch 5, loss: 483.6066589355469
In epoch 10, loss: 103.05918884277344
In epoch 15, loss: 142.61648559570312
In epoch 20, loss: 88.5102767944336
In epoch 25, loss: 61.44527053833008
In epoch 30, loss: 42.86172866821289
In epoch 35, loss: 28.663013458251953
In epoch 40, loss: 22.353992462158203
In epoch 45, loss: 20.000226974487305
In epoch 50, loss: 17.80171012878418
In epoch 55, loss: 15.992810249328613
In epoch 60, loss: 14.295371055603027
In epoch 65, loss: 12.842416763305664
In epoch 70, loss: 11.677030563354492
In epoch 75, loss: 10.720305442810059
In epoch 80, loss: 9.886724472045898
In epoch 85, loss: 9.262994766235352
In epoch 90, loss: 8.668993949890137
In epoch 95, loss: 8.155742645263672
```

Figure 5: The learning curve of the model with the processed dataset.

As the test accuracy can be improved with a finer-tuned model, we can use the output result to predict possible node connections. In a practical sense, a brand may build a network of all used hashtags in its existing posts. Then, new hashtags may be introduced to the built network for analyzing connections with the used hashtags. Negative edges with high predicted probability point to the hashtags that can be potentially popular and trigger a new trend, but are yet discovered. Consequently, companies can develop products catering to this new trend indicated by those promising hashtags or send posts with these hashtags to improve brand popularity and gain competitive advantages.

7 Challenges, Fairness, and Future Improvements

7.1 Improvements on Data Collection and Processing

The current structure for data collection only involves two steps (i.e hashtags → posts → hashtags) and it is hard for the GNN to actually learn the relationship between the hashtags that are weakly connected. Thus, we will repeat the two-step process one more time in the future to increase the number of hashtags included for our study.

In the next steps, hashtags with a low degree (e.g., 80) will also not be filtered out, since they might provide more information about possible connections and will increase the dataset size.

7.2 Improvements on Input Features

The current node features only contain the frequency of each hashtag, which underrepresented the nodes. Next `Word2Vec` can be used to convert hashtags texts to a vector, which captures the similarity of the hashtags from the semantic perspective and is suitable for using the `DotPredictor` for the final result.

Also, edge weights computed by the R scripts are not well utilized in our current project. Instead, they should be used as edge features as a future improvement, such that the relationship between hashtags are well represented and learned.

Besides, the overall emotion associated with each hashtag can be another input feature, it can be estimated by scanning through all the related posts. This feature can suggest the preference of the users and enhance the prediction.

7.3 Improvements on Model Structure

Bidirectional graphs actually better represent social networks, whose performance will be evaluated and compared as a next step. Additionally, more layers can be added to the model and different aggregator functions can be tested. The predictor used during the training process should also be treated as a hyperparameter. By fine tuning the model using the playbook provided in class, the prediction accuracy should be improved and the suggested hashtags will effectively improve companies' revenue and advance its long term development.

8 Conclusion

In this project, we aimed to forecast product and social media trends by gathering a dataset of Instagram hashtags. To achieve this, we collected hashtags related to our field of interest and extracted related hashtags from posts. We then used Python and R scripts to process the dataset and generate inputs for a Graph Neural Network model. At this stage, we have collected 49 total hashtags with 25 initial hashtags, resulting in 795 edges and a model which has a final test accuracy of 51.2%, indicating its ability to predict potentially popular hashtags. As a next step, we will optimize the data preparation process and incorporate additional input features and an improved model to enhance the prediction accuracy.

References

[1] M. Singh and Gobindbir Singh. Impact of social media on e-commerce. *International Journal of Engineering and Technology(UAE)*, 7:21–26, 05 2018.

[2] Yunpeng Xiao, Yu Zhu, Weikang He, and Mengyang Huang. Influence prediction model for marketing campaigns on e-commerce platforms. *Expert Systems with Applications*, 211:118575, 2023.

[3] Amandeep Singh, Malka Halgamuge, and B. Mouss. *An Analysis of Demographic and Behavior Trends Using Social Media: Facebook, Twitter, and Instagram.* 01 2019.

[4] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The World Wide Web Conference*, WWW '19, page 417–426, New York, NY, USA, 2019. Association for Computing Machinery.

[5] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[6] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216, 2017.

[7] Schroeder Jonas. Instacrawlr. `https://github.com/JonasSchroeder/InstaCrawlR`, 2018.

[8] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216, 2017.