# Graph-Based User Behavior Modeling and Simulation in Subscription Business

**Changhee Han**
University of Texas at Dalls
changhee.han2@utdallas.edu

**Abstract**

Customer retention is critical in the subscription business since it generates a fundamental driving force for the business model. Previous researchers and managers in this organization analyzed customer behavior to prevent the churning event. Our research focuses on Netflix customer pattern analysis with a novel approach, graph theory. The graph methods capture the customer's movie session with the feature of each movie and predict the link for the next film. Based on IMDB data, we simulated the prediction graph in a movie recommendations environment. The simulation environment recommends a hundred sets of movies to the customer graph model, and the model decides whether it retains the subscription. Our study contributed to the behavioral research using a graph theoretical framework and also the simulation study by utilizing trained graph models as agents.

## 1 Introduction

A subscription-based organization, like Netflix, provides an unlimited video streaming service to subscribers who registered a periodic payment plan. The streaming platform maintains a revenue structure by attracting new customers with content and retaining existing customers not to cancel the recurring payment. Customer management is the main challenge for this business model since customer churn directly affect the entire business process. To maintain subscriber retention, managers identify a user behavior pattern to provide customized service by analyzing a watch history, search queries, and time spent watching.

Previous research proposed various algorithms to predict customer churn in a subscription-based business model. Katelaris and Themistocleous [1] reviewed previous churn classification research in subscription-based services and suggested the development of accurate customer profiling for a comprehensive prediction model. Kolomiiets et al. [2] analyzed client outflow in a software subscription business and performed a churn prediction with statistical machine learning methods. Kumar and Kumar [3] built a multi-layer perceptron algorithm for a churn classification task in the telecommunication subscription business and highlighted dominant variables in the model. Previous research focused on a user profile and transaction history in customer data and utilized artificial intelligence techniques for the subscription cancellation prediction. Our research introduces a novel theoretical approach to predicting churn in subscription-based business and evaluates models with a simulation study.

Graph theory depicts the relation of data as a connected network within mathematical structures, and graph-based methodologies discover knowledge from the structure. Each user's video history can be represented as a consecutively connected action in a video streaming service. We converted each user's movie selection into a connected node set and encoded all user's movie sessions into the network. Moreover, we improved the movie session graph by combining with movie's attributes collected from IMDB and developed an artificial user model using the graph with node features. Our research utilized three different graph network models and trained them to predict the churn event.

To evaluate our trained model, we set an environment that can simulate a streaming service and customer reaction. A trained graph model behaves like a customer who choose a movie in sets of movie list based on a prediction score. A simulated streaming service provides a hundred of movie lists to the customer model based on pre-defined recommendation algorithm, depending on a prior selection. According to simulation results, our environment policy determines a customer churn event.

Our research contributed to the customer behavior literature with two ways. Firstly, we utilized the graph theory in the subscription-based business model. Graph methods encoded customer behavioral patterns into network structures and proposed a trained model for a prediction task. In addition, we evaluated the trained model with simulation environment based on the recommendation system and analyzed a customer model's reaction. This approach potentially improve a customer behavior analysis in the subscription-based organization.

The paper is composed as follows: Section 2 introduces the basic concept of the graph theory and its application in the subscription business field. Section 3 presents the graph-based research methodologies and recommendation methods used in the simulation study. Section 4 explains the entire process of the experiment and the result. Section 5 shows the simulation study and our model's reaction to the environment. Finally, in section 6, we discuss the experiment and simulation results and present the conclusion.

## 2 Literature Review

### 2.1 Graph Theory

Recent researchers have considered graph-structured data as a new challenge for machine learning methodologies. Various studies have transformed the graph structure into a vector type, utilized with machine learning concepts, and developed graph-based methods like graph-attention and Graph Convolutional Network (GCN). This research trend has catalyzed a deeper analysis of existing graph-based business research.

Piao et al. [4] found high-value customers with a graph attention mechanism by modeling customer demographic attributes and social networks jointly and predicted future customer value with the graph neural network. Kpiebaareh et al. [5] applied a graph-based method with product review data for an exploratory sensitive analysis which discover a textual connectivity from text data. Their graph technique enabled them to extract the specific information from the graph structure and potentially improved a future business by handling customer feedback effectively. Jalilifard et al. [6] embedded three different user graph structures, user-user relationships, user-activity sequences, and user-seller interactions, into vector space to capture customer behavior. The research suggested the user vector model as an inferring tool for financial worthiness analysis and a future behavior prediction task. Xu and Qu [7] developed a graph attention network, which encodes purchase attributes and behavior sequences to predict user behavior patterns.

Previous research have utilized graph-based methods to analyze customer data and generate implications in various ways. Our approach is similar in terms of representing a graph structure in a behavioral session with features of behavior and utilizing it as an application. However, the graph understanding of user patterns in subscription-based media streaming business remain unexplored. Thus, our work introduces the theory on the customer behavioral study to shed light on user pattern in video streaming industry.

### 2.2 Simulation and Agent-Based Modeling

A simulation study imitates a real-world event to find key characteristics or effects of elements in the environment. Wooldridge and Jennings [8] introduced the agent-based modeling (ABM) concept, a simulation system composed of agents, who make decisions in a given situation, and an environment that simulates problematic phenomena. The ABM can incorporate complex dynamics arising from interactions between individual behaviors and systems across various domains [9]. Previous studies advanced the ABM environment by setting theoretical frameworks and diverse conditions in the business domain and proposed a solution for the real-world problem by investigating the agent's reactions.

Zhang et al. [10] extended ABM research by modeling user behavior in the movie choosing process, investigating user-recommender interaction, and suggesting user consumption strategies for movie recommendation services. Jiang et al. [11] simulated a review website to examine user's reaction against the page components and suggested an optimal site design for the user decision support. Ahn

[12] modeled users' shopping behavior in an e-commerce simulation and analyzed the feasibility of personalized recommendations for a customer aid function.

Our study adopted the above customer analysis framework to identify customer retention in the subscription-based video streaming service. We proposed the agent model with graph methods and the simulation environment with movie recommender models explained in the methodology section. The simulation work focuses on the agent's decision on whether the agent retains the service or not by calculating prediction scores from the movie recommendations.

## 3 Methodology

Our behavior modeling work pursues to encode a user movie session with movie information and predict the probability of the next movie consumption. Netflix customers using the streaming service all rely on the same given movie list and watch movies sequentially. Therefore, we regarded each movie as a node in the whole given film list and a session as a link between movies. We also considered that users select a movie based on information, including directors, actors, genres, and others' reviews. Based on this assumption, we collected film data from the IMDB site explained in section 4. Thus, our research designed the user behavior prediction using the movie graph and the movie property and tested the prediction model with a recommendation simulation. In this section, we introduce link prediction graph methods which utilize a graph structure with the node property and simple recommendation methods.

### 3.1 Link Prediction

A link prediction purposes for forecasting a future structure of given graphs based on the similarity of the node attributes. In our behavior modeling research, we utilized three link prediction algorithms. All three methods train node features, but the concepts in each algorithm are different.

Before specifying our models, we followed the standard notation in graph theory[13]. An graph is represented as $G = (V, E, F)$, which denotes an undirect connected graph with $n$ nodes $(e.g., v_i \in V)$, $m$ edges $(e.g., e_ij = (v_i, v_j) \in E)$, and $d$ features $(e.g., f_i \in F)$. A feature vector is expressed as $X = [x_1, x_2, ..., x_n]^T \in \{0, 1\}^{n \times d}$, where $x_i \in \{0, 1\}^{n \times d}$ denotes the feature for $i$-th node $v_i$. The adjacency matrix $A \in \{0, 1\}^{n \times n}$ indicates the connection between nodes, where each component presents the existence of an edge between two nodes.

#### 3.1.1 Graph Convolution Network

GCN combines each node with neighboring nodes and filters significant information from the corresponding node, following the concept of a convolutional neural network. Kipf and Welling [14] suggested a novel approach, which encodes the entire graph structure on a neural network, while decreasing information loss. The entire architecture of GCN can be summarized in two steps: converting the graph into a vector and forward propagation with filtering related node features. Firstly, the GCN algorithm calculates an adjacency matrix, $\hat{A}$, as

$$\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} D^{-\frac{1}{2}}.$$

$\tilde{A} = I + A \in \mathbb{R}^{\mathbb{N} \times \mathbb{N}}$ is equivalent to the adjacency matrix $A$ added with identity matrix $I$ from the entire graph, and $\tilde{D}$ is the diagonal degree matrix of $\tilde{A}$. A matrix $\hat{A}$ is represented as a symmetric weighted adjacency matrix, which stores the entire information in a graph and enables the graph to be learned on a neural network. The next task is the forward network with a feature filtering function:

$$Z^k = \sigma(\hat{A} X^{k-1} W^k),$$

where $X \in \mathbb{R}^{\mathbb{N} \times \mathbb{C}}$ is a filtered feature vector with $C$ dimensions of features, $W \in \mathbb{R}^{\mathbb{C} \times \mathbb{F}}$ is a trainable parameter matrix, and $\sigma$ is an activation function. The convolution framework linearly operates, filtering features on a given graph matrix and multiplying a weight matrix. An activation function finally generates a representation of a graph, $Z$, with filtered features.

#### 3.1.2 Attri2Vec

Attri2Vec algorithm employs the concept that a latent subspace of a graph can represent the whole structure[15]. The framework consists of two parts, one is generating a latent space of a graph based on a random walk method, and the other is mapping the latent set of nodes on the embedding vector. A random walk explores neighboring nodes in the graph and closely represents similar nodes in the embedding vector [16]. A random walk follows:

$$\max_{f} \log \Pr(\{v_{r_{i-t}}, ..., v_{r_{i+t}}\} \backslash v_i \mid f(v_{r_i}))$$

, where a node sequence $\{v_{r_{i-t}}, ..., v_{r_{i+t}}\}$ with $t$ window size is generated with the probability of node selections conditioned on the random walk function $f$. This occurrence probability can be calculated as

$$\prod_{j=i-t, j \neq i}^{i+t} \Pr(v_{r_j} \mid f(v_{r_i})) = \frac{\exp\left(f(v_{r_i}) \times W_{r_j}^{out}\right)}{\sum_{k=1}^{|V|} \exp\left(f(v_{r_i}) \times W_{r_k}^{out}\right)}.$$

,$f(v_{r_i})$ transit a node vector into the random walk function, $W_r^{out}$ is a parameter matrix in the vector projection process. The probability $\Pr(v_{r_j} \mid f(v_{r_i}))$ can be calculated with an optimization of the softmax function by updating parameters. The Attri2vec method leverages each node with feature data and projects an embedding vector on the trained weight matrix. The embedding vector is applied with binary prediction methods for a prediction task.

### 3.1.3 GraphSAGE

The basic framework of GraphSAGE predicts two links' connection with the node embedding vector[17]. GraphSAGE follows an inductive embedding approach, which samples a neighboring graph of a node, extracts feature information from neighbors by using an aggregator function, and projects embedding for all nodes. Instead of generating an embedding vector from all nodes, GraphSAGE trains aggregator functions that can be generalized to unsampled nodes. This concept follows :

$$h_v^k = \sigma(W^k \cdot CONCAT(h_v^{k-1}, F(h_u^{k-1}, \forall u \in N(v)))),$$

where $h_v$ is the embedded vector of the node $v$, $k+1$ is the current index of the node, $W^k$ is the weight matrix of learning parameters, $F$ is the aggregator function, and $N(v)$ is the neighboring nodes of node $v$. The aggregator function, $F$, incorporates node vector $h_v$ with the neighboring feature vectors into one single vector. The algorithm then concatenates the node vector with aggregated neighborhood vector. This concatenated vector passes the neural network layer and updates the node embedding vector.

## 3.2 Recommendation System

After the completion of user modeling, our work inputs the trained model to the simulation framework, and the model interacts with a simulation environment. The environment provides one hundred movie recommendations to the user agent based on recommendation algorithms. We utilized three recommender algorithms: random sampling, collaborative filtering, and content-based filtering. In this subsection, we explained these three methods.

Firstly, we used a random generation, which selects movie pairs randomly. A random recommendation is the most straightforward concept in the recommender system.

$$Rand(A, B)$$

The second method is a collaborative filtering which filters items according to the population's preference. This method assumes that all service users would select items in similar preference and recommends items based on others' choices. The algorithm, called a neighborhood-based algorithm, calculates the similarity of two indices in vector space. The similarity equation finds the nearest neighbor in an item-user matrix by using cosine-similarity between items. The equation follows as

$$Simil_1(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|},$$

where $\|A\|$ is the euclidean mean of vector $A$, defined as $\sqrt{\sum_{i=1}^{n} A_i^2}$. $Simil_1(A, B)$ computes the cosine of the angle between two input vectors, and the similarity score ranges from 0 to 1.

The last method is a content-based filtering, which assumes that the item feature affects the user's preference. The algorithm selects item sets which have similar characteristics. The mechanism

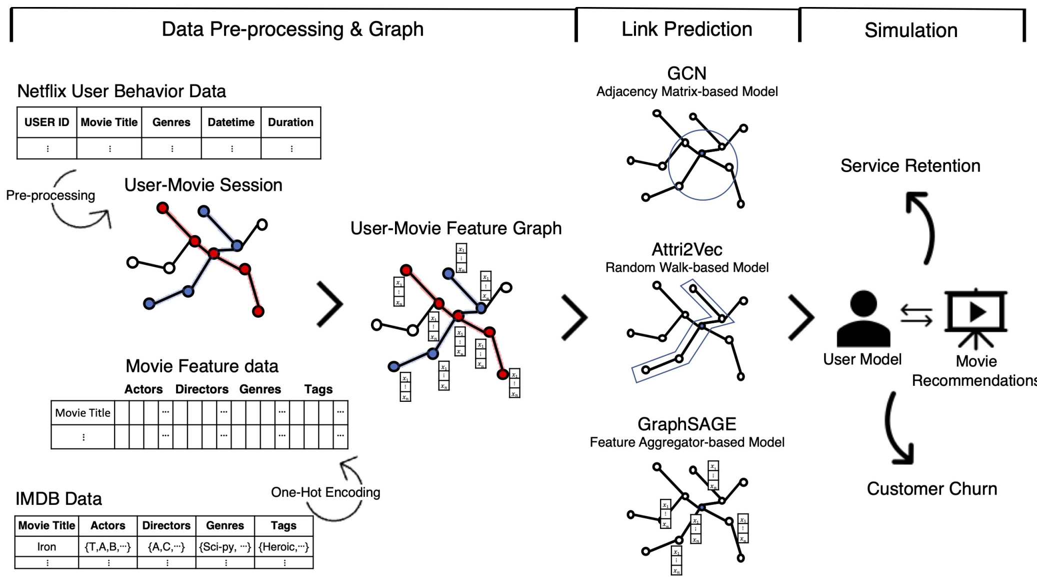

**Figure 1:** Research Framework.

of this filtering is that the number of shared features between items determines the similarity. We developed this concept by calculating an inner product between two items in an item-feature vector. The equation follows as

$$Simil_2(A, B) = \langle A, B\rangle = \sum_{i=1}^{d} A_i B_i,$$

where two items, A and B, both items have the same $d$-features in binary vectors. An inner product measures the similarity between two items, and the similarity score ranges from 0 to $d$.

## 4 Experiment

Our objective of this research is an identification of an user behavior pattern. The experiment trains three graph methods with customer's movie session data. In this section, we introduce our datasets with additive feature crawling, graph-based user modeling, and simulation study, respectively (Figure 1).

### 4.1 Data Collection and Preprocessing

We used publicly available datasets, Netflix audience behavior sets [18], which tracked UK users' activity in an anonymized system. Each row of this data contains an user ID, a datetime, movie title, movie genres, and a duration which is a click interval from when the user starts watching a movie to the next click. We converted this data into an user-movie session form by sorting the entire data with an user ID and listing movie titles in a datetime order. Our data preprocessing dismissed duration data because Netflix captures every click of users and recommends items to users. Our graph consisted of 8,472 nodes and 267,926 edges.

Moreover, we collected IMDB movie data and utilized the sets as attributes of movie data. IMDB datasets contain movie titles, actors, directors, genres, and tags from user text reviews. We converted movie features into a vector of size 177,197 by using one-hot encoding. This feature set was a highly sparse matrix which caused a poor performance of the proposed models. Therefore, we performed a dimension reduction on the feature vector by using a PCA method and experimented with a graph link prediction with 300 dimensions of a feature vector [19].

### 4.2 Experimental Setting

In this subsection, we benchmarked three models, proposed in Section 3. Each model performed a link prediction task with the same training condition and was developed in Stellargraph package [20].

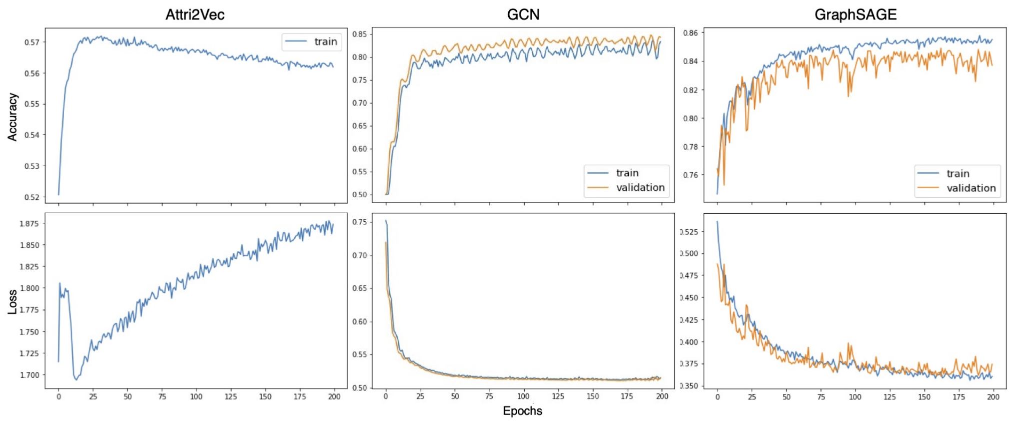

**Figure 2:** Training Results.

**Table 1:** Test Results.

| Method | Loss | Accuracy |
|---|---|---|
| Attri2Vec | 1.85 | 0.56 |
| GCN | 0.37 | 0.85 |
| GraphSAGE | 0.36 | 0.84 |

We set the training environment using Adam optimizers for stochastic gradient descent, a learning rate of 0.01, and 200 train epochs. For Attri2Vec, we set a weight matrix, which performs a random walk-based embedding for a node representation, with 128 dimensions. The model performs a random walk on subgraphs in the entire graph structure; thus, our work randomly selects 10 percentages of nodes from the whole nodes. The random walk performed on 100 of window sizes and 40 of the number of walks. The node representation of Attri2Vec passed a logistic regression to obtain the link prediction score. For GCN, we set two trainable parameter matrices as convolution filter layers with 128 and 64 dimensions for each. The output of the graph convolution network passes a sigmoid function to obtain the link prediction score, the probability of the link's existence. For GraphSAGE, our work set two neural network layers with 128 dimensions for both. The model requires sample sets of nodes to perform an aggregator function; thus, we set the number of sample to 200. A final sigmoid layer scores the linkage of pairs delivered from the GraphSAGE embedding.

## 4.3 Link Prediction Experiments

From Figure 2, we observed that GCN and GraphSAGE performed a better training performance in comparison to Attri2Vec. The test process also showed a similar result in Table 1. Attri2Vec, the random walk-based embedding algorithm, is learned on sampled graphs in the entire graph. Thus, the sample size is a key indicator of the model's performance. Netflix user-data has a large volume of edge, and a small proportion of data creates a biased subgraph, which cannot fully explain a population of data. GCN trained stably and showed similar results in both train and test courses. The whole graph structure is passed to the neural network, and this learning algorithm causes the result. The test accuracy score shows that the model can represent the unknown graph structure. Finally, the GraphSAGE model, which updates the parameter by aggregating neighborhood feature data, showed a similar accuracy score to the GCN model. However, this model showed volatility in the training part because of the algorithm's characteristics.

## 4.4 Simulation

Our simulation follows a concept of ABM in Section 2. we employed each of trained graph models as an agent, developed a streaming service system with recommendation algorithms described in Section 3, and set a retention checking policy. In this subsection, we introduce a simulation experiment.

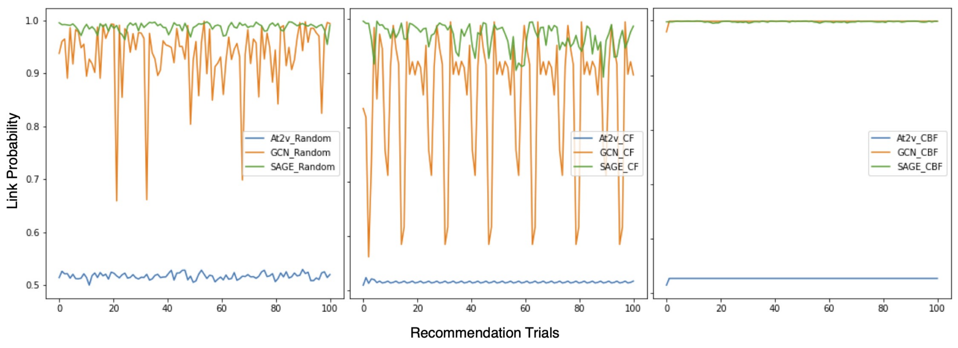

**Figure 3:** Simulation Results.

We simulated a real-world streaming service of a movie recommendation and user interaction. We supposed that a customer watches at least one film from the set of items, and a system recommends sets of movies based on the user's prior selection. A recommendation algorithm generates recommendation sets unless an user keeps the service usage. Following this concept, our algorithm starts with an initial movie selection, supposing that an user chooses a movie at the beginning of the service. After then, Our recommendation model delivers a hundred sets of movie pairs to an agent; the first one in the pair is an initially selected movie, and the other is a recommended movie based on an algorithm in Section 3. The agent predicts a link in every pair and selects a pair that has the highest link prediction score. The simulation system saves a selected pair and removes the first item in the pair from entire movie sets to prevent the system from selecting the same pair. The process repeats with a second movie in the prior pair and recurs until the system recommends movies 100 times or the policy decides to stop the process. The policy checks an average of the entire prediction score, and if all probabilities are under 0.50, the system policy classifies the simulation result as a churn event.

The simulation is programmed various options in the selection of an agent and a recommender algorithm. We experimented with different agents on the same condition of a recommendation system. The system recorded historic prediction scores for each user model, and each model showed a difference performance in difference recommendation models; random generation, collaborative filtering, and content-based filtering (Figure 3). An Attri2vec-based model showed a poor result compared to other models, meaning that a simulation depends on the model training. In recommendation systems of the random generation and the collaborative filtering, the GCN model's link prediction score shows higher variance than the GraphSAGE model's result. The random generation creates pairs constituted with irrelevant items, and the collaborative filtering chooses a pair based on others' historical events. Thus, GCN and GraphSAGE, sensitive models with attributes of nodes, had a poor performance in both recommendation systems that have no relation with feature data result. Conversely, in a contents-based filtering system, GCN and GraphSAGE recorded stable scores because the filtering method recommends a movie based on features of the movie.

## 5 Discussion

In the modeling part, we focused on the graph-based behavior modeling with combined data of Netflix user and IMDB movie attributes. Our experiment trained three graph methods which have different concepts, and we argued that the concept of the model affects the training result. Our experiment illuminated how a training algorithm in each user model affects the result of a link prediction. Considering a key characteristic of a movie industry, which new genres and movies are released in every year, an algorithm which heavily depends on a sample of the entire graph would not suit for the user modeling.

Our simulation examined an interaction between a recommendation algorithm and a trained user model. The simulation result showed that a basic principle of an user model algorithm is related with a logic of recommendation algorithm. In a real-world business, a service director can employ various algorithms for their recommendation model. If an user make a decision based on the movie's attributes, then a service director choose an algorithm that utilizes attribute data for the service user.

However, our research has several limitations in the user modeling experiment. Firstly, our movie session data is tracked on a click-based system. Even if a user chooses a movie mistakenly, the system will capture the moment and record it as clicking behavior. Netflix presented that the service captures the entire behavior of the service user, thus we encoded the clicking action as nodes [21]. However, for precise modeling, an accurate tracking service would be required. In addition, our research utilized graph methods in a vanilla condition. We selected the proposed models because these models have their unique concepts, and it is suitable to find the model for the user pattern modeling. However, numerous fine-tuned models performed highly in a link prediction task, and these methods could have a better result than the proposed models.

In conclusion, we introduced a graph technique to customer behavior prediction in a subscription-based movie streaming industry. We converted user action sets into the graph structure and combined them with feature data. Our work utilized three methods to train the graph and identified that the mechanism of algorithms affects user behavior modeling. We discovered that encoding the entire information of the graph structure performed better in a link prediction task. Furthermore, we simulated a real-business environment with the graph-user models. The study implicates dependency of a recommendation system on the user's preference in a movie selection. This finding provides new insight into behavior modeling and business strategy.

## References

[1] Leonidas Katelaris and Marinos Themistocleous. Predicting customer churn: Customer behavior forecasting for subscription-based organizations. In *European, Mediterranean, and Middle Eastern Conference on Information Systems*, pages 128–135. Springer, 2017. 1

[2] A Kolomiiets, Olga Mezentseva, and Kateryna Kolesnikova. Customer churn prediction in the software by subscription models it business using machine learning methods. In *CEUR Workshop Proc*, volume 3039, pages 119–128, 2021. 1

[3] Sanjay Kumar and Manish Kumar. Predicting customer churn using artificial neural network. In *International Conference on Engineering Applications of Neural Networks*, pages 299–306. Springer, 2019. 1

[4] Jinghua Piao, Guozhen Zhang, Fengli Xu, Zhilong Chen, and Yong Li. Predicting customer value with social relationships via motif-based graph attention networks. In *Proceedings of the Web Conference 2021*, pages 3146–3157, 2021. 2

[5] Michael Y Kpiebaareh, Wei-Ping Wu, Brighter Agyemang, Charles R Haruna, and Tandoh Lawrence. A generic graph-based method for flexible aspect-opinion analysis of complex product customer feedback. *Information*, 13(3):118, 2022. 2

[6] Amir Jalilifard, Dehua Chen, Lucas Pereira Lopes, Isaac Ben-Akiva, and Pedro Henrique Gonçalves Inazawa. Friendship is all we need: A multi-graph embedding approach for modeling customer behavior. *arXiv preprint arXiv:2010.02780*, 2020. 2

[7] Fangyao Xu and Shaojie Qu. Data mining of students' consumption behaviour pattern based on self-attention graph neural network. *Applied Sciences*, 11(22):10784, 2021. 2

[8] Michael Wooldridge and Nicholas R Jennings. Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(2):115–152, 1995. 2

[9] Charles M Macal and Michael J North. Agent-based modeling and simulation. In *Proceedings of the 2009 winter simulation conference (WSC)*, pages 86–98. IEEE, 2009. 2

[10] Jingjing Zhang, Gediminas Adomavicius, Alok Gupta, and Wolfgang Ketter. Consumption and performance: Understanding longitudinal dynamics of recommender systems via an agent-based simulation framework. *Information Systems Research*, 31(1):76–101, 2020. 2

[11] Guoyin Jiang, Xiaodong Feng, Wenping Liu, and Xingjun Liu. Clicking position and user posting behavior in online review systems: A data-driven agent-based modeling approach. *Information Sciences*, 512:161–174, 2020. 2

[12] Hyung Jun Ahn. Evaluating customer aid functions of online stores with agent-based models of customer behavior and evolution strategy. *Information sciences*, 180(9):1555–1570, 2010. 3

[13] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001. 3

[14] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*, 2017. 3

[15] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. Attributed network embedding via subspace discovery. *Data Mining and Knowledge Discovery*, 33(6):1953–1980, 2019. 3

[16] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014. 4

[17] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *NIPS*, pages 1024–1034, 2017. 4

[18] VOD Clickstream. Netflix audience behaviour-uk movies, Feb 2021. URL `https://www.kaggle.com/datasets/vodclickstream/netflix-audience-behaviour-uk-movies/metadata`. 5

[19] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987. 5

[20] CSIRO's Data61. Stellargraph machine learning library. `https://github.com/stellargraph/stellargraph`, 2018. 5

[21] Carlos A Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4): 1–19, 2015. 8