MINISTRY OF EDUCATION AND TRAINING

VIETNAM ACADEMY OF SIENCE AND TECHNOLOGY

GRADUATE UNIVERSITY OF SIENCE AND TECHNOLOGY

NGUYEN TUAN KHANG

RESEARCH AND DEVELOPMENT OF SOME SESSION-BASED RECOMMENDATION TECHNIQUES WITH DEEP LEARNING MODELS

SUMMARY OF DISSERTATION ON COMPUTER SCIENCE
Major code: 9 48 01 01

Ha Noi – 2023

The dissertation has been completed at: Graduate University of Science and Technology - Vietnam Academy of Science and Technology

Supervisors

1. **Supervisor 1**: **PhD. Nguyen Phu Binh**
   Victoria University of Wellington, New Zealand
2. **Supervisor 2**: **Assoc. Prof., PhD. Nguyen Viet Anh**
   Institute of Information Technology
   Vietnam Academy of Science and Technology

Referee 1: …

Referee 2: …

Referee 3: …

The dissertation will be examined by Examination Board of Graduate University of Science and Technology, Vietnam Academy of Science and Technology at ……, …… (time, date)

This dissertation can be found at:

1. Graduate University of Science and Technology
2. The National Library of Vietnam

Introduction

1 Thesis motivation

In the context of rapid development of e-commerce and online services, recommendation systems have become an important tool to enhance customer experience and drive business growth. Traditional recommendation models such as content-based approaches and collaborative filtering methods mainly focus on long-term personal preferences and overlook short-term interactions.

With such research motivation, the session-based recommendation system has been proposed, and its aim is to predict the next customer action based on the behavior of the current session. From this perspective, the author emphasizes the importance of researching models that recommend customer purchasing behavior based on sessions and exploring new possibilities that they bring to enhance the field of recommendation systems in order to forecast customer behavior.

2 Thesis objective

Problem statement

Analyzing customer sessions to predict their likelihood of purchasing a specific product or choosing the next product is a common forecasting problem in the e-commerce industry. This forecasting helps businesses come up with appropriate sales ideas during the user's interaction with their sales system.

Research subject

The research subject of this dissertation is the mouse-click behavior sequence in the process of customer product selection. The mouse-click behavior sequence is recorded during a shopping session on an e-commerce system or any social networking platform.

Research objective

The objective of this thesis is to study and propose a model for predicting a customer behavior during the product selection in the current session of the sales system. Specifically, this thesis has several main research objectives as below:

- Research and propose a method for representing session data
- Research and propose several deep neural network models and graph neural network models to build a predictive model for purchasing behavior.
- Experiment with various alternative options and compare them with several baseline models to evaluate the effectiveness of the proposed model.

Research scope

The scope of the research approaches two specific problems:

- The Problem 1 answers the question "*With the current list of selected products in the interactive session, what is the likelihood that customers will make a purchase, and if they do, which item are they likely to choose?*".
- The Problem 2 is more general and answers the question "*With the current list of selected products in the interactive session, what is the likelihood that customers will choose certain products next?*"

3 Research methodology

Problem 1 is a simple binary classification. The thesis proposes two neural network models: wide and deep learning networks, and transformer-based machine learning networks to analyze session data in tabular format. The session data consists of attributes with numerical and categorical data to predict whether a customer will make a purchase or not. These neural network models are simple and suitable for tabular data sessions. However, their limitation is that they only evaluate data within specific sessions (intra-session) and do not assess the relationships between sessions in the entire dataset.

With Problem 2, the research approach needs to be improved by Learn and propose a method for representing session data, especially the ability to clearly show the relationship between millions of sessions in real-world datasets, this concept is called inter-session. Graph is a very suitable approach to represent session data of millions of customers in the process of selecting from a set of products of a certain system. From the architectural perspective, the thesis research and propose using graph neural network model to build recommendation models for Problem 2.

4 Thesis layout

The structure of the thesis consists of an Introduction and four chapters, and the Conclusion is described briefly as follows:

- *"Introduction"*: The introduction presents an overview of the research problem, its urgency, and the practical scientific significance of the topic.
- *Chapter 1 "Overview of recommendation systems"*: This chapter presents the problem of recommendation that many e-commerce systems or social networking platforms are implementing. It defines and states two problems corresponding to two specific objectives of the thesis mentioned in the Introduction section, including Problem 1 as a binary prediction model of whether or not to make a purchase and Problem 2 as a top-k recommendation system based on the current customer's browsing session when clicking on product selections in the e-commerce system.
- *Chapter 2 "Proposal of a deep neural network model for customer purchasing prediction"*: This chapter addresses Problem 1 of the dissertation, answering the question "*Does the customer make a purchase in the current session?*". It proposes two specific neural network models, namely wide and deep neural networks, and transformational neural networks, to build a purchasing prediction model.
- *Chapter 3 "Proposal of a graph neural network model for the top-k recommendation"*: Chapter 3 addresses the general top-k problem, which is the main focus of this thesis. This chapter presents several graph design options to model the input information as customer session logs, including two single graphs $\mathcal{G}$ and $\mathcal{H}$, and a multi-relational graph $\mathcal{K}$.
- *Chapter 4 "Improvement of GNN model with embedding"*: In order to further improve the proposed GNN model in Chapter 3, Chapter 4 introduces graph transformation to enhance the effectiveness of the model. The author suggests optimizing the GNN graph neural network model by proposing a new special graph embedding layer to improve top-k prediction. This chapter designs a session embedding layer using a combination of embedding transformations, including vertex embedding, graph embedding, and label embedding.
- *"Conclusion"*: The final section presents general conclusions and comments on the achieved results of the thesis to explain the research motivation and steps for improving the models.

Chapter 1 | Overview of recommendation systems and deep neural networks

1.1 Recommendation system

1.1.1 Overview

There are several different recommendation systems depending on the context of the problem. The simplest system relies on the user's historical information or preferences to find the most suitable product. This type of system is easy to understand but faces challenges when it comes to providing recommendations for new users, as it has no historical information from them. A new form of recommendation system relies solely on the current interaction process of the user, known as a session. Based on session information, the system can provide recommendations after just a few interactions with the user, and this model is called a session-based recommendation system.

1.1.2 Classification of recommendation systems

Each type of recommendation system uses different algorithms and techniques to understand and analyze data, in order to provide suitable recommendations based on user preferences and needs.

- Content-Based Filtering.
- Collaborative Filtering.
- Hybrid Recommendation Systems.
- Knowledge-Based Recommendation Systems.
- Context-Aware Recommendation Systems.
- Reinforcement Learning-Based Recommendation Systems.
- Session-Based Recommendation Systems.

1.2 Two fundamental problems

1.2.1 Definition of a working session

Definition 1. *A customer's working session is a sequence of mouse click events when selecting products, recorded by the system as a vector* $s = \{id_1, id_2, ..., id_c\}$ *where* id_i *is the product identifier,* c *is the number of products clicked in the working session* s*, and also the length of that session.*

1.2.2 Problem 1 - Purchase behavior prediction

Problem 1. *Given a time-ordered sequence of mouse clicks generated from a customer's working session when selecting products, we need to build a model to predict whether the customer will make a purchase in the current work session or not.*

1.2.3 Problem 2 - Top-k recommendation system

Problem 2. *Given a sequential mouse click string generated from a customer's session when selecting products, we need to build a recommendation model to determine which item the customer will choose next in the current session.*

1.3 Theory of deep neural networks

1.3.1 Feedforward neural network model

This section explores some specific improved models of feedforward neural networks (FNN) to provide a broader understanding of deep learning techniques in solving Problem 1. These three models have similar characteristics to FNN but differ in the pre-processing method of the embedding layer before entering the deep learning layer. The variations of the FNN model are illustrated in Figure 1.1.

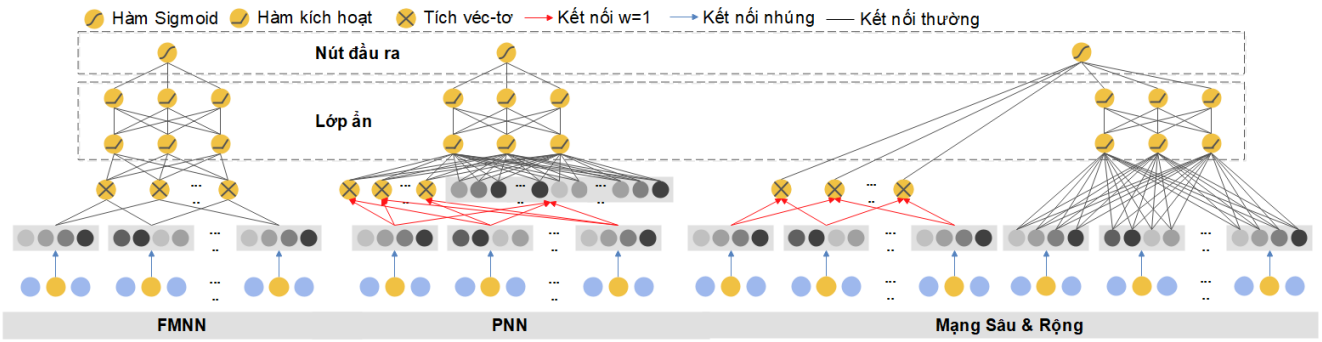


Figure 1.1: Some neural network models used in mouse click prediction

1.3.2 Wide and deep neural network model

With a focus on researching and applying deep neural network for Problem 1, the author utilizes a wide and deep neural network to serve the stated objective. This model was proposed in 2016 by a team at Google.

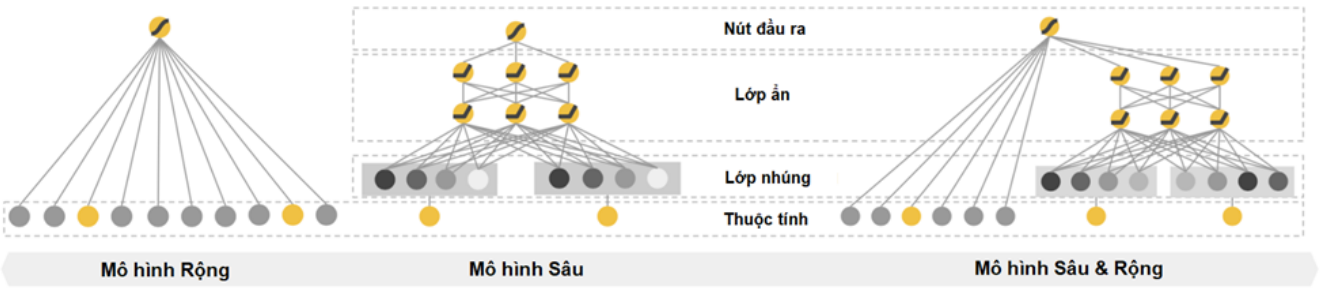


Figure 1.2: The architecture of a wide and deep neural network

The wide and deep model is a hybrid neural network with a structure consisting of two branches described as follows:

Wide branch

The wide component is a linear model in the form of:

$$y = W^T x + b \tag{1.1}$$

The input field includes raw attributes and some special attributes generated through cross-product transformations, as shown in formula 1.2:

$$\varphi_k(x) = \prod_{i=1}^{d} x_i^{c_{ki}}, c_{ki} \in \{0, 1\} \tag{1.2}$$

where c_{ki} takes the value of 1 if the if the i-th attribute belongs to the k-th transformation of φ_k, and takes the value of 0 otherwise.

Deep branch

The deep part is a feedforward deep neural network combined with embedding technique, where the first layer of the feedforward network is the attribute embedding layer. The output of the embedding layer is in the form of $a^{(0)} = [e_1, e_2, ..., e_m]$, where m is the number of attribute fields and e_i is the embedding vector of the i-th attribute field. These vectors are then combined with numerical attributes and passed to the next hidden layers of the deep neural network.

$$a^{l+1} = \sigma(W^{(l)}a^{(l)}) + b^{(l)}) \tag{1.3}$$

In which σ is the activation function, usually the $ReLU$ function in the form $f(x) = x^+ = \max(0, x)$; $W^{(l)}$, $a^{(l)}$, and $b^{(l)}$ are the output and bias of the l-th neuron layer.

The learning process of the network occurs simultaneously for both parts to generate the final result of the predictive model according to Formula 1.4

$$\hat{y} = Sigmoid(y_R + y_S) = \frac{1}{1 + e^{-(y_R + y_S)}} \tag{1.4}$$

In which $\hat{y} \in (0, 1)$ is the prediction value of purchase probability, y_R is the output of wide branch and y_S is the output of deep branch.

1.3.3 Transformer neural network model

The Transformer transformation model consists of two main modules: the encoding block and the decoding block, described in Figure 1.3:

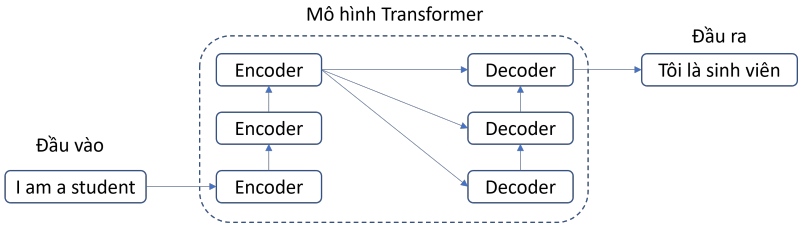


Figure 1.3: Illustration of the Transformer architecture

The Transformer architecture is quite similar to basic deep neural networks such as W&DNN, FNN, PNN... as presented in the previous section, as it also uses a combination of embedding layers and feed-forward neural networks. However, there are two differences: (1) the Transformer architecture uses self-attention mechanism for transforming the input data into sequential form, (2) these blocks are stacked together to process different attributes from the input data in parallel.

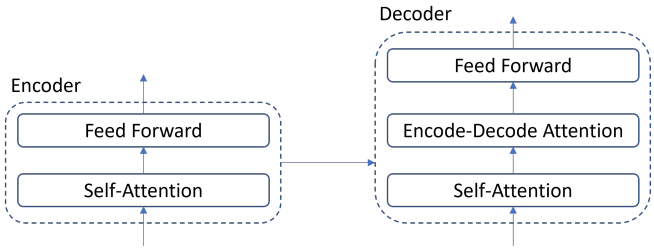


Figure 1.4: Layers of the Transformer architecture

1.4 Graph neural network theory

1.4.1 Definition of graph

According to the basic definition, a graph is a collection of objects called vertices connected by edges, where each edge represents a specific relationship between two vertices. Depending on the specific problem, the edges can be directed or undirected, and the corresponding graph is then referred to as directed or undirected, respectively, as stated in some statements.

Definition 2. *A simple graph G consists of a non-empty set V, whose elements are called vertices, and a set E, whose elements are called edges, which are unordered pairs of distinct vertices. This graph is also known as an undirected graph.*

The mathematical expression represents a graph described by Formula 1.5.

$$G = (V, E) \tag{1.5}$$

where

- $V = \{v_1, v_2, ..., v_n\}$ lis the set of vertices of the graph, and the number of vertices $n = |V|$.
- $E = \{e_1, e_2, ..., e_m\}$ is the set of edges of the graph, and the number of edges $m = |E|$.

Definition 3. *A directed graph $G = (V, E)$ consists of a set of vertices V and a set of edges E, which are ordered pairs of elements belonging to V.*

With more complex graph structures, they can have different types of edges connecting vertices. This graph is called a multi-relational graph because it contains multiple layers of different relationships. For a multi-relational graph, we need to add a parameter to indicate the type of relationship (type of edge) between 2 vertices (u, v) through a function f such that $f(e) = (u, v)$.

Definition 4. *A multi-relational undirected graph $G = (V, E)$ consists of a set of vertices V, a set of edges E, and a function f from E to $\{\{u, v\}|u, v \in V, u \neq v\}$. Edges e_1 and e_2 are called parallel edges if $f(e_1) = f(e_2)$.*

Definition 5. *A directed multi-relational graph $G = (V, E)$ consists of a set of vertices V, a set of edges E, and a function f from E to $\{\{u, v\}|u, v \in V\}$. Edges e_1 and e_2 are called parallel edges if $f(e_1) = f(e_2)$.*

Definition 6. *(Adjacent vertices) Two vertices u and v in an undirected graph G are called adjacent if $\{u, v\}$ is an edge of graph G. If $e = \{u, v\}$, then e is called an incident edge with vertices u and v. Edge e is also called a connecting edge between vertices u and v, and vertices u and v are called the endpoints of edge $\{u, v\}$.*

Definition 7. *When $e = \{u, v\}$ is an edge of the directed graph G. u is called the starting vertex and v is called the ending vertex of the edge $\{u, v\}$.*

Definition 8. *(Degree of a vertex) The degree of a vertex in an undirected graph is the number of edges connected to it. The degree of vertex v is denoted as $deg(v)$.*

Definition 9. *In a directed graph, the incoming degree ($deg^-(v)$) of vertex v is the number of edges with v as the ending vertex. The outgoing degree ($deg^+(v)$) of vertex v is the number of edges with v as the starting vertex.*

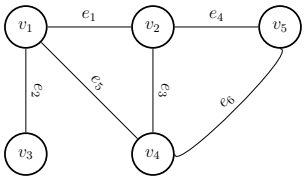
Definition 10. *(Path) A path P from vertex v_1 to vertex v_k is a set of vertices $\{v_1, v_2, ..., v_k\}$ such that there exists $(v_i, v_{i+1}) \in E, \forall i : 1 \leq i < k$. The length of path P is $P(v_1, v_k) = k - 1$ as it does not count the starting vertex v_1, and this length is also the number of edges contained in that path.*

1.4.2 Graph representation

a. Adjacency list

The adjacency list is a list that represents all the edges of a graph. In an undirected graph, each element of the list is a pair of two vertices that are the endpoints of the corresponding edge. In a directed graph, each element is an ordered pair of two vertices representing the starting and ending vertex of the corresponding arc.

Figure 1.5 illustrates how to represent a graph using an adjacency list.



(a) Graph illustration

Vertex	Adjacent vertices
v_1	v_2, v_3, v_4
v_2	v_1, v_4, v_5
v_3	v_1
v_4	v_1, v_2, v_5
v_5	v_2, v_4

(b) List of adjacent vertices

Figure 1.5: Representation of a graph using an adjacency list

b. Adjacency matrix

When representing a graph using an adjacency list, the algorithm construction process can be quite cumbersome if the graph has many edges. To simplify the computation, we can represent the graph using an adjacency matrix.

Let's assume $G = (V, E)$ is a simple graph with n vertices. We can represent the graph using a matrix $A_G = [a_{ij}] \in \mathbb{R}^{n \times n}$, which is also known as an adjacency matrix:

- $a_{ij} = 1$ if $\{v_i, v_j\} \in E$.
- $a_{ij} = 0$ if there is no edge connecting vertex v_i and vertex v_j.
- It is conventionally set that $a_{ii} = 0$ with $\forall_i$.

In the case of weighted graph representation, the value $a_{ij} = w(i, j)$ represents the weight of the edge connecting two adjacent vertices v_i and v_j.

1.4.3 Mô hình mạng n-ron đ` thị

The graph neural network model was first introduced in 2005. GNN is a type of neural network that operates directly on graph structures. By using neurons as nodes in the network structure, each node contains its own information and collects additional information from neighboring nodes to represent their relationships in the graph. These nodes are organized and combined according to a specific model architecture to make predictions or classify results. Typically, GNN focuses on solving the following problems:

- Node classification.
- Link prediction.
- Clustering detection.
- Graph classification.

1.5 Embedding transformation

1.5.1 Concept of embedding

In the field of machine learning, embedding is a technique used to transform discrete attribute data, such as words or categories, into continuous vectors in a lower-dimensional space. Thus, embedding maps each discrete variable to a real-valued vector, which can be used as input for a neural network.

Embedding techniques can be used with various types of data such as discrete data, text, time series data, images, or graphs. The next section of the thesis will present some embedding techniques used in the following chapters of the thesis, including:

- Embedding techniques for discrete data used for feedforward neural networks proposed in chapter 2 and chapter 3.
- The technique of embedding sequential data (such as text sentences) is used for the proposed transformation of neural networks in chapter 2, or time series data is used for recurrent neural networks.
- The technique of embedding graph data is used for the proposed graph neural networks in chapter 4.

1.5.2 Embedding transformation with discrete data

The two most common types of data are continuous and discrete data, which are categorized as tabular data. Continuous data is represented by real numbers, while discrete values like product categories are represented by text labels or numeric labels. In reality, labeling is just a convenient way to represent the value dictionary of a discrete attribute, these labels don't actually have any useful value like continuous attributes.This type of data is called categorical attributes, they can be ordered or unordered.

The important point is that the neural network model is not suitable for processing categorical data due to their discrete nature. Therefore, discrete attributes need to be transformed into vector form to represent their continuous nature within their value range. The transformed vector representations will help improve the learning ability of neural network models in capturing the correlation between discrete attribute values as well as the interactions between attributes. The transformation process consists of two steps as shown in Figure 1.6.

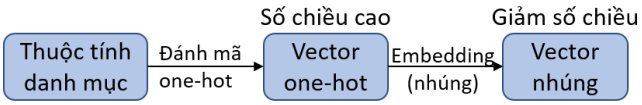


Figure 1.6: Transformation of categorical attributes into embedding vectors

The technique of feature embedding is used to construct a feature vector for a categorical attribute in its value domain. This technique aims to represent and rearrange elements with similar influences close to each other in order to (1) discover the continuity of data in the embedding space, and (2) capture the relationships between discrete categories of the attribute, thereby enabling deep neural networks to learn more effectively. With this technique, the transformed embedding vector has lower dimensions and its components are real numbers instead of just 0 and 1 values like a one-hot vector.

1.5.3 Embedding transformation with sequential data

Basic deep neural network models (such as feedforward neural networks) can handle numerical and categorical data well, but they cannot process sequential data such as word sequences in a sentence or time series data. Therefore, when processing text, neural network models not only compute each word in a sentence but also consider how those words appear in order and relate to each other. The meaning of words can change depending on the words that appear before and after them in a sentence.

a. Sequential text data

There are three transformation techniques combined with embedding, as shown in Figure 1.7, using neural networks to process sequential data.

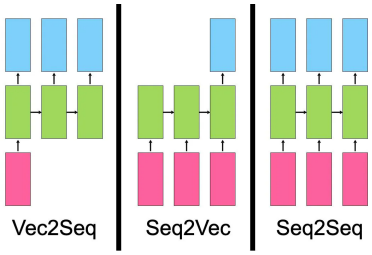


Figure 1.7: Techniques for processing sequential data for neural networks

b. Sequential time-series data

Time series data is quite common, such as stock prices, electrocardiogram signals, or more complex signals collected from IoT devices or smartphones. For this type of time series data, more suitable neural network models are needed, such as convolutional neural networks (CNN) or recurrent neural networks (RNN). Especially when working with multivariate time series data, Principal Component Analysis (PCA), although not entirely considered an embedding technique, is a very popular method for analyzing and reducing the dimensionality of this multivariate data.

1.5.4 Embedding transformation with graph data

Graph embedding, also known as graph embedding, is a technique that allows representing a graph as high-dimensional vectors. This enables the use of suitable machine learning algorithms or neural networks to process and analyze information within the graph, such as node classification, link prediction, and graph clustering.

There are multiple ways to perform graph embedding, such as random walk, deep walk, matrix factorization, and other methods based on deep neural networks. The results of graph embedding have numerous practical applications, such as social network analysis or recommendation system development. For example, it can be used to cluster similar users in a social network or suggest similar products to customers during the purchasing process.

Chapter 2 | Proposal of a deep neural network model for customer purchasing prediction

Chapter 2 presents the approach to solving Problem 1, which is the binary prediction of whether a customer will make a purchase in the current working session. This chapter proposes the use of two deep neural networks, including wide and deep neural networks and transformational neural networks, to learn from sequential data representing customer session information.

2.1 Problem statement

Let's assume that the training dataset consists of n samples $(\mathcal{X}, y)$, where $\mathcal{X}$ is a recorded data string with m attributes related to customers and products, and $y \in (0.1)$ is the label corresponding to the customer's purchasing behavior ($y = 1$ if the customer buys the product, and $y = 0$ otherwise). Therefore, Problem 1 is to build a model that predicts $y \approx \hat{y} = f(x)$ in order to estimate the probability of a user making a purchase based on the input data string.

2.2 Proposed models

2.2.1 Wide and deep neural network

The proposed wide and deep neural network model has the following architectural design:

- Wide branch: consists of 2 feed-forward layers, with the output layer having one neuron and the input layer having a number of neurons determined by: $N = N_{cat} + N_{num}$, where N is the number of neurons in the input layer, N_{cat} is the number of categorical attribute fields, and N_{num} is the number of pairwise interactions between categorical attribute fields.
- Deep branch: consists of 6 feed-forward layers, including 1 input layer with a number of neurons equal to the number of attribute fields, 1 embedding layer, 3 hidden layers with neuron numbers taken as $400 - 400 - 400$ respectively, and 1 output layer with 1 neuron. The hidden neurons use the *ReLU* activation function, while the output neuron uses the *sigmoid* activation function.

The structure of the model is shown in Figure 2.1.

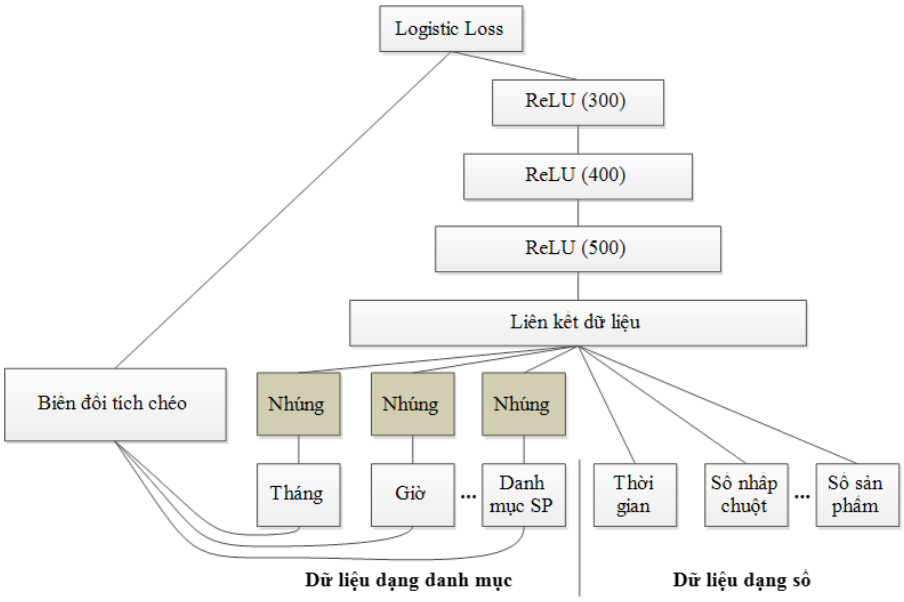


Figure 2.1: A wide and deep model structure used for mouse click prediction

This proposed network model has the following improvements:

- The proposal suggests using embedding with categorical attributes and linking data with other attributes to create a feature embedding vector for session work.
- Building a network architecture with several neuron layers in the deep branch (FNN branch).
- Performing cross-product transformation between pairs of attributes to discover hidden interactions between attribute fields.

Combining both deep and wide learning techniques helps improve the accuracy of the forecasting model compared to models that only use one technique.

2.2.2 Transformer network

The author proposes a modified Transformer architecture by adding an attribute embedding layer to improve the model's performance on tabular data, as described in Figure 2.2, called the FE-Transformer model. This model suggests adding an embedding layer to transform all attributes, including numerical and categorical ones, into embedding vectors, which will then be processed by a sequence of Transformer layers. Therefore, each Transformer layer has the ability to learn distinct features in the dataset.

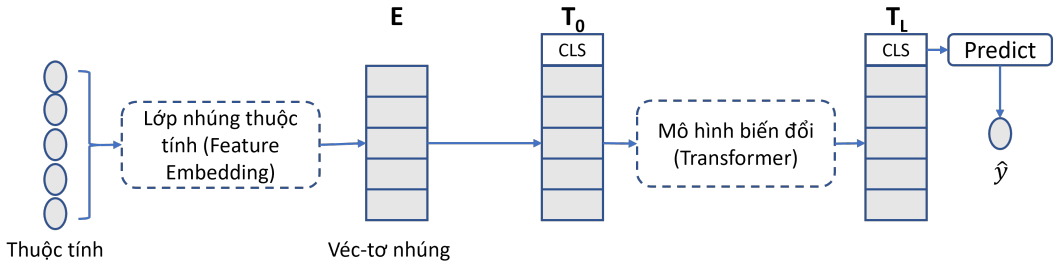
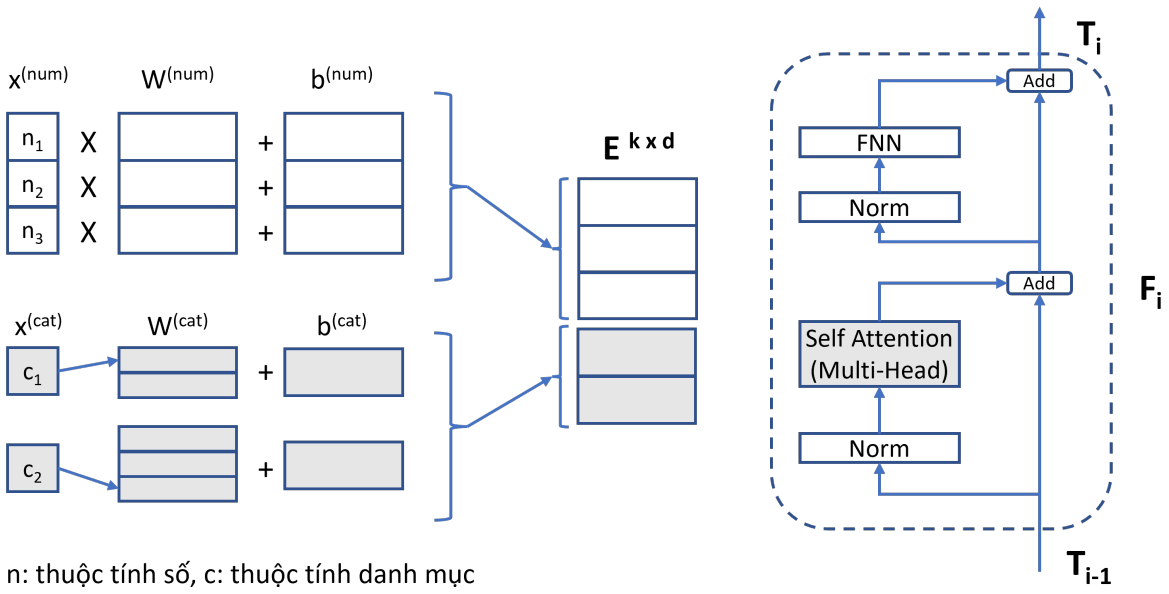


Figure 2.2: FE-Transformer Architecture

The detailed design of the two components of the FE-Transformer architecture is illustrated in Figure 2.3:



(a) Attribute embedding layer (b) Transformer layer

Figure 2.3: Design of layers for FE-Transformer model

2.3 Experimental technique

2.3.1 Experimental dataset

This experimental section uses the dataset provided by Yoochoose GmbH.

2.3.2 Feature engineering

Table 2.1 lists the extracted base attributes.

Table 2.1: List of extracted attributes

I	**Product attributes (2 attributes)**		
1	Product ID	Category	Product code
2	Cat ID	Category	Product category code
II	**Session attributes (11 attributes)**		
3	The First Product	Category	The first product in the session
4	The Pre Product	Danh mục	The previous product in the session
5	Session Duration	Numberic	Length of the session
6	Current Duration	Numberic	
7	#Clicks/Session	Numberic	Number of clicks in the session
8	#Products/Session	Numberic	Number of products in the session
9	#Clicks So Far	Numberic	Number of clicks up to now in the session
10	#Products So Far	Numberic	Number of products clicked up to now
11	#Views of Product	Numberic	Number of views for this product
12	#Products of the same Cat	Numberic	Number of products in the same category
13	#Cats	Numberic	Number of categories
III	**Time attributes in hours, minutes, seconds (9 attributes**		
14-16	Session Start	Category	Session start time
17-19	The first time that product is clicked	Danh mục	First time selecting a product
20-22	Current Time	Category	Current time
IV	**Boolean attribute (4 attributes)**		
23	The most clicked product	Boolean	The most clicked product in the session
24	The most viewed product	Boolean	The most viewed product in the session
25	The first clicked product	Boolean	The first clicked product in the session
26	The most viewed category	Boolean	The category with the highest views

2.3.3 Data splitting method

The entire dataset is randomly divided into 60% for training, 20% for evaluating the effectiveness during network structure optimization, and 20% for testing and comparing between expected network models during network structure construction.

Table 2.2: Label quantities for each dataset after splitting

Data	Yes	No	Total
Train	325,966	5,593,860	5,919,826
Validation	81,808	1,398,149	1,479,957
Test	101,922	1,748,024	1,849,946

2.3.4 Model evaluation metric

In order to find the best forecasting model, the experimental part uses the following basic indicators to analyze and evaluate different network structures:

- AUC (*Area Under the Curve*).

- Logloss (*Logarithmic Loss*).
- Đ. chính xác (*Accuracy*).

2.4 Experimental results

2.4.1 Experimental results

Table 2.3: Comparison of effectiveness among models in mouse click prediction

Model	AUC	Logloss	Accuracy
LR	0.7604	0.5842	0.6967
FNN	0.8521	0.6145	0.7789
FMNN	0.8620	0.5061	0.7814
PNN	0.8596	0.5332	0.7808
W&DNN	0.8670	0.4519	0.7826
FE-Transformer	0.7868	0.1844	0.9449

2.4.2 Comparison with related works

The study also compared the results with Yandex Data Factory in the RecSys Challenge 2015, using the Yoochoose dataset. According to this study, they used a combined method including Gradient Boosted Decision Tree + Factorization Machine + Singular Value Decomposition (SVD) analysis with an AUC score of 0.85 and an accuracy of 0.77. Therefore, it can be seen that the current research achieves better results with fewer computational resources.

The contributions of proposing and designing two deep neural networks are as follows:

- Both models use an improved feedforward neural network architecture. The W&DNN model combines the FNN network with a linear model in the wide branch. The FE-Transformer model uses self-attention to learn important features from session components.
- The W&DNN model uses embedding in the deep branch and cross-product transformation in the wide branch, allowing the model to capture both low and high-order attribute interactions. The FE-Transformer model is enhanced with attribute embedding.

2.5 Chapter conclusion

This chapter investigates and proposes the use of two specific neural network models, namely wide & deep networks and transformer networks, to address Problem 1 in predicting customers' shopping behavior based on clickstream data. The results show that the wide and deep model has several advantages: (1) it does not require pre-training, (2) it can learn both low-level and high-level interactions of attribute fields, (3) it leverages the memorization ability of linear models and the generalization ability of deep neural networks within the same model. The transformer model performs well in processing sequential data after applying an attribute embedding layer. The research results of the wide and deep model have been published in [A-1], and the transformer model has been submitted for publication in [A-8] (to ensure diversity in experiments, [A-8] uses a different dataset from this dissertation).

One important conclusion for Problem 1 is that accurate predictions of customer purchasing behavior can be achieved by analyzing mouse click sequences in the current session, without considering the user's historical information.

Chapter 3 | Proposal of a graph neural network model for the top-k recommendation

Chapter 3 presents an approach to solving Problem 2 in constructing a suggestive model. This chapter proposes representing session data as a graph, from which the use of graph neural networks to build the top-k recommendation problem is studied.

3.1 Problem statement

The top-k problem is a recommendation system that suggests products (such as movies, music, or items when making purchases...) for users based on their interactions and those of others with the system. The recommendation system will rank all proposed products in descending order of probability. The options can be chosen by users, and the return will be limited to the top-k recommended products.

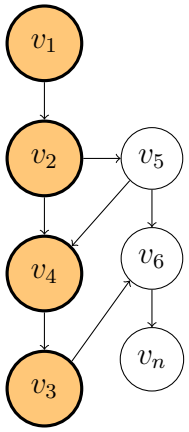
3.2 Proposed design using graph

3.2.1 Session representation using graph

A session s can be represented by a directed graph $G_s = (V_s, E_s)$. In which, each vertex represents a product $v_{s,i} \in V$ (V is the set of overall vertices of the entire system). Illustration of graph representation from work sessions sk is shown in Figure 3.1.

Session s_1	$v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_3$
Session s_2	$v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_4$
Session s_3	$v_2 \rightarrow v_5 \rightarrow v_6 \rightarrow v_n$
...	...
Session s_k	$v_5 \rightarrow v_4 \rightarrow v_3 \rightarrow v_6$
...	...

(a) List of sessions



(b) Graph representation

Figure 3.1: Illustration of representing sessions with a graph

Similar to a graph, when representing a session as a graph, we have some definitions:

Definition 11. *(local path length) Let v_i and v_j be any two clicked items in session s with click orders x and y respectively, where $x < y$. The length of the path from v_i to v_j in session s is denoted as $p_s(v_i, v_j)$ satisfying the formula:$p_s(v_i, v_j) = y - x$.*

Definition 12. *(p-click) Two clicks on the products v_i and v_j in a session s are called **p-click** if the item v_j is clicked exactly p times after v_i in the session s. In other words, two clicks on v_i and v_j in a session s are **p-click** if and only if $p_s(v_i, v_j) = p$.*

Definition 13. *(adjacent click) Two clicks on the products v_i and v_j in a session s are called adjacent click if the item v_j is clicked immediately after v_i in the session s. In other words, two clicks on v_i and v_j in a session s are adjacent click if and only if $p_s(v_i, v_j) = 1$.*

Definition 14. *(adjacent click weight) Two clicks on products v_i and v_j in a session s have a weight equal to the number of adjacent clicks generated by the two products v_i and v_j in session s, denoted as $w_s^{v_i,v_j}$. This weight is called the **adjacent click weight**.*

Definition 15. *(p-click weight) Two clicks on products v_i and v_j in a session s have a weight equal to the number of p-clicks generated by the two products v_i and v_j in session s, denoted as $w_{s,p}^{v_i,v_j}$. This weight is called the* ***p-click weight****.*

Definition 16. *(global path) A path P from vertex v_1 to vertex v_k where the vertices v_1 to v_k can be in different sessions, then the global path between these two vertices is the path between 2 vertices in the overall graph G representing the entire set of working sessions, denoted as $P(v_1, v_k)$.*

The question is: "With a vertex set $V = \{v_1, v_2, ..., v_n\}$ with a fixed number of n products, how should we represent the overall graph G?".

3.2.2 Proposed graph design

This section proposes several options for constructing graph G from the working session list of customers. Specifically, the author suggests three types of graphs as follows:

a. Graph $\mathcal{G}$

Let's call $\mathcal{G}$ a graph that satisfies adjacency matrix $M_G \in \mathbb{R}^{n\times n}$, where $M_G^{v_i,v_j}$ is the number of times product v_j is immediately clicked after product v_i in a session.

$$M_G^{v_i,v_j} = \sum_s w_s^{v_i,v_j}, \forall s \tag{3.1}$$

where $w_s^{v_i,v_j}$ is "*adjacency weight*" of two vertices v_i, v_j in working session s.

b. Graph $\mathcal{H}$

Let's call $\mathcal{H}$ be a graph satisfying the adjacency matrix $M_H \in \mathbb{R}^{n\times n}$ where $M_H^{v_i,v_j}$ is the number of times product v_j is clicked after product v_i in a session.

$$M_H^{v_i,v_j} = \sum_s \sum_{p=0}^{|s|} w_{s,p}^{v_i,v_j}, \forall s \tag{3.2}$$

where $w_{s,p}^{v_i,v_j}$ is "*p-click weight*" of two vertices v_i, v_j in working session s.

c. Graph $\mathcal{K}$

Suppose c is the maximum number of clicks in a session in the dataset. Let's call $\mathcal{K}$ be a graph satisfying the block adjacency matrix $M_K \in \mathbb{R}^{n\times n\times c}$, where $M_K^{v_i,v_j}[p]$ represents the number of times the product v_j is clicked after product v_i exactly p clicks in a session.

$$M_K^{v_i,v_j}[p] = \sum_s w_{s,p}^{v_i,v_j} \tag{3.3}$$

3.3 Proposed models

3.3.1 Feedforward neural network model (*FNN*)

This section proposes the use of a Feedforward Neural Network (FNN) as in chapter 2, but addresses Problem 2, which is to build a top-k recommendation model instead of Problem 1.

a. Product embedding layer

The section proposes the construction of a product embedding layer as shown in Figure 3.2. This embedding layer will be used as the base layer to build various models in this thesis.

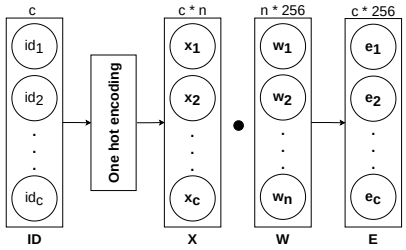


Figure 3.2: Product embedding layer (Layer.ItemEmbed)

b. Feedforward neural network model

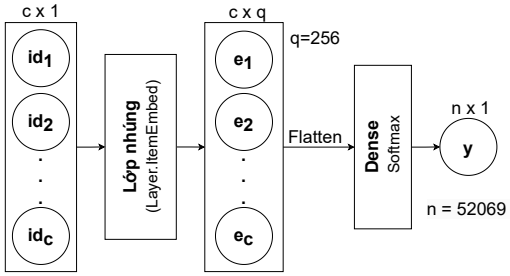


Figure 3.3: Baseline FNN model

3.3.2 Graph Neural Network (*GNN*)

a. Models for graph $\mathcal{G}$ và $\mathcal{H}$

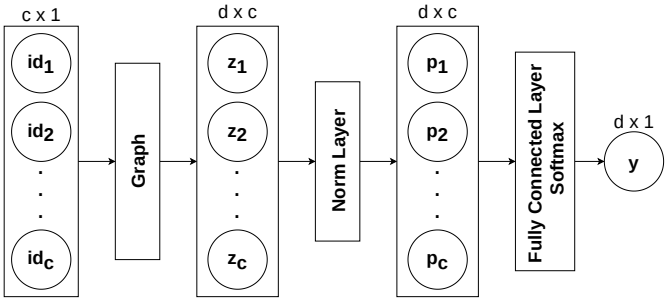


Figure 3.4: Models $\mathcal{G}$ và $\mathcal{H}$

b. Model for graph $\mathcal{K}$

To improve the graph neural network model when working with a multi-relational graph $\mathcal{K}$ with weights of edges is a vector c, the thesis proposes to use an additional deep learning layer as shown in Figure 3.5.

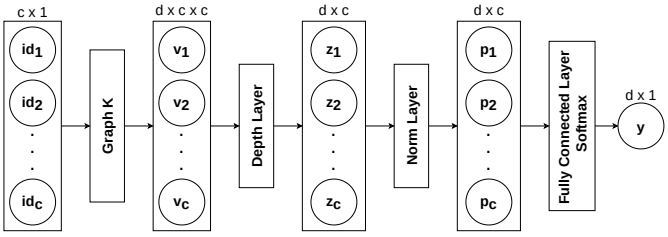


Figure 3.5: Model for graph $\mathcal{K}$

3.4 Experimental technique

3.4.1 Data preprocessing

The dataset after preprocessing is described in Table 3.1. The distribution chart of the number

Table 3.1: Statistics on the Yoochoose click dataset after preprocessing

	Train set	**Test set**	**Total**
Number of sessions	7,990,018	1,996,408	9,986,426
Number of products	52,069	38,733	52,069
Number of clicks	31,744,233	7,926,322	39,670,555
Highest clicks	200	200	200
Lowest clicks	2	2	2
Avg clicks	3.97	3.97	3.97

of sessions clicked from 1 to 10 times in Figure 3.6, as the number of sessions clicked more than 10 is very small, it does not need to be shown in this chart.

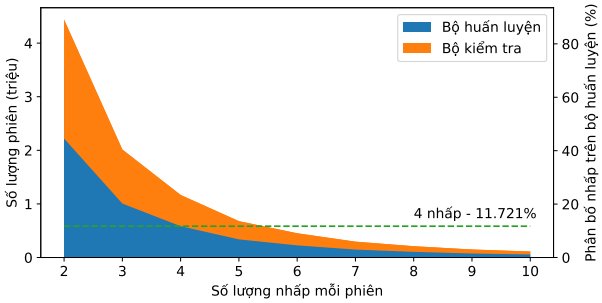


Figure 3.6: Distribution chart of the number of mouse clicks (after preprocessing).

3.4.2 Data normalization for training

The data sessions in the original dataset have different click counts, so they cannot be directly used for classification models. To obtain suitable training data for the models, the author proposes several algorithms to normalize the training data according to the input standards designed for the proposed models.

a. Data normalization for model FNN

The FNN model is a basic model that does not use graphs, so the algorithm for normalizing the data is quite simple and is shown in model 3.7:

The pseudo code of the data normalization steps is described in Algorithm 3.1:

b. Data normalization for model GNN

To obtain standardized input vectors for graph-based models, the normalization steps are described as shown in Figure 3.8 for each session of each graph.

The pseudo code of the data normalization steps is described in Algorithm 3.2:

3.4.3 Model evaluation metrics

Proposed evaluation metrics $Recall@k$, $MRR@k$ và $ACCs@k$ to evaluate top-k recommendation systems.

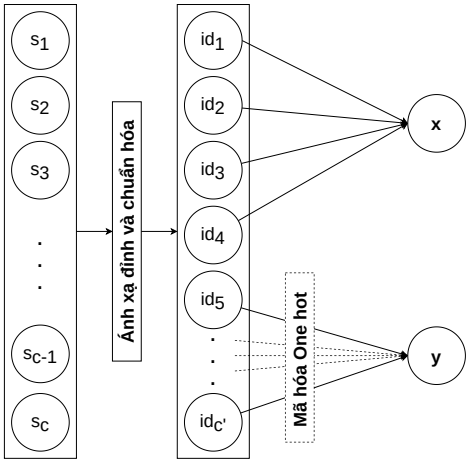


Figure 3.7: Data normalization for model FNN

Algorithm 3.1: NORM.FNN:
Data normalization for model FNN

Input: $s = \{id_1, id_2, ..., id_c\}$
Output: input for training as vector x và output y

$c' \leftarrow c$;
while $c' < 5$ **do**
  Append to session s an item $None$;
  $c' \leftarrow c' + 1$;
$\mathbf{x} \leftarrow \{id_1, id_2, id_3, id_4\}$;
$Z \leftarrow \{id_5, id_6, ..., id_{c'}\}$;
$\mathbf{y} \leftarrow OneHotEncoding(Z)$
return $\mathbf{x} \in \mathbb{R}^4$, $\mathbf{y} \in \mathbb{R}^{n \times 2}$;

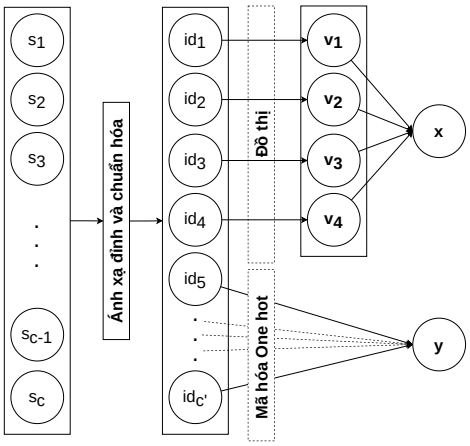


Figure 3.8: Data normalization for model GNN

$$Recall@k = \frac{1}{n}\sum_{i=0}^{n-1}\frac{|S^i_{pred} \cap S^i_{labels}|}{|S^i_{labels}|} \tag{3.4}$$

$$MRR@k = \frac{1}{n}\sum_{i=0}^{n-1} RR(id^i_*, S^i_{pred}) \tag{3.5}$$

Algorithm 3.2: NORM.GNN:
Data normalization for model GNN

Input: $s = \{id_1, id_2, id_3, ..., id_{c-1}, id_c\}$
Output: input for training as vector x và output y

$c' \leftarrow c$;
while $c' < 5$ **do**
    Append to session s an item $None$;
    $c' \leftarrow c' + 1$;
$\mathbf{x} \leftarrow \{\}$;
for $i \leftarrow 1$ **to** 4 **by** 1 **do**
    if $id_i == None$ **then**
        $v_i \leftarrow$ vector with all 0;
    else
        $v_i \leftarrow$ weight vector of vertex id_i in the graph;
    Thêm v_i vào $\mathbf{x}$
$Z \leftarrow \{id_5, id_6, ..., id_{c'}\}$;
$\mathbf{y} \leftarrow OneHotEncoding(Z)$
return $\mathbf{x} \in \mathbb{R}^4$, $\mathbf{y} \in \mathbb{R}^{n \times 2}$;

$$ACCs@k = \frac{1}{n}\sum_{i=0}^{n-1} \min(1, |S^i_{pred} \cap S^i_{labels}|) \tag{3.6}$$

3.5 Experimental results

Figure 3.9 describe the experimental results for models

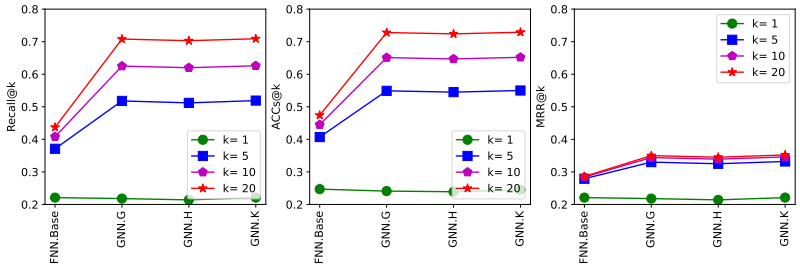


Figure 3.9: Comparision between model GNN and FNN

3.6 Chapter conclusion

In this chapter, the author discusses the proposal of designing three different graphs, namely a simple graph $\mathcal{G}$, another simple graph $\mathcal{H}$, and a multi-relational graph $\mathcal{K}$. These graphs differ in how they are designed in terms of edge sets and edge weights, representing different relationships between nodes, including relationships within intra-sessions and between inter-sessions in the dataset. Experimental results show that the Graph Neural Network (GNN) model combined with graph representations of work sessions yields very promising results compared to the Feedforward Neural Network (FNN) model without using graphs. The chapter concludes that Graph Neural Networks (GNNs) can be effectively used to build top-k recommendation systems.

Chapter 4 | Improvement of GNN model with embedding

With the results achieved in Chapter 3 for Problem 2 by representing work sessions as graphs, however, there is still a challenge that the proposed model must handle multi-label problems with a number of labels equivalent to the number of vertices in the graph, which is very large.

4.1 Challenges of multi-label classification problem

Multi-label classification is a difficult problem in machine learning for several reasons such as label dependency, large label space, imbalanced data, and feature extraction.

4.2 Graph embedding method

Definition 17. *Graph embedding is a technique to represent a graph as high-dimensional vectors with the purpose of supporting machine learning algorithms for processing and analyzing graph information, such as node classification, link prediction, and graph clustering.*

4.2.1 Vertex embedding

Vertex embedding is used to transform a vertex $v \in V$ into a d-dimensional embedding space to generate vertex embedding vectors in the new space $\vee \in \mathbb{R}^d$, as illustrated in Figure 4.1.

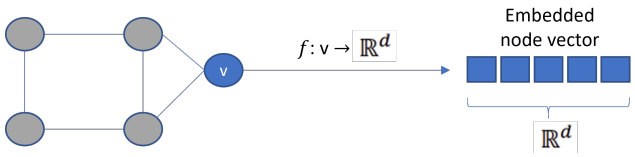


Figure 4.1: Vertex embedding

4.2.2 Graph embedding

The graph embedding is a transformation that takes a group of related vertices and embeds them into a d-dimensional embedding space to create embedding vectors in the new space $\vee \in \mathbb{R}^d$, as illustrated in Figure 4.2.

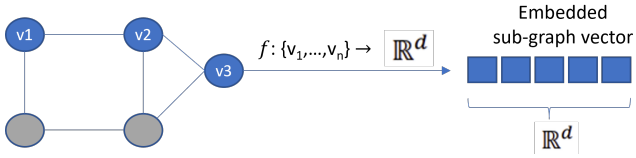


Figure 4.2: Sub-graph embedding

4.3 Proposed improvement of GNN.K model

4.3.1 Conversion of multi-label problem into binary problem

The author proposes an additional binary model to evaluate the effectiveness between the multi-label model and the binary model. To transform a multi-label model into a binary model, we input the labels to the model to answer ”yes” or ”no” with that label.

4.3.2 Proposed binary feedforward neural network

The author converts it into a binary model by continuing to use the *Layer.ItemEmbed* product embedding layer as the base model for FNN, but with the difference of adding the label component id^* and cross-combination with each id_i component of the input data. The proposed model is described in Figure 4.3.

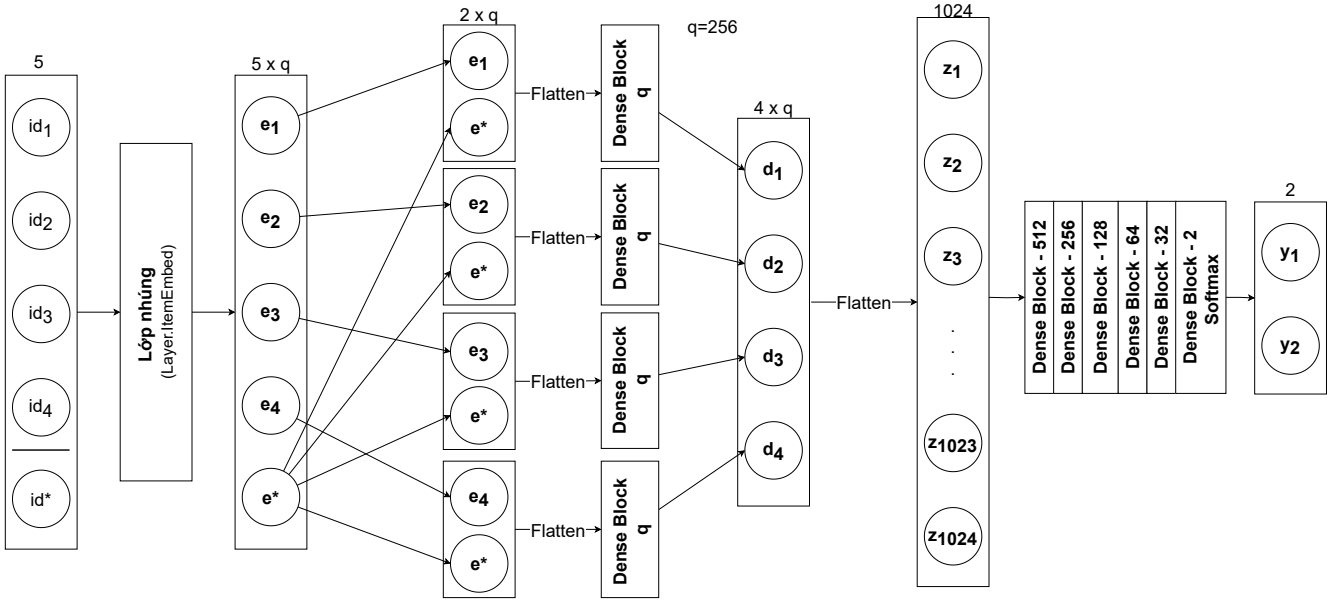


Figure 4.3: Binary FNN model ($FNN.bin$)

4.3.3 Proposed embedding model for binary graph K

a. Proposed session embedding layer combination

First, the thesis proposes a technical embedding of the session representation graph by combining the FNN.bin model (Figure 4.3) using the *Layer.ItemEmbed* product embedding layer and the $\mathcal{K}$ graph embedding layer, where the $\mathcal{K}$ graph embedding layer also utilizes cross-embedding technique by combining label id^* with each component id_i. The proposed session embedding layer, named *Layer.SessionEmbed*, is designed as shown in Figure 4.4.

b. Proposed model

The proposed model is complex due to the integration of multiple improvements through experimental models to handle the multi-label problem with a large label space, including: (1) binary transformation; (2) graph representation; (3) graph embedding combined with label embedding. The suggested model has a binary structure as shown in Figure 4.5.

4.4 Experimental technique

4.4.1 Data normalization for training

The algorithm for data normalization for training is described in Algorithm 4.1 as follows for each session corresponding to graph K.

Thụt toán chun hóa dˇ liụ huˊn luyn đc mô ta nh sau cho mˇi phiên ´ng vˊi đ` thị $\mathcal{K}$ đc mô ta tại Thụt toán 4.1:

4.5 Experimental results

Figure 4.6 represents the aggregated results of $k \in [1, 5, 10, 20]$ in the same chart for convenient comparison. The results show that the embedded model with graph $\mathcal{K}$ ($GNN.Bin.K$) outperforms all other models using different neural networks.

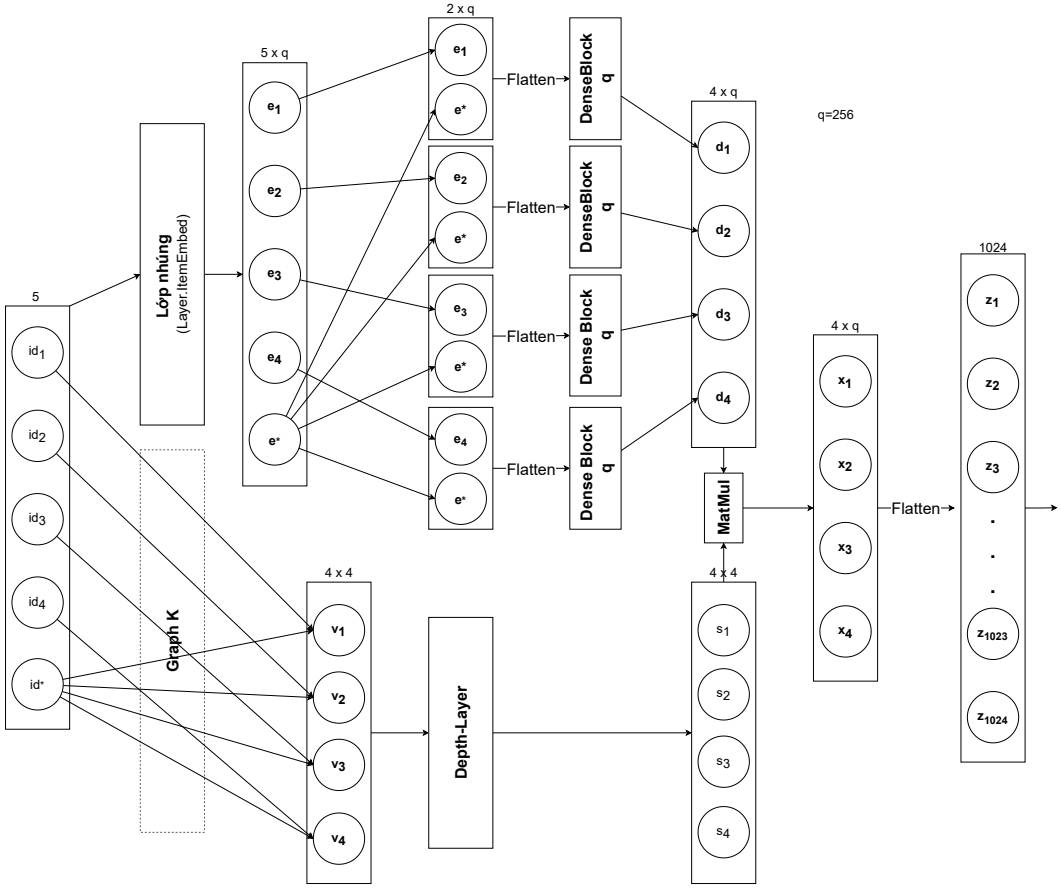


Figure 4.4: Session embedding layer with graph $\mathcal{K}$ (*Layer.SessionEmbed*)

Algorithm 4.1: NORM.GNN.Bin:
Data normalization for model GNN binary

Input:

$s = \{id_1, id_2, ..., id_c\}$

n_{id^*} //s´ lng đinh c`n c`n quan sát xem có phai là nhãn không

Output: Input data for traiing as x and output as y

$c' \leftarrow c$;
while $c' < 5$ **do**
  Append to session s a click $None$;
  $c' \leftarrow c' + 1$;
$Z \leftarrow id_5, id_6, ..., id_{c'}$;
$I \leftarrow$ tp ch´a n_{id^*} *đinh k`* cua các đinh ng˜u nhiên trong phiên, u tiên đinh có trong các $\{id_5, id_6, ..., id_{c'}\}$; //lu ý bo các đinh có giá trị là $None$.
for *đinh* $o \in I$ **do**
  $x^o \leftarrow \{v_1^o, v_2^o, v_3^o, v_4^o\}$ v´i v_i^o là trọng s´ cạnh n´i t` đinh id_i đ´n đinh o;
  $y^o \leftarrow \{0, 1\}$; //true label
  if $o \notin Z$ **then**
    $y^o \leftarrow \{1, 0\}$ //false label
$\mathbf{x} \leftarrow \{x^o | o \in I\} \in \mathbb{R}^{n_{id^*} \times 4}$;
$\mathbf{y} \leftarrow \{y^o | o \in I\} \in \mathbb{R}^{n_{id^*} \times 2}$;
return $\mathbf{x}$, $\mathbf{y}$;

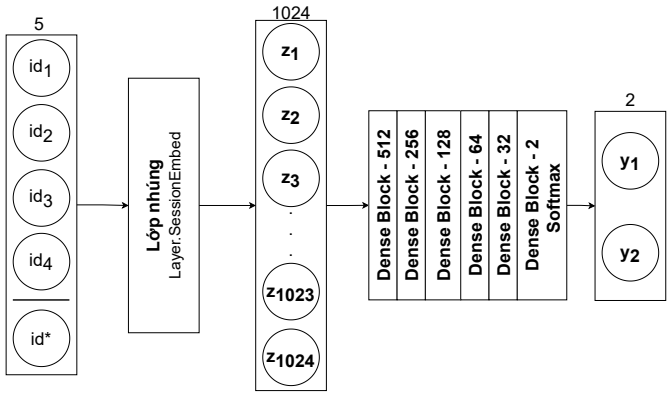


Figure 4.5: Binary embedding model with graph $\mathcal{K}$ ($GNN.Bin.K$)

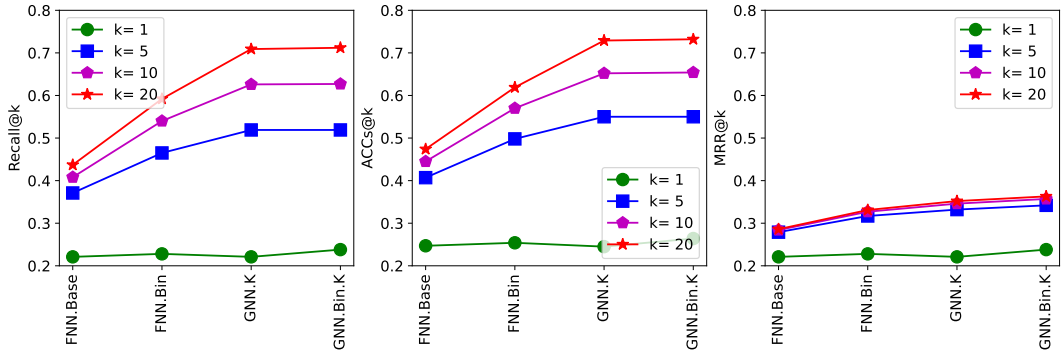


Figure 4.6: Comparison of model $GNN.Bin.K$ with others

4.6 Chapter conclusion

The graph embedding transformation is an important technique for building top-k recommendation systems, especially for problems related to representing user interactions when selecting products during work sessions in the form of graphs. By learning how to represent the graph into a new embedding space to capture the underlying features of session embedding vectors, the top-k recommendation model operates more effectively.

The experimental results in this chapter have demonstrated that the proposed model achieves good performance with three improvements: (1) binary model conversion, (2) the proposal of a graph embedding layer for session representation, and (3) the design of combining label embeddings.

Conclusion

1 General conclusion

The thesis researches using graphs to represent clickstream event data for online shopping, consisting of three graphs $\mathcal{G}$, $\mathcal{H}$, and $\mathcal{K}$ with varying complexities to evaluate the effectiveness of top-k recommendation models. With the graph representation of the data, the author proposes using Graph Neural Networks (GNN) as the recommendation model.

2 The achievements

Some observations on the achievements compared to previous studies:

✓ This thesis explores and proposes a deep neural network model for Problem 1 and a graph neural network for Problem 2. Problem 1 is a binary problem, while Problem 2 is a top-k multi-label problem.

✓ This thesis utilizes both the training and testing datasets from the original dataset, which consists of over 52 thousand products, or labels.

➜ Previous studies did not use a separate testing dataset, but instead extracted it from the training dataset.

✓ This thesis proposes and constructs a highly scalable GNN model that operates on graphs with over 52 thousand vertices. The thesis suggests designing graph $\mathcal{G}$ with the concept of neighboring nodes, graph $\mathcal{H}$ using edge weights as paths between nodes in a session, and graph $\mathcal{K}$ with edge weights as a c-dimensional vector.

➜ Some related studies present the inability to run the model with complete datasets, therefore they have to experiment with smaller datasets with even fewer labels.

✓ The proposed model achieves a *Recall*@20 of 0.712 and MRR@20 of 0.363.

➜ The result above is better than Kiewan's study with a *Recall*@20 of 0.691 and Tan's study with a *Recall*@20 of 0.680, and significantly better than Balázs Hidas' initial study with a *Recall*@20 of 0.632.

3 The main contributions

This thesis has the following main contributions:

- Using graphs to model customer shopping behavior through clickstream in work sessions, including both single and multiple relationships.
- The thesis proposes a deep neural network model for Problem 1 and a graph neural network for Problem 2. For Problem 2, the thesis suggests designing three graphs: $\mathcal{G}$, $\mathcal{H}$, and $\mathcal{K}$. For the $\mathcal{K}$ multi-relational graph, the thesis proposes using edge weights as a vector and also incorporating an additional linear deep learning layer to enable more effective learning of this graph by the GNN.
- The algorithm proposed embeds the graph to allow the GNN model to learn hidden attributes of user behavior during the selection of product categories in the current session.

LIST OF THE PUBLICATIONS RELATED TO THE DISSERTATION

1. **Khang Nguyen**, Anh V. Nguyen, Lan N. Vu, Nga T. Mai, and Binh P. Nguyen, "*An Efficient Deep Learning Method for Customer Behaviour Prediction Using Mouse Click Events*", Proceedings of the 11th National Conference on Fundamental and Applied Information Technology Research (FAIR'2028), 2018, pp.10, Vietnam, doi = 10.15625/vap.2018.0002.

2. **Khang Nguyen**, Nga T. Mai, An H. Nguyen, and Binh P. Nguyen, "*Prediction of Wart Treatment Using Deep Learning with Implicit Feature Engineering*", Soft Computing for Biomedical Applications and Related Topics, Springer International Publishing, 2020, pp.153–168, doi = 10.1007/978-3-030-49536-7_14.

3. **Nguyễn Tuấn Khang**, Nguyễn Viết Việt, Nguyễn Hải An, Mai Sơn, Mai Thúy Nga, và Nguyễn Việt Anh, "*Phát hiện giao dịch thẻ gian lận sử dụng mô hình học sâu*", hội thảo quốc gia lần thứ XXIII, 2020, pp.335

4. **Nguyễn Tuấn Khang**, Mai Thúy Nga, Nguyễn Hải An, và Nguyễn Việt Anh, "*Phân Tích Hành Vi Khách Hàng Với Mô Hình Mạng Học Sâu Đồ Thị*", hội thảo quốc gia lần thứ XXIV, 2021, p.439

5. **Nguyễn Tuấn Khang**, Nguyễn Tú Anh, Mai Thúy Nga, Nguyễn Hải An, và Nguyễn Việt Anh, "*Hệ Gợi Ý Mua Sắm Dựa Theo Phiên Làm Việc Với Mô Hình Mạng Học Sâu Đồ Thị*", chuyên san Các công trình nghiên cứu, phát triển và ứng dụng CNTT và Truyền thông, Bộ Thông tin và Truyền thông, 2022, vol. 2022, no. 02.

6. **Khang Nguyen**, Viet V. Nguyen, Nga T. Mai, An H. Nguyen, and Anh V. Nguyen, "*Behavioral gait recognition using hybrid Convolutional Neural Networks*", Journal of Computer Science and Cybernetics, 2023

7. **Khang Nguyen**, Nga T. Mai, An H. Nguyen, and Anh V. Nguyen, "*A Computational Model for Predicting Customer Behaviors Using Transformer Adapted with Tabular Features*", International Journal of Computational Intelligence Systems, vol. 16, no. 1, pp. 1–8, 2023, doi = 10.1007/s44196-023-00307-5.

8. **Khang Nguyen**, Anh T. Nguyen, Nga T. Mai, An H. Nguyen, and Anh V. Nguyen, "*Developing Advanced Product Recommendation System using Embedding Graph Neural Networks*", Applied Intelligence, Springer, 2023 (bài đang nộp)