Article Feed Recommendations Using Position-Aware Deep Cross Network

Mr. Dhata Muangrux

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Software Engineering
Department of Computer Engineering
Faculty Of Engineering
Chulalongkorn University
Academic Year 2023

การแนะนำบทความในฟีดโดยใช้ดีปครอสเน็ตเวิร์คที่รับรู้ตำแหน่ง

นายธตา เมืองรักษ์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2566

Thesis Title	Article Feed Recommendations Using Position-Aware Deep Cross Network
By	Mr. Dhata Muangrux
Field of Study	Software Engineering
Thesis Advisor	Assistant Professor Dr. PITTIPOL KANTAVAT

Accepted by the FACULTY OF ENGINEERING, Chulalongkorn University in Partial Fulfillment of the Requirement for the Master of Science

.... Dean of the FACULTY OF ENGINEERING

(Professor Dr. SUPOT TEACHAVORASINSKUN)

ธตา เมืองรักษ์ : การแนะนำบทความในฟีดโดยใช้ดีปครอสเน็ตเวิร์คที่รับรู้ตำแหน่ง. (Article Feed Recommendations Using Position-Aware Deep Cross Network) อ.ที่ปรึกษาหลัก : ผศ. ดร.พิตติพล คันธวัฒน์

แพลตฟอร์มโซเชียลมีเดียได้เปลี่ยนโฉมภูมิทัศน์ดิจิทัลอย่างมากในปัจจุบัน โดยมีฐานผู้ใช้ที่เพิ่มขึ้นอย่างต่อเนื่อง เนื่องจากแพลตฟอร์มเหล่านี้กลายเป็นสิ่งหนึ่งชีวิตประจำวันของคนทุกคน ความจำเป็นสำหรับระบบแนะนำเนื้อหาที่ตอบสนองความชอบส่วนบุคคลอย่างแท้จริงจึงมีความสำคัญอย่างยิ่ง ในขณะที่อัลกอริทึมการแนะนำปัจจุบันประสบความสำเร็จในการจัดการเนื้อหาตามความสนใจของผู้ใช้ แต่ก็มักมองข้ามอคติที่เกิดขึ้น โดยเฉพาะอย่างยิ่งอคติเชิงตำแหน่งที่ผู้ใช้มีส่วนร่วมกับเนื้อหาเนื่องจากตำแหน่งของเนื้อหามากกว่าความสนใจในเนื้อหานั้น อคตินี้เห็นได้ชัดเจนในระบบแนะนำในโซเชียลมีเดียทุกระบบ เพื่อแก้ไขปัญหานี้ จึงได้เสนอระบบแนะนำที่คำนึงถึงตำแหน่งของเนื้อหากับอัลกอริทึม Deep and Cross ซึ่งเรียกโมเดลอย่างเหมาะสมว่า 'Position-Aware DCN' โมเดลที่เสนอนี้มีจุดมุ่งหมายเพื่อแนะนำเนื้อหาที่แท้จริงและไม่ลำเอียงมากขึ้น โดยผู้ใช้จะได้รับเนื้อหาที่สอดคล้องกับความสนใจของพวกเขาและไม่ได้รับอิทธิพลจากตำแหน่งในฟีด โดยการประเมินที่ได้ดำเนินการบนชุดข้อมูลโซเชียลมีเดียของไทยแสดงให้เห็นว่าโมเดลที่เราเสนอนำเสนอมีประสิทธิภาพการแนะนำอย่างเห็นได้ชัดเมื่อเทียบกับระบบแนะนำแบบดั้งเดิม ซึ่งทำให้ประสบการณ์ดิจิทัลเป็นประสบการณ์ที่เข้าถึงผู้ใช้งานมากขึ้น โดยผู้จัดทำได้นำเสนอโมเดลเป็นการเชื่อมต่อโปรแกรมประยุกต์หรือ API ในการใช้งานแบบออนไลน์ โดยแสดงฟังก์ชันการใช้งานของโมเดลและการทำงานร่วมกันกับเว็บแอปฟรอนต์เอนด์

สาขาวิชา	วิศวกรรมซอฟต์แวร์	ลายมือชื่อนิสิต
		....
ปีการศึกษา	2566	ลายมือชื่อ อ.ที่ปรึกษาหลัก
		....

6372052021 : MAJOR SOFTWARE ENGINEERING

KEYWORD: Recommendation System, Deep and Cross Network, Position-Aware, Position Debiasing

Dhata Muangrux : Article Feed Recommendations Using Position-Aware Deep Cross Network. Advisor: Asst. Prof. Dr. PITTIPOL KANTAVAT

The presence of social media platforms has recently transformed the digital landscape, with an ever-increasing user base. As these platforms become central to daily life, the need for recommendation systems that genuinely cater to individual preferences has never been more paramount. While current recommendation algorithms excel at curating content based on user interests, they often overlook inherent biases, notably the positional bias where users engage with content due to its placement rather than its inherent relevance. This oversight is particularly evident in every social media recommendation system. Addressing this challenge, we proposed a position-aware methodology within the Deep and Cross framework, aptly termed 'Position-Aware DCN.' By explicitly accounting for positional preferences, our proposed model aims to provide more genuine, unbiased recommendations, ensuring that users are presented with content that aligns with their interests and is not just influenced by its position in the feed. Evaluations conducted on Thai social media datasets reveal that our proposed model offers a marked improvement over traditional recommendation systems, underscoring its potential to foster a more user-centric digital experience. The author also implements the proposed model as an application programming interface (API) in an online deployment format by showcasing its functionality and seamless integration into the front-end web app.

Field of Study:	Software Engineering	Student's Signature ....
Academic Year:	2023	Advisor's Signature ....

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to my advisor, Asst. Prof. Pittipol Kantavat, for his support and guidance throughout my thesis journey. Dr. Ekapol Chuangsuwanich,for his deep understanding of recommender systems. I am also grateful to Ltman for providing me with the dataset from Blockdit used in my thesis. Finally, I would like to thank my family and friends for their love and support throughout my studies. Their encouragement and belief in me were the most important thing in helping me achieve my goals.

Dhata Muangrux

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Background

In the last decade, social media platforms have experienced rapid growth, becoming an integral part of our daily lives. According to a report by Demandsage, as of 2023, there are over 4.9 billion social media users worldwide, projected to increase to almost 5.42 billion by 2025 [1]. This rise underscores the transformative power of social media, reshaping communication, information dissemination, and even commerce.

Recommender systems are central to providing a personalized user experience on these platforms. These systems sift through vast amounts of data, analyzing user behaviors, preferences, and interactions to curate and suggest content that aligns with individual interests. Historically, recommender systems were implemented using methods like collaborative filtering or content-based filtering [2] . While effective, these traditional methods had limitations, such as the cold start problem and scalability issues. However, with the advent of advanced computational techniques, newer implementations have emerged, offering more nuanced and scalable solutions.

The integration of deep learning techniques, particularly neural networks, has significantly shifted the landscape of recommender systems. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), with their ability to model complex non-linear relationships, have found applications in recommendation systems, enhancing the granularity and adaptability of recommendations [3]. By leveraging embeddings and deep architectures, these models can capture intricate patterns in user data, offering a dynamic recommendation experience that evolves with changing user preferences.

The development of recommender systems took a turn with the introduction of neural networks, leading to the creation of the "Wide and deep" learning model [4]. This model smartly combines the strengths of linear models and deep neural networks, balancing between memorizing and generalizing feature interactions. Following this, the "Deep & Cross" network [5] came into play, advancing the field by adding a cross-network that applies feature crossing at each layer, allowing it to effectively learn bounded-degree feature interactions, improving predictive performance in various recommendation situations while keeping the benefits of deep networks in automated feature engineering. This progress highlights a significant step towards finding the right balance between expressiveness and generalization in recommender systems.

However, a persistent challenge that has emerged is positional bias [6]. Users, influenced by the layout and presentation, often exhibit a propensity to interact with content based on its placement, leading to a skewed feedback loop and potentially sub-optimal content recommendations. For instances, we have observed this scenario with dataset from one of Thai social media platform "Blockdit". The dataset contains user and article information, and their interactions. As shown in Fig. 1, where we illustrate the relationship between the position of articles and their respective total clicks for the first 100 positions.

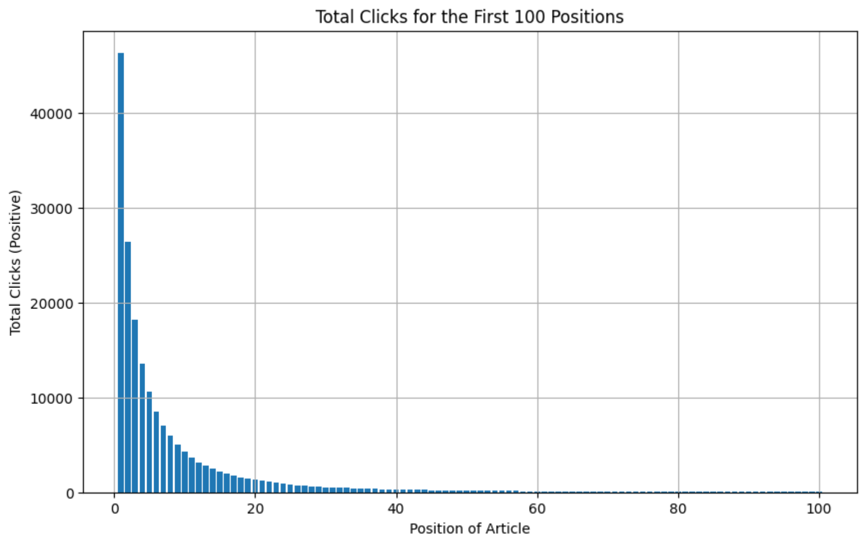


Figure 1 Total clicks for the first 100 positions

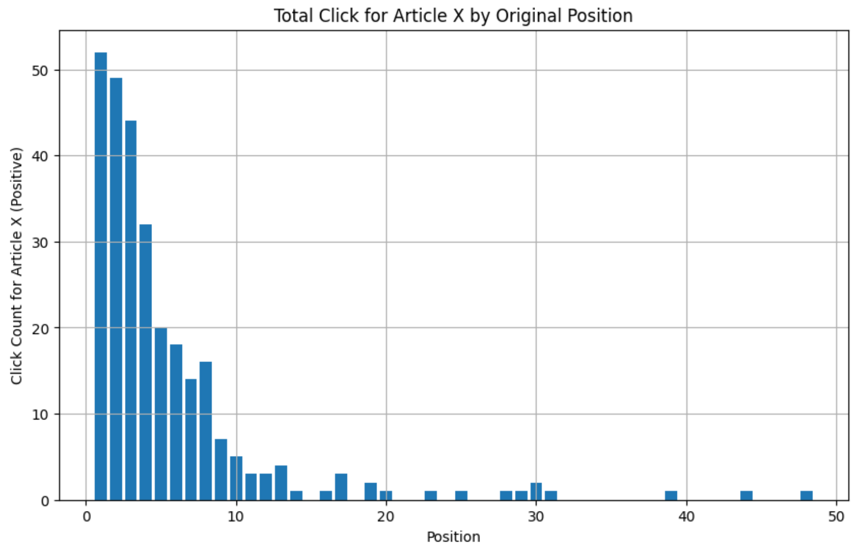


Figure 2 Mean CTR for article by their positions in the feed

It is evident that articles in the top positions receive significantly more clicks, with the number of clicks decreasing exponentially as the position increases. Similarity, in Fig. 2, where we sample the data and explore the correlation between each article position in the feed and its click-through rate (CTR). We observe that the CTR drops dramatically with its position. These imply that a user clicks on an item not only because they favor it but also because it is in a good position.

To address this challenge, we present an article feed recommendation using Position-Aware Deep Cross Network (Position-Aware DCN). By integrating positional embeddings into the DCN architecture, our method aims to mitigate the effects of position bias, ensuring that content recommendations are both relevant and unbiased. Moreover, we broaden the scope by implementing the proposed model as an API in an online deployment format. Through showcasing the model's functionality and its seamless integration as an API, our objective is also to enrich the collective understanding of deploying recommender systems in real-world scenarios.

1.2 Research Objectives

1) To develop a position-debiasing article recommendation system.
2) To develop an application programming interface (API) for position-debiasing article recommendation system.

1.3 Research Scopes

1) Using python on developing position-debiasing article recommendation system.
2) Using the dataset has been provided by LTMAN Co., Ltd., the service provider of the Blockdit application, a Thailand social media platform which allowing users to read articles or content from other members.
3) Utilizing user-article interaction data for model creation.
4) Developing an API as an implementation demo for position-debiasing article recommendation system.

1.4 Research Methodologies

1) Study and research related theories for position-debiasing article recommendation system.

2) Study and research literature review.
3) Study python and required libraries for developing position-debiasing article recommendation system.
4) Conduct exploratory data analysis.
5) Conduct data preprocessing.
6) Build position-debiasing article recommendation system.
7) Test and evaluate research results.
8) Summarize the results.
9) Build an API for position-debiasing article recommendation system.
10) Prepare and submit an academic paper.
11) Make conclusions and produce thesis.

1.5 Research Outcomes

1) To be able to develop a position-debiasing article recommendation system.
2) To be able to develop an API for position-debiasing article recommendation system.

An activity diagram of research methodologies is displayed in Fig. 3.

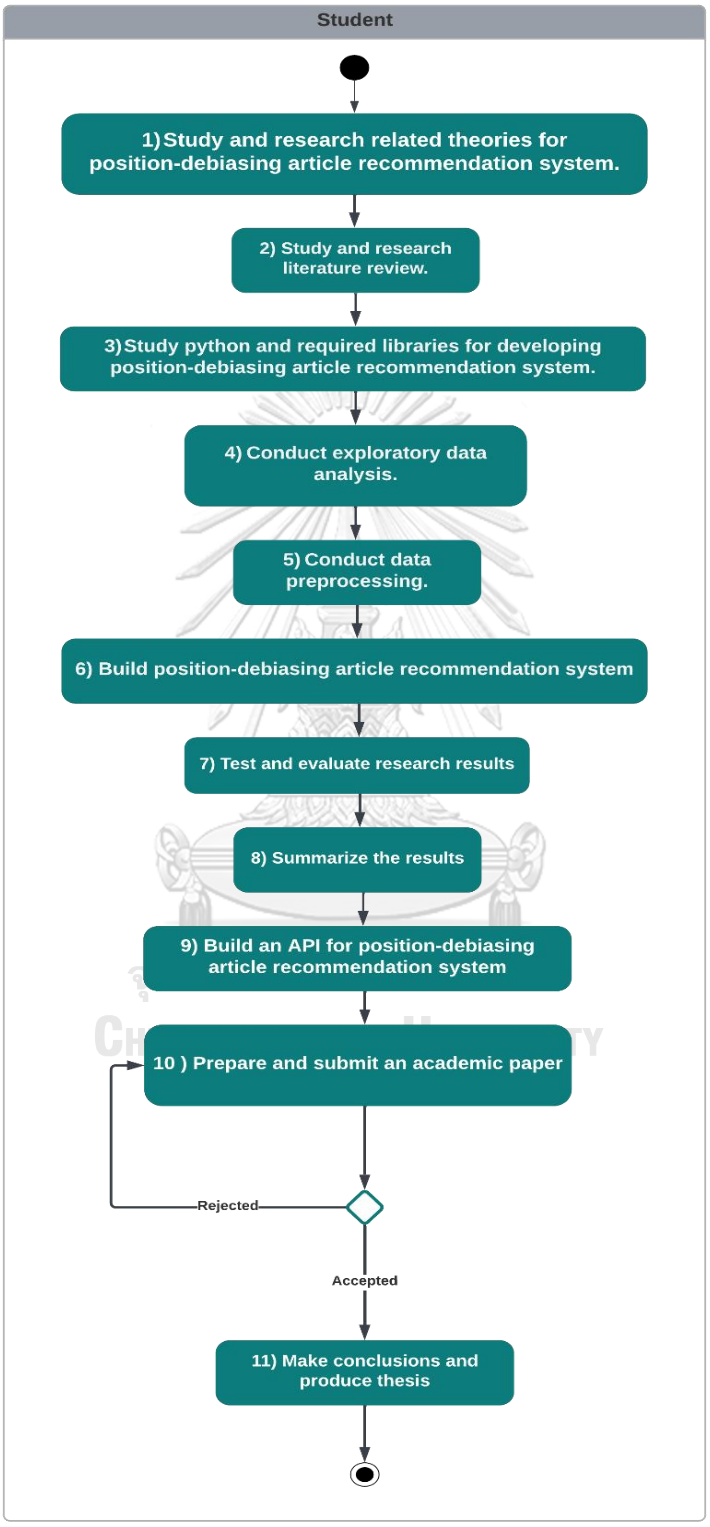

Student
1) Study and research related theories for position-debiasing article recommendation system.
2) Study and research literature review.
3) Study python and required libraries for developing position-debiasing article recommendation system.
4) Conduct exploratory data analysis.
5) Conduct data preprocessing.
6) Build position-debiasing article recommendation system
7) Test and evaluate research results
8) Summarize the results
9) Build an API for position-debiasing article recommendation system
10) Prepare and submit an academic paper
Rejected
Accepted
11) Make conclusions and produce thesis

Figure 3 Research Methodology Activity Diagram

Chapter 2

Related Work

In this section, we delve into research that has contributed to the understanding and implementing of the two-tower architecture and position-debiasing techniques. The two-tower architecture involves the parallel processing of user and item information through separate neural network towers, enabling more effective learning of user-item interactions. Additionally, we explore the research dedicated to addressing the issue of position bias in recommender systems.

2.1 Research in two-tower recommender systems

In recommender systems, deep neural networks have emerged as a pivotal tool for building recommender systems due to their ability to capture complex nonlinear relationships between users and items. DNNs have consistently outperformed traditional collaborative filtering methods, such as matrix factorization, in various recommendation tasks. He et al. [7] introduced a general framework for applying DNNs to recommendation tasks. NCF utilizes a multilayer perceptron (MLP) to model the interactions between users and items, achieving state-of-the-art performance on several benchmark datasets. Cheng et al. [4] proposed a hybrid DNN architecture combining a wide layer and a deep layer to capture low-order and high-order interactions between users and items. W&D achieved significant performance improvements over traditional methods and became widely adopted in recommender systems.

Building upon the extensive research on DNNs in recommender systems, researchers have leveraged this innovative approach to craft state-of-the-art algorithms. Among these, one prominent architecture that has gained significant attention is "DeepFM."

Guo et al. [8]. proposed a novel neural network architecture called DeepFM, which operates by integrating the capabilities of factorization machines (FMs) and Deep Neural Networks (DNNs) to capture a spectrum of feature interactions in recommendation tasks adeptly. The model excels at comprehensively addressing low-order and high-order interactions, contributing to remarkable advancements in performance compared to its predecessors. Its widespread adoption as a go-to model for Click-Through Rate (CTR) prediction in recommender systems underscores its efficacy and impact in practical applications. The DeepFM architecture is displayed in Fig. 4.

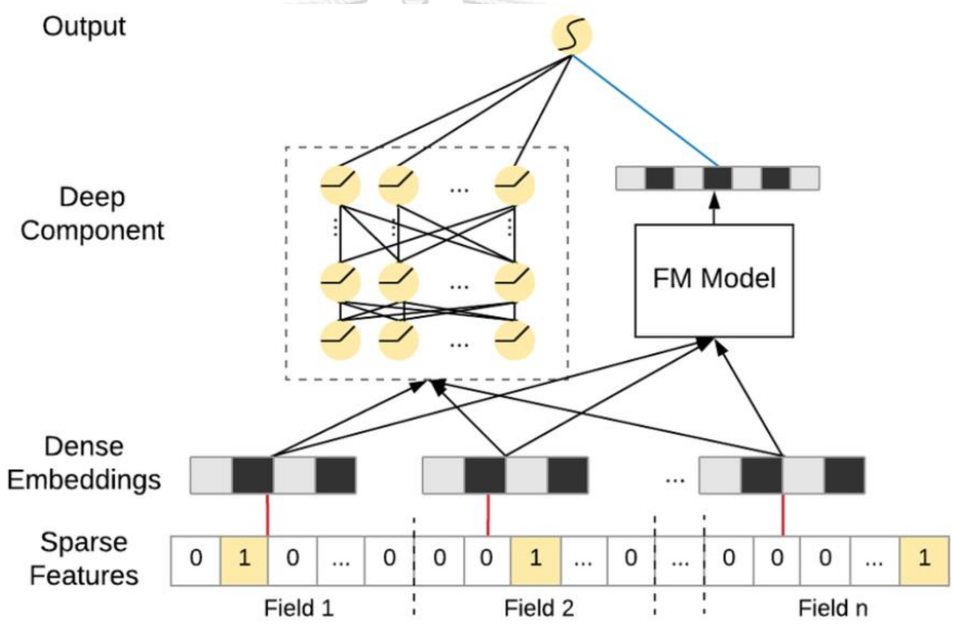


Figure 4 DeepFM architecture (Source: Huifeng Guo, Ruiming Tang. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction in Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)

The author uses this research as a foundational exploration, and the insights gained from DeepFM's success lay the groundwork for a better understanding of recommendation system architectures.

2.2 Research in positional-debiasing recommender system

In addressing position debiasing, deep neural networks ensure that the recommendations generated reflect genuine user preferences rather than artifacts of positional biases. Xingyuan et al. [10] propose a Deep Presentation Bias Integrated Framework (DPBIF), as seen in Fig. 5, that considers overall presentation bias, including context. DPBIF introduces a presentation block into user behavior sequences and predicted target items to personalize the integration of presentation bias into CTR prediction. It also avoids the independence assumption and estimates multiple integrated CTRs for each item under different presentations. These CTRs transform the ranking problem into an item-to-position assignment problem, optimized using the Kuhn-Munkres (KM) algorithm. Offline experiments and online A/B tests demonstrate the effectiveness of DPBIF.

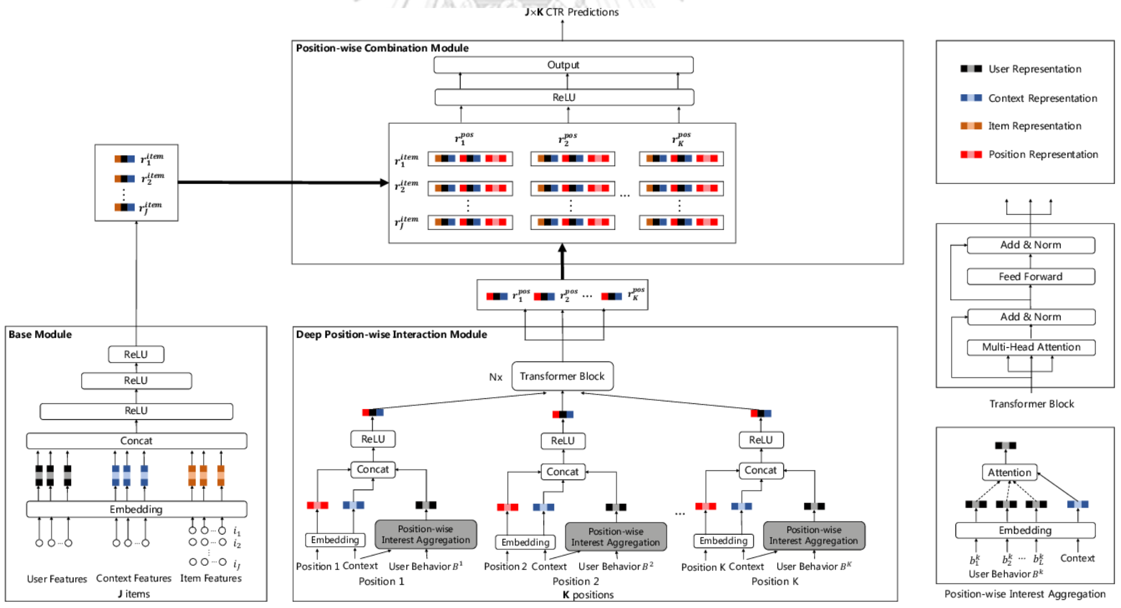


Figure 5 DPIN architecture (Source: Jianqiang Huang, Ke Hu. Deep Position-wise Interaction Network for CTR Prediction in Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval)

H, Jianqiang et al. [9] introduce a Deep Position-wise Interaction Network (DPIN) to address the bias problem in click-through rate (CTR) prediction in online advertising and recommender systems. It aims to efficiently combine all candidate items and positions for estimating CTR at each position, achieving consistency between offline and online models. The DPIN model utilizes a two-layer transformer with self-attention and is trained using stochastic gradient descent with actual position features and cross-entropy loss. It has been deployed in a sponsored search advertising system and shows statistically significant improvement over a highly optimized baseline in a rigorous A/B test.

Positional debiasing in recommender systems addresses the inherent bias that emerges due to the positional influence on user interactions. Users often interact more with items at certain positions (e.g., the top of a list) not because of their relevance but due to their visibility and accessibility, which can introduce bias into the learned models. There are several ways to mitigate position bias [12]. One is introducing randomness while collecting click data. Because multiple items can appear in the same position, we can log which items performed better and train our models accordingly. Another approach is to use the measured position bias to derive logged data. The Google paper [13] used inferred position bias to train models optimized on inverse propensity weighted precision.

G, Huifeng et al. [11] propose the Position-bias Aware Learning framework (PAL), which models position bias during offline training and conducts online inference without position information. PAL utilizes a two-stage training process. In the first stage, as seen in Fig. 6, a position-aware model is trained using historical data, capturing the relationship between item position and CTR. In the second stage, a position-independent model is trained using the same data, excluding position information. During online inference, the position-independent model predicts CTR, effectively removing position bias. Experimental results demonstrate that PAL significantly outperforms existing methods in offline and online settings.

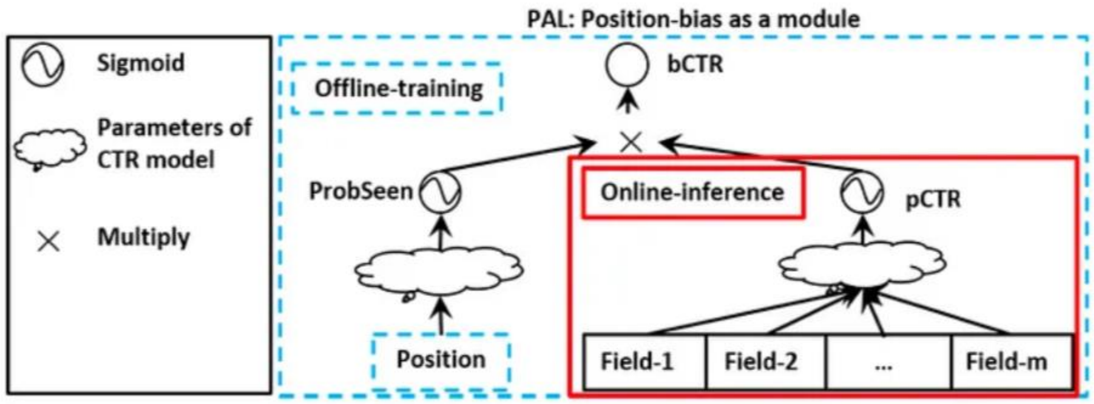


Figure 6 PAL architecture (Source: Guo, H.a.Y. PAL: a position-bias aware learning framework for CTR prediction in live recommender systems in Proceedings of the 13th ACM Conference on Recommender Systems)

Based on these research studies, integrating deep neural network models in mitigating bias, especially positional bias, has shown powerful improving performance in recommender systems. Therefore, this approach will be applied in this research.

Chapter 3

Background

In this section, we provide knowledge for implementing our proposed approach. The proposed approach leverages a collaborative filtering recommendation system, a two-tower recommendation system with deep and cross neural network framework, a position bias in the recommendation system, and the application of positional debiasing in the recommendation system.

3.1 Recommender System

Recommender systems, also known as recommendation systems or engines, are information filtering tools designed to predict and suggest items or content a user might be interested in based on their preferences and historical interactions. As seen in Fig. 7, These systems are pivotal in addressing the information overload problem by delivering personalized recommendations that enhance user experience and engagement in various domains, such as e-commerce, entertainment, social media, and more.

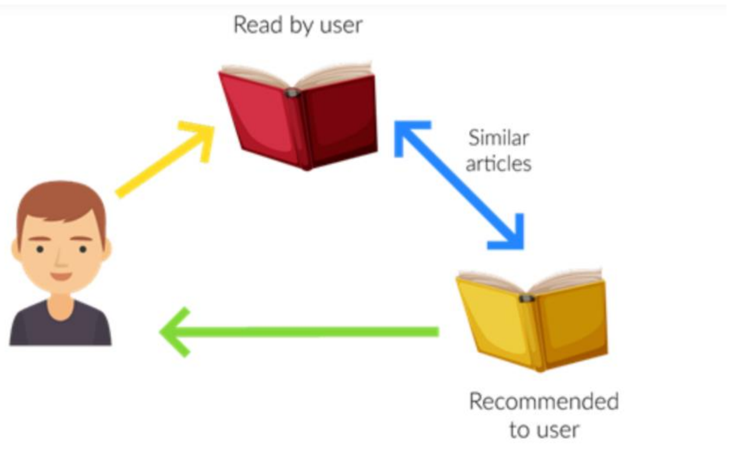


Figure 7 An example of recommender system concept

Recommender systems are trained to understand the preferences, previous decisions, and characteristics of people and products using data gathered about their interactions. These include impressions, clicks, likes, and purchases. Because of their capability to predict consumer interests and desires on a highly personalized level, recommender systems are a favorite with content and product providers. They can drive consumers to think about any product or service that interests them, from books to videos to health classes to clothing.

To enable the delivery of personalized recommendations, a recommender system necessitates access to user information, often referred to as a user profile or user model. In the context of our example, such as a book recommendation in Fig. 3, this involves capturing data on a user's preferences, notably the books they have previously read. The user profile forms the nucleus of every recommender system, serving as the foundation for predicting which additional books might align with a user's interests.

The method through which a recommender system acquires user information varies across recommendation techniques. Regardless of the approach, the collection and maintenance of user profiles remain integral. User preferences can be obtained implicitly through observing and analyzing user behavior, such as their interactions with books. Alternatively, explicit information can be sought by directly asking users about their preferences. Recognizing that the core concepts underlying recommender systems hinge on fundamental techniques is crucial. One is collaborative filtering. This technique embodies distinct strategies for leveraging user profiles to generate personalized recommendations, contributing to the diversity and effectiveness of recommender systems.

3.2 Collaborative Filtering

Collaborative filtering (CF) represents a cornerstone in recommender system design, offering a robust method for predicting user preferences based on historical interactions. At its core, collaborative filtering operates on the premise that users with similar preferences will likely share familiar tastes. Compared to other recommendation techniques, collaborative filtering does not require an intricate understanding of item characteristics. Picture a scenario, as seen in Fig. 8, where you and your close friends often watch movies together. If, during a meal, your friend prefers a specific type of movie, you might choose a similar one based on your shared past movie preferences. In this manner, collaborative filtering draws upon the collective tastes of users with similar preferences akin to the dynamics observed in real-life situations.

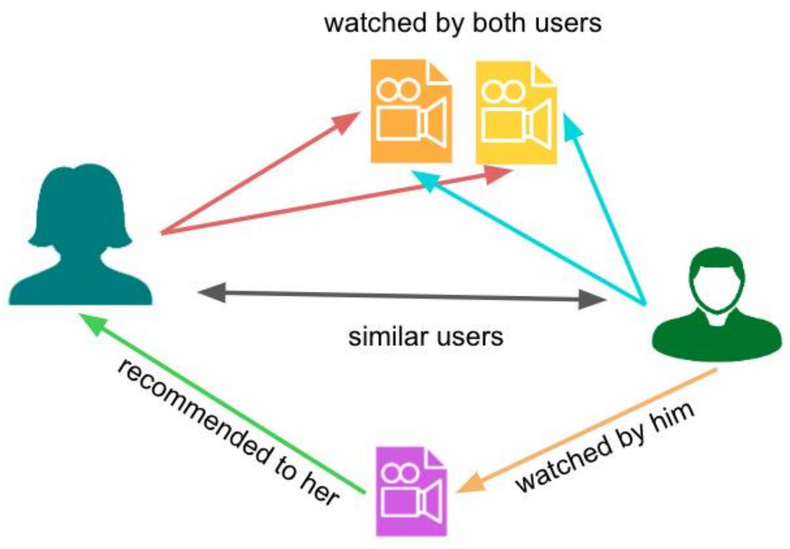


Figure 8 An example of collaborative filtering scenario

Collaborative filtering, relying on user behavior, boasts several advantages over content-based filtering. One significant strength lies in its effectiveness without requiring additional development work, especially in scenarios with large user databases. The recommendation list remains potent as it leverages the behavioral patterns of users, fostering diversity in recommendations. For example, collaborative filtering might connect seemingly unrelated items if you and your friend are interested in hiking. If your friend has data on camping gear, the system could recommend camping equipment based on your shared interest in hiking.

However, collaborative filtering has its challenges. One notable drawback is the cold start problem, particularly affecting new users, items, or systems. In such instances, where information about the user is limited or ratings for new items are limited, collaborative filtering needs to provide accurate recommendations. Scalability is another concern, especially when matching target users with others who exhibit similar behavior. The technique's efficiency relies heavily on the number of users; each user must interact with enough items to maintain effectiveness. Lastly, the sparsity of data poses a critical challenge, as users often need to rate the majority of items, leading to sparse ratings.

To illustrate, consider a scenario where a user explores a new hobby, such as photography. The recommendations may be less accurate if the collaborative filtering system lacks sufficient data on this new interest. Additionally, as more users join the platform, the challenge of matching behaviors and maintaining efficiency becomes increasingly complex. The sparsity issue arises when users explore niche interests, resulting in limited ratings for various items related to that specific interest.

The Two-Tower model emerges as a solution to enhance recommendation system performance from the limitation mentioned above. This model addresses the cold start problem by leveraging auxiliary information to understand better and recommend items for new users or items with limited interaction history. Scalability concerns are mitigated through the model's ability to efficiently handle many users and items by optimizing the recommendation process. Moreover, the Two-Tower model tackles the sparsity of data challenge by incorporating auxiliary features and embeddings, providing a more comprehensive understanding of user preferences, and ensuring more accurate and diverse recommendations. As a result, the Two-Tower model represents a promising advancement in recommender systems, offering a solution to the inherent limitations of collaborative filtering.

3.3 Deep Neural Network

Deep Neural Networks (DNNs) [14] represent a powerful class of machine learning models that have evolved from the broader field of neural networks. The term "deep" in DNNs refers to their characteristic depth, indicating the presence of multiple hidden layers between the input and output layers. As seen in Fig. 9, These hidden layers enable DNNs to learn intricate hierarchical representations of features from the input data. Each layer contains nodes, or neurons, connected by weights adjusted during training, allowing the network to capture complex patterns and relationships within the data.

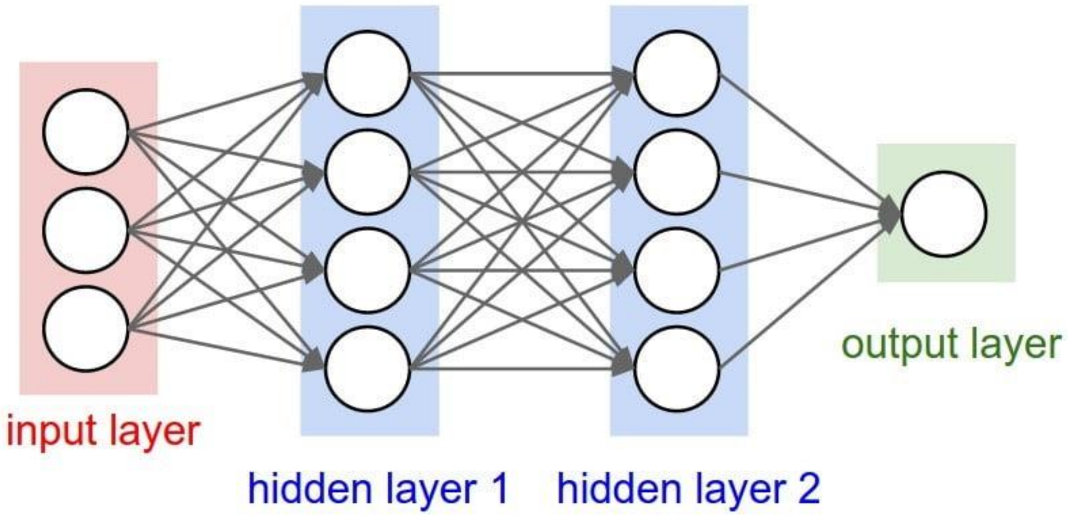


Figure 9 Example of deep neural networks

In DNN, each layer contains a given number of units (neurons) that apply a certain functional transformation to the input. These types of models can approximate the behavior of any function. the deep neural network can be represented as:

$$y_i^l = f \sum_{j+1}^{J} w_{i,k}\, x_k + b_i \tag{1}$$

Where, the output y of a unit i in layer l is related to the output x of the earlier layer k with j outputs through a set of weights $w_{i,k}$, a bias b_i and a non-linear activation function f.

3.4 The “Two-Tower” Model

The "Two-Tower" model is an innovative approach in recommender systems designed to address the limitations of collaborative filtering. This model is characterized by its architectural framework, consisting of two distinct "towers" or neural networks: one focuses on encoding user information, and the other on encoding item information. By employing this dual-tower architecture, the model aims to capture intricate patterns in user-item interactions, providing a more nuanced understanding of user preferences.

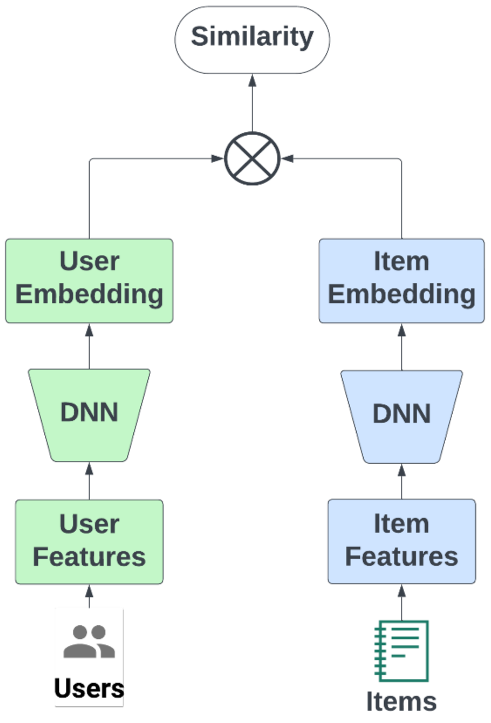


Figure 10 The two-tower architecture

The Two-Tower model leverages a dual-structure architecture to facilitate efficient [15] and nuanced learning of user-item interactions to learn more expressive and meaningful representations of users and items by handling their features separately before fusing them for prediction, thereby enhancing the system's ability to make accurate and personalized recommendations. The model is visualized as two towers, as seen in Fig. 10, where each tower is a neural network that processes and learns representations for users and items independently. The "user tower" ingests user-related features (such as user ID, historical interactions, and demographic information) and processes them to generate a fixed-size embedding vector that represents the user. Simultaneously, the "item tower" processes item-related features (such as item ID, category, and other properties) to produce an embedding vector for the item. Subsequently, the embeddings from both towers are combined, often through dot product or concatenation, and passed through additional layers (if present) to predict the interaction (such as click, purchase, or rating) between the user and the item.

3.3.1 Embedding Representation

Embeddings serve as numerical representations that capture the latent features of users and items in a shared space. These embeddings are learned during the training process and enable the model to uncover complex patterns in user-item interactions. In the context of the Two-Tower model, embeddings are utilized to map users and items into a common latent space where their preferences and characteristics are represented as vectors.

The equation for the embedding process in the Two-Tower model involves projecting users and items into a shared embedding space *R*. Let *U* represent the set of users, *I* denote the set of items, and E denote the embedding space dimension. The user embedding u_i for a user *i* and the item embedding v_j for an item *j* are calculated as follows:

$$u_i = UserEmbedding(i) \in R^E \quad (2)$$

$$v_j = ItemEmbedding(j) \in R^E \quad (3)$$

Where $UserEmbedding(i)$ and $ItemEmbedding(j)$ are functions that map users and items to their respective embeddings in the shared latent space. The resulting embeddings, u_i and v_j, from Equation 1 and 2 represent the users and items in a continuous vector space, where the proximity of vectors indicates similarity in preferences.

The Two-Tower model further combines these embeddings to calculate a user-item interaction score, often used for making recommendations. The interaction score s_{ij}, between user *i* and item *j* is computed to Equation 3 using the dot product of their embeddings:

$$s_{ij} = u_i \cdot v_j \quad (4)$$

The dot product captures the similarity between the user and item embeddings. A higher dot product implies a stronger user-item interaction score, indicating a higher likelihood that the user would be interested in the item.

In terms of similarity, Cosine similarity is a metric that measures the cosine of the angle between two non-zero vectors. It is frequently employed to assess the similarity between vectors. The formula for cosine similarity between user embedding u_i and item embedding v_j are calculated as follows:

$$cosine\ similarity(\theta) = \frac{u_i \cdot v_j}{\|u_i\| \cdot \|v_j\|} \quad (5)$$

Where $u_i \cdot v_j$ represents the dot product of vectors and $\|u_i\| \cdot \|v_j\|$ denote the Euclidean norms (magnitude) of vectors respectively. The resulting cosine similarity ranges from -1 (completely dissimilar) to 1 (completely similar), with 0 indicating orthogonality.

Embeddings in the Two-Tower model provide a compact and expressive representation of users and items in a shared latent space. The model learns these embeddings through training, capturing complex relationships and patterns in user-item interactions, ultimately enabling more effective and personalized recommendations.

3.4 Deep and Cross Network

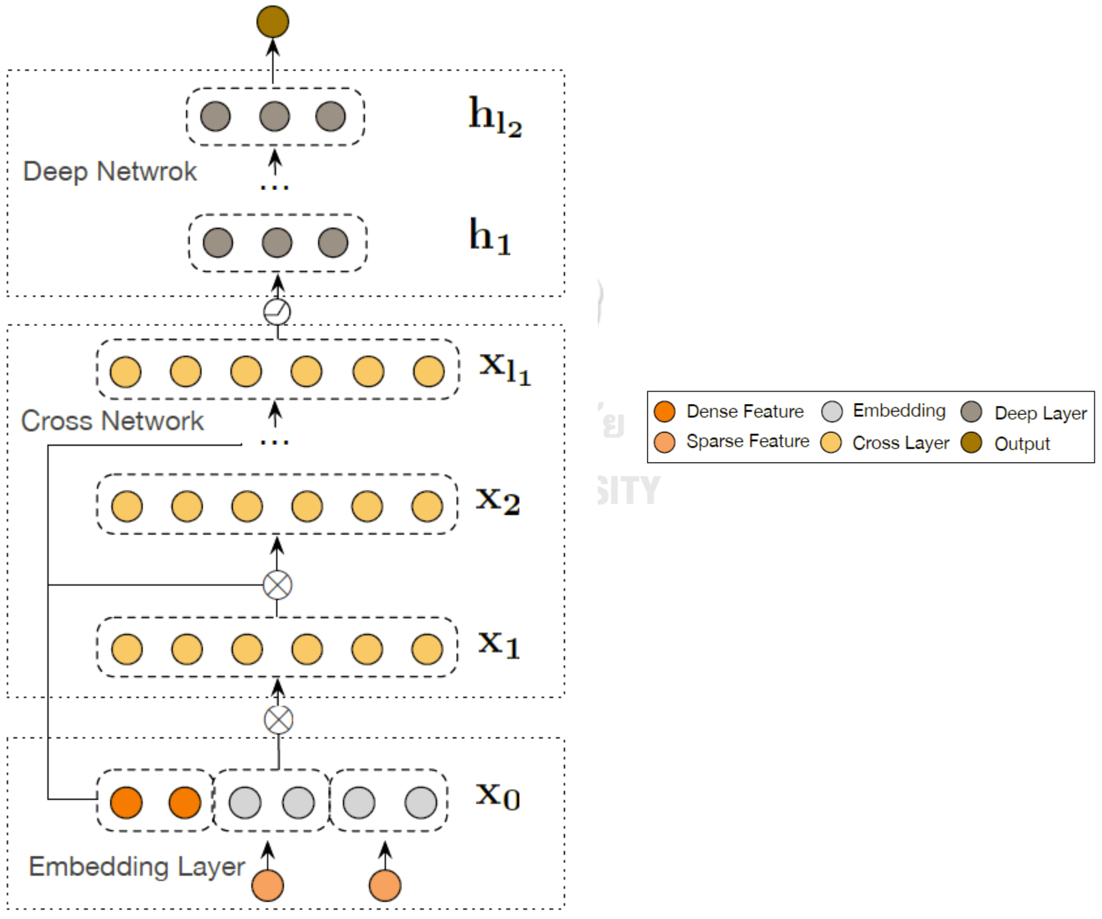


Figure 11 Deep and Cross network architecture

The Deep and Cross Network (DCN) [5] is a two-tower neural network architecture designed to model low- and high-order feature interactions in tabular data, commonly used in tasks like click-through rate prediction. The DCN, as seen in Fig, 11, combines the strengths of deep neural networks and explicit cross-feature generation to capture intricate patterns in the data.

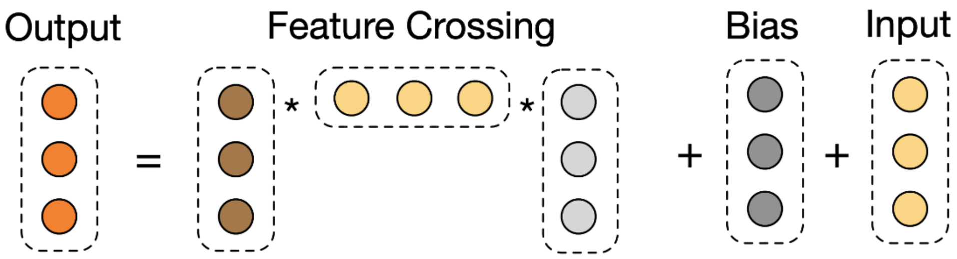


Figure 12 Cross Network Operation

The Cross Network is designed to learn bounded-degree feature interactions explicitly. It takes the original input features and applies a series of cross-transformations, where each transformation captures specific feature interactions. The idea is to allow the model to learn which features must be combined (crossed) to improve the prediction. Mathematically, the cross operation, as seen in Fig. 12, can be represented as:

$$x_{l+1} = x_0 * (x_l * w_l + b_l) + x_l \tag{6}$$

Where, x_{l+1} is the output from l^{th} layer, w_l is the weight vector, and b_l is the bias vector.

The Deep Network is a stack of fully connected layers, like traditional feed-forward neural networks. It's designed to capture low-order feature interactions and can generalize well from the input features. The deep component allows the model to learn intricate patterns and representations from the data.

The outputs of the Cross Network and the Deep Network are concatenated and passed through a final stack of fully connected layers to produce the prediction. This combination ensures that the model benefits from both explicit high-order feature interactions and deep representations.

Chapter 4

Proposed Method

The proposed methodology of this study is mitigating position bias by combining positional features as one of the input feature vectors on the Deep and cross-network. The positional feature is embedding on a) cross-network specifically and b) deep network specifically. These positional feature embeddings help the model learn how position affects the user's preference, thus ensuring that content recommendations are both relevant and unbiased. As the positional feature embedding is modeled in the offline training, a feature should also be included in prediction or online inference. However, position information is unavailable when prediction is performed. To resolve this problem, the decision is to select a position for all items as the value of the position feature.

The architecture of the proposed model is explained in Fig. 13. The model leverages both deep and cross networks in parallel to understand deeper relationships of the features while also learning from the embedding positional features of the item for each user. In user context, user embedding features include gender, age, the number of articles read in the last 14 days, the ratio of article categories read by the user, and the age of the articles on the day they were read. Regarding items, the embedding article features its category, type, length, and the number of user engagements the previous day. Subsequently, all the feature input undergoes deep and cross network embedding, transforming sparse features into embeddings and concatenating them with dense features. These are then separately channeled into the deep and cross networks before being merged in the final combination output layer to obtain the final result. To observe the impact of positional features in the Deep & Cross model, the feature will be introduced as a sparse feature separately, a dense feature separately, and both feature types. Then, an evaluation will be conducted.

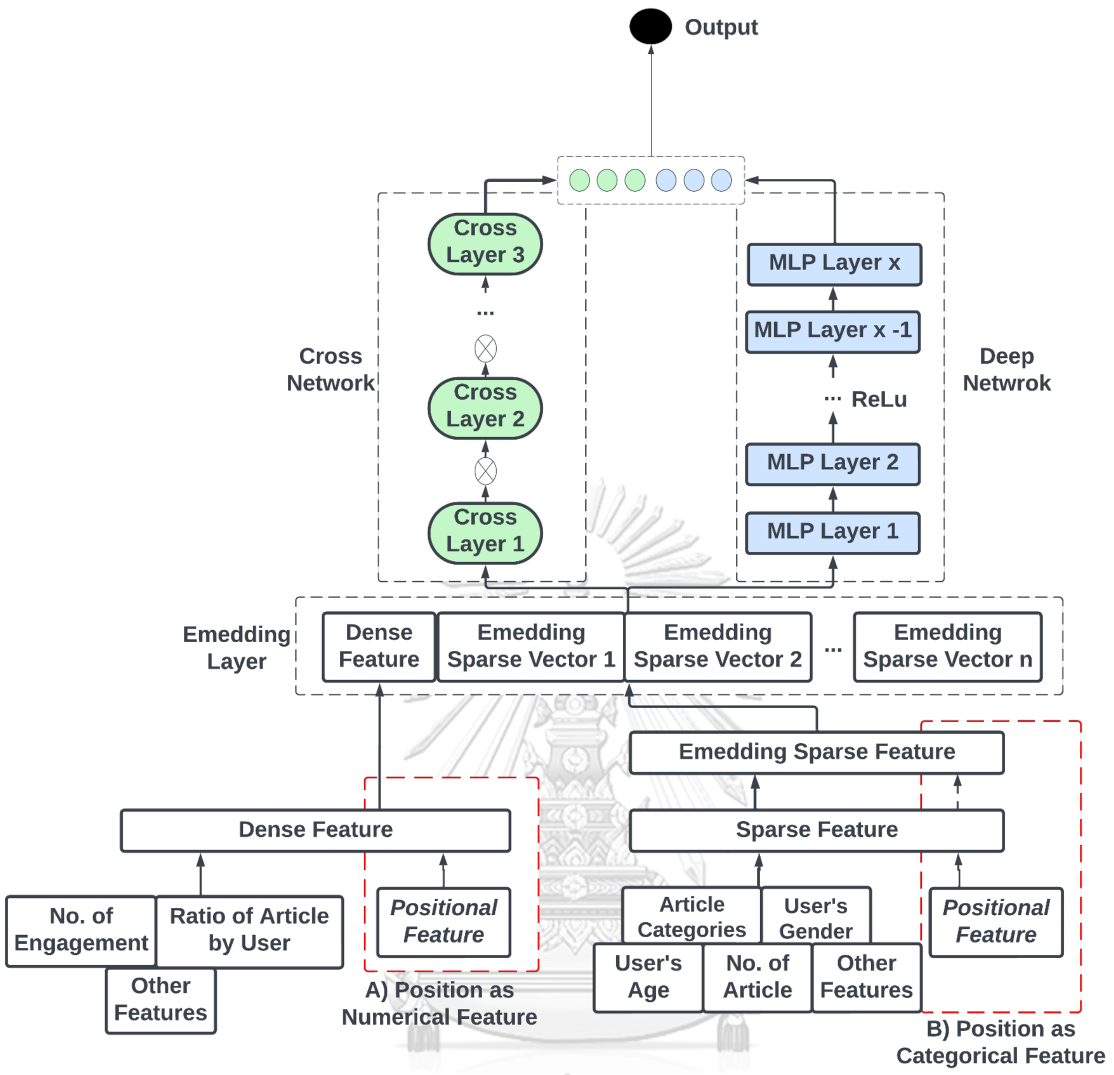


Figure 13 The proposed architecture framework

4.1 Input and Embedding Layer

We classified all the features of the raw dataset into two types: categorical features and numerical features. The categorical features were transformed into a low-dimensional space using embedding for dimensional reduction. To avoid being affected by the dimension, numerical features in the model were scaled into a fixed range between zero and one using normalization or min-max normalization. Finally, we stacked all the above into one vector and fed it to the proposed model simultaneously.

4.2 Cross Network Layer

The cross-network layer consists of x layers, where each cross layer could be calculated as follows:

$$x_{j+1} = x_0 x_j^T w_j + b_j + x_j = f(x_j, w_j, b_j) + x_j \tag{7}$$

Where, $j \in [0, 1, 2, \ldots., m-1]$, $m \in [1, 2, 3, \ldots.]$; x_{j+1} and x_j represent the output of the $j-th$ cross layer and $(j+1)-th$ cross layer, respectively, the other represent the same as in Equation. 6. All the variables were column vectors. Furthermore, the features of each layer were crossed and combined with the previous layer and original features and then added back to the previous layer. This is similar to the structure of a residual network in which the function f of each layer fits the residual of $x_{j+1} - x_j$. Thus, the gradient dispersion problem caused by the DNN could be solved using this residual network.

4.3 Deep Network Layer

The rectified linear unit (ReLU) [16] was used as the activation function in the proposed model due to its calculation simplicity. Moreover, the convergence speed of ReLU significantly outperformed that of other activation functions, such as sigmoid [17].

4.4 Output Layer

We calculated the output through a perceptron using a sigmoid activation function S, which is expressed as follows:

$$output = S(wx_{final} + b) \tag{8}$$

Where w and b represent the weight vector and bias parameter for the combination layer and x_{final} represent the final combination result of the output from both deep and cross network layer.

The cost function C is the log loss along with a regularization term:

$$C = \frac{1}{N}\sum_{i=1}^{N} y_i \log(y_{pred}) + (1 - y_i)\log(1 - y_{pred}) + \gamma \sum_l \|y_i - y_{pred}\|^2 \qquad (9)$$

Where y_{pred} represents the predicted value, y_i represents the true labels, N represents the total number of inputs, and γ represents the L2 regularization parameter.

4.5 Positional Features Preparation

To incorporate positional features into the model training process, an essential step involves feature engineering derived from the order of user interactions in log data. This method aims to capture the temporal aspects of user-item interactions, allowing the recommender system to discern the influence of position within the recommendation list. However, a key challenge arises during prediction. Users typically only engage with some items listed in a single day, necessitating a strategy to replace unavailable positional features with a constant value (1 to 10), as seen in Fig. 14.

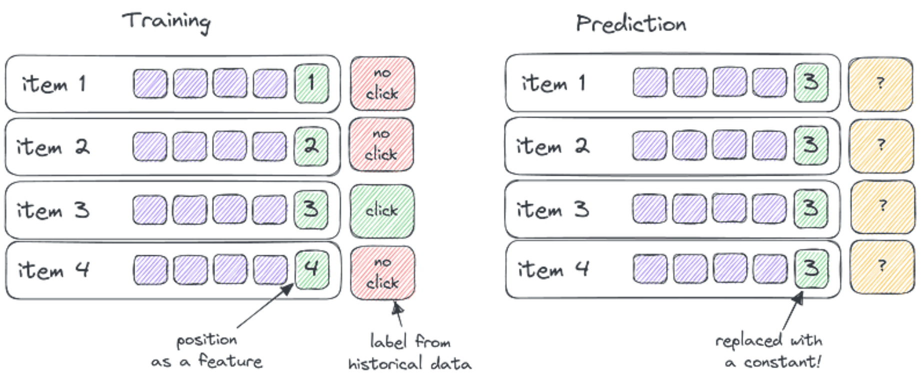


Figure 14 Diagram of applying positional features in this research

This application is crucial for nullifying the undue influence of absent position-related information during prediction, ensuring the model's robustness and generalizability in real-world scenarios where user engagement patterns may vary widely.

Chapter 5

Experimental Setup

This section describes our dataset, data preparation methods, experimental design, and evaluation criteria. We compare the performance of our approach to various mechanisms and baselines.

5.1 Dataset

The experiments were conducted using the Blockdit dataset. This dataset is provided by the Blockdit company. Blockdit is a well-known social media application in Thailand that presents content in a unique ``book-style'' format, where users can create stories or articles page by page, resembling a digital book or a magazine. Allowing them to express their ideas, share knowledge, or tell stories in a structured manner. The dataset contains user and article interactions and other attributes from the Blockdit application. The original dataset consists of 4 parts.

5.1.1 The characteristics of the user dataset.

The characteristics of the user dataset are detailed in Table 1.

Table 1 User data characteristics

No.	Column Name	Description
1	User.ts	The timestamp when the user data point is logged
2	User.id	User unique identifier number
3	User.status	The status of user

4	User.createdTime	The time the user creates their accounts
5	User.profile.gender	The gender of user
6	User.profile.birthTime	The user birthdate in unix time
7	User.profile.about	The description of user on their profiles
8	User.profile.interest	The interest user assigned when creating their accounts
9	User.profile.work	The occupation of user
10	User.profile.education	The education degree of user

5.1.2 The characteristics of the article dataset.

The characteristics of the article dataset are detailed in Table 2

Table 2 Article data characteristics

No.	Column Name	Description
1	Article.ts	The timestamp when the article data point is logged
2	Article.id	Article unique identifier number
3	Article.creator	Article's Creator unique identifier number
4	Article.page	Article's Page unique identifier number
5	Article.status	The status of article Ex. PUBLISHED, DELETED
6	Article.type	The type of article Ex. Read, Audio
7	Article.createdTime	The article created time to database
8	Article.publishedTime	The article published time to database
9	Article.attachment.video	Video unique identifier number in each

		article
10	Article.attachment.audio	Audio unique identifier number in each article
11	Article.ad	The status of the presence of ad in article
12	Article.categories	The categories of article
13	Article.original.article	The article's unique identifier (in case the article is a shared article from another source)
14	Article.original.page	The page's unique identifier (in case the article is a shared article from another source)
15	Article.origin.user	The Author unique identifier (in case the article is a shared article from another source)

5.1.3 The characteristics of the Interaction dataset.

The characteristics of the Interaction dataset are detailed in Table 3.

Table 3 Interaction data characteristics

No.	Column Name	Description
1	Interaction.ts	The timestamp when the interaction logged in unix time
2	Interaction.tsOrigin	The original timestamp when the interaction happened (from user device) in unix time
3	Interaction.user	User unique identifier number
4	Interaction.firUser	User temporary unique identifier number

		(when user is not registered)
5	Interaction.action	The type of interaction the user made Ex. Read25p = user read 25% of that article, IMPRESSION = User see the article **BUT NOT INTERACT** (Read, Like, Comment, etc.) to that article
6	Interaction.target.user	Author or user's target unique identifier number that user interacted with
7	Interaction.target.page	Page's unique identifier number that user interacted with
8	Interaction.target.article	Article's unique identifier number that user interacted with
9	Interaction.platform	The type of user device Ex. Web, IOS, Android
10	Interaction.language	The language the user selected/configured

5.1.4 The characteristics of the article's detail dataset.

The characteristics of the article's detail dataset are detailed in Table 4.

Table 4 Article's detail characteristics

No.	Column Name	Description
1	Article_detail.ts	The timestamp when the article's detail logged in unix time
2	Article_detail.id	Article's detail unique identifier number
3	Article_detail.blockContent	The content in the article
4	Article_detail.blockCount	The amount of block (paragraph-like) in the article

5.2 Data Preparation

First step, we conducted data cleansing by removing duplicate entries. We selectively filtered the data to retain only active and non-banned articles and users, adhering to the 'user status = registered' and 'article status = published' criteria. Furthermore, we excluded interaction data that lacked corresponding user IDs and article IDs, rendering it impossible to identify the associated interaction pairs. Additionally, we eliminated rows from the interaction data that lacked values in the 'article_blockContent' column, as article content is an indispensable feature for our study. For gender and date of birth fields with missing values, we assigned them with 'unknown'. To prepare the date dimension data for subsequent analysis, we converted it from Unix time format to datetime format.

Secondly, in the feature engineering phase for article data, we introduced a new feature called 'freshness', which represents the number of days that have passed since the article's publication at the time of the interaction. This feature captures the varying preferences of users, as some may favor recent news and only view articles updated within a certain timeframe, while others may be more interested in viral content, regardless of its publication date. Another newly created feature, 'user_1d', tracks the number of users who read the article on the previous day, reflecting its popularity and viral status.

To enrich the user data, we introduced a new feature called 'age range', which is derived from the 'date of birth' column and categorizes users into demographic groups such as 'kid', 'high school', and 'adult'. Additionally, we developed a feature named 'blockcount_category_ratio', which is based on users' past behavior and determines the proportion of articles a user reads from each category relative to their overall reading activity. Another feature, 'blockcount_category_mean', indicates the average length of articles the user tends to read in each category. The 'nunique_article' feature quantifies the number of distinct articles a user has read in the last 15 days, serving as a measure of user engagement. 'mean_blockcount' represents the average length of articles read by the user, while 'mean_freshness' calculates the average 'newness' of the articles they interact with.

Following the completion of data cleansing and feature engineering, the features listed in Tables 5 were employed in this experiment.

Table 5 Features engineering description

No.	Feature Name	Description	Type
1	Gender	The gender of user	Categorical
2	Age_ordinal	The age range of user	Categorical
3	Blockcount_category_ratio	Proportion of articles a user reads from each category relative to their reading activity	Numerical
4	Blockcount_category_mean	Average articles length the user read in each category	Numerical
5	Nunique_article	Unique articles a user has read in the last 15 days	Numerical
6	Mean_blockcount	Average article's length read by the user	Numerical
7	Mean_freshness	Average 'newness' of the articles they engage with based on article's published time	Numerical
8	Categories	The category of article	Categorical
9	Blockcount	The length of article	Numerical
10	Type	The type of article	Categorical

11	Freshness	Article’s published time – the time the user interacts to the article	Numerical
12	User_1d	Number of unique articles a user has read in the last 15 days	Numerical

The labels employed in this study reflect the nature of user interaction with articles. **A negative label** is assigned when a user merely views the article content without engaging with its entirety (**Interaction.action = “IMPRESSION”)**. Conversely, **a positive label** is assigned when a user actively interacts with the article by reading, sharing, or reacting to it in some way (Ex. **Interaction.action = “READ25P”).**

The collected data were split into three groups: training data (user-article interaction logs from February 12, 2023), validation data (user-article interaction logs from February 13, 2023), and testing data (user-article interaction logs from February 14, 2023). The traits of the training data are summarized in Table. 6.

Table 6 The data characteristic

Dataset	#Interaction	#User	#Article	#Positive	#Negative
Training	456,311	22,277	19,887	60,346	395,985
Validation	578,396	28,661	23,511	83,582	494,814
Testing	600,006	29,077	24,424	83,625	516,380

5.3 Exploratory Data Analysis (EDA)

Following data collection, an initial examination, exploration, and analysis of the dataset, called Exploratory Data Analysis (EDA), was conducted. This critical phase provided researchers with foundational insights into the dataset, which was crucial in validating its precision. To exemplify their exploratory analysis, researchers presented a graph in Fig. 15 to 19 below.

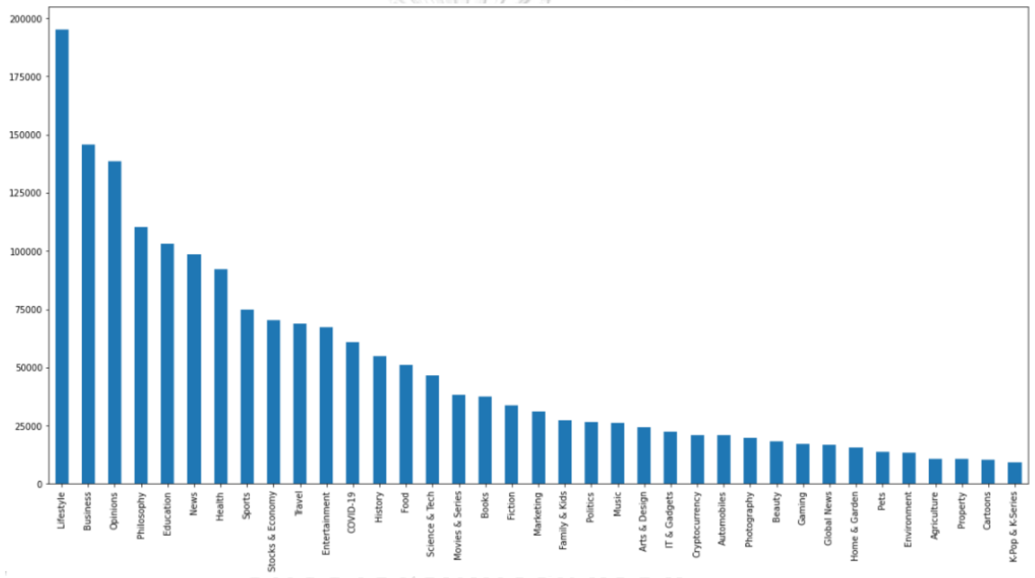


Figure 15 Article categories distribution

From Fig. 15, it can be observed that the top three most published article categories are Lifestyle, Business, and Opinions.

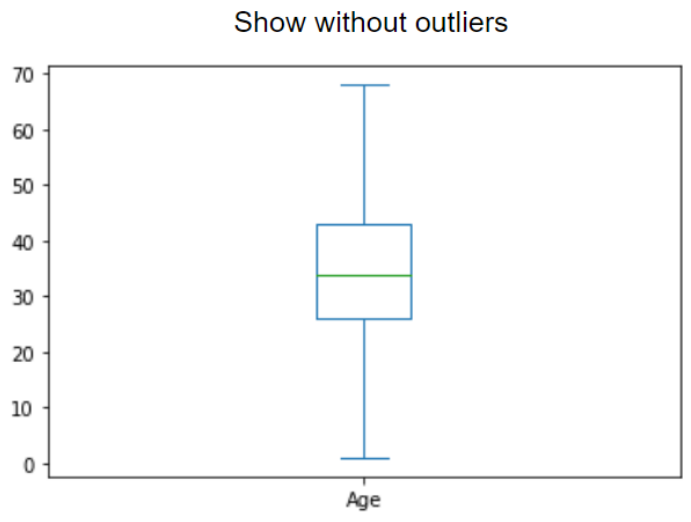


Figure 16 User's age distribution

From Fig. 16, it can be observed that the average age of users falls within the range of 25 to 45 years.

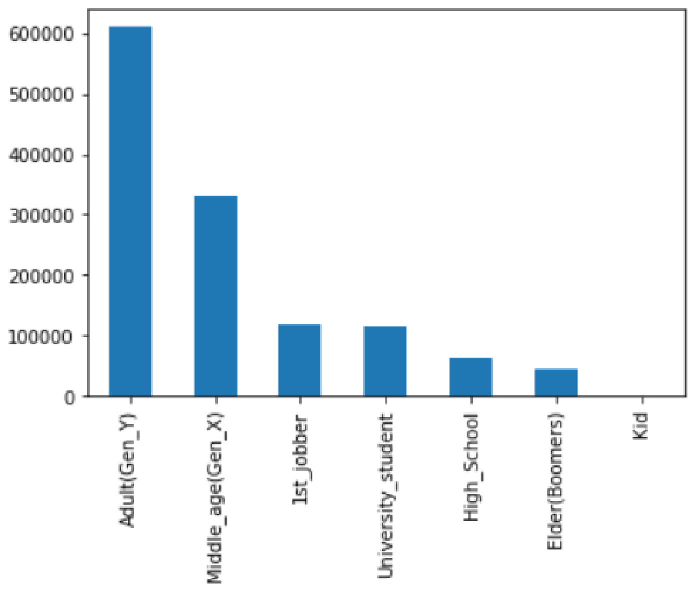


Figure 17 User's age distribution categorized by generation.

From Fig. 17, It can be observed that the majority of users belong to Generation Y and Generation X.

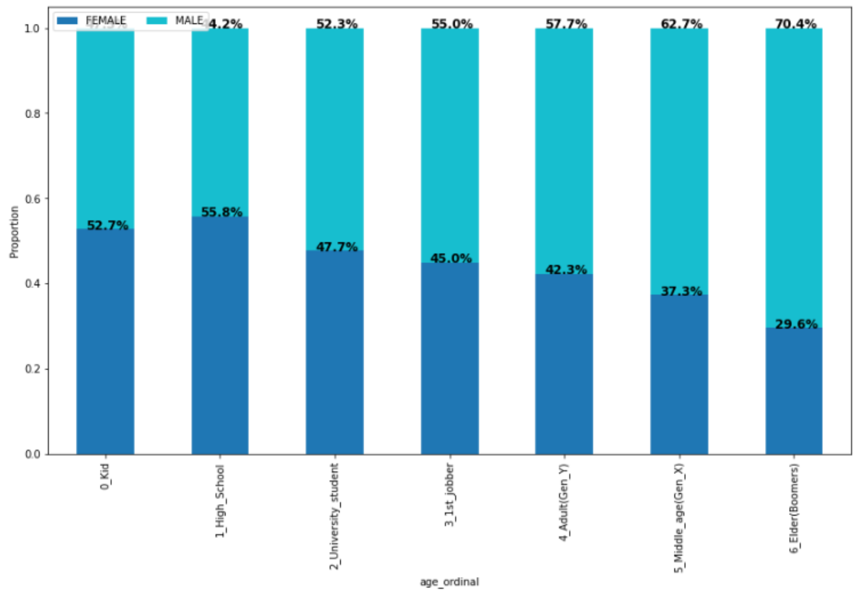


Figure 18 The distribution between age groups and the gender.

From Fig. 18, it can be observed that in each age group, the number of male and female users is relatively close, except for the elderly group where there is a higher proportion of male users.

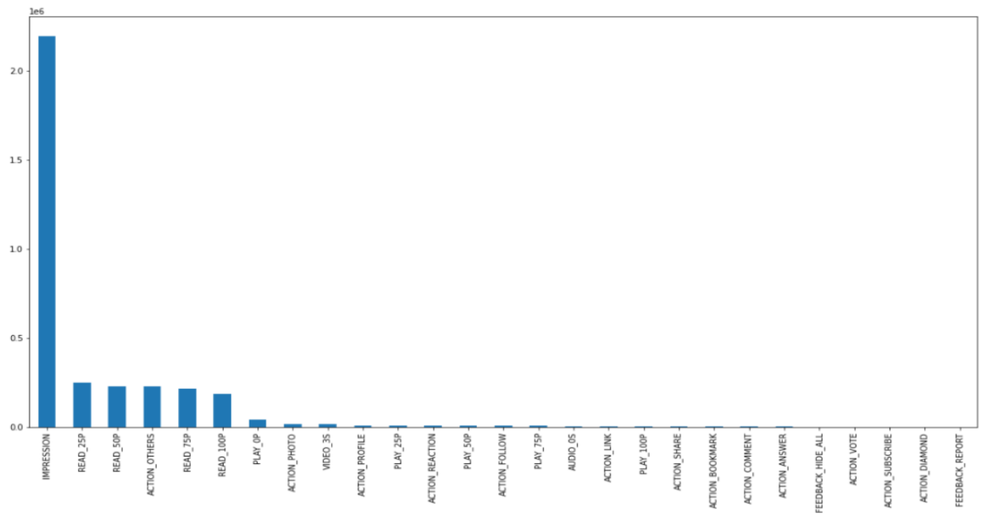


Figure 19 the distribution of user action types

From Fig. 19, it is evident that the majority of users exhibit a high frequency of viewing the articles (IMPRESSION). However, the number decreases significantly when it comes to reading more than 25% of the articles (READ_25P) and further decreases in subsequent actions levels.

5.4 Evaluation Metric

To evaluate performance, we used the testing data from February 14, 2023, and processed all articles for each user to obtain the score for all articles. We then ranked the top 100 articles and compared them to the actual data using the Mean Average Precision at K (MAP@k) as metrics to evaluate the offline performance of each different assigned position values. The MAP@K can be expressed as follows.

$$AP@K = \frac{1}{M}\sum_{k=1}^{K} P(k).rel(k) \tag{10}$$

$$MAP@K = \frac{1}{|U|}\sum_{K=1} U\,(AP@k)u \tag{11}$$

Where, $AP@K$ represents mean precision of k-rank object recommendation, $MAP@K$ represents mean average precision of k-rank object recommendation, M represents the number of times the object recommendation system matches the target user's preferences and U represents the total user numbers.

5.5 Experiment Settings

In training, **for sparse feature**, the positional feature will be treated as categorical features and will be added into DCN model using an embedding layer to convert the categorical positional data into vectors representation. **For dense feature**, the positional feature will be treated as numerical feature, normalized using min-max scaling to ensure it's on a similar scale to other dense features and then feed directly into deep side of DCN model. The implementation of positional features on both layers (Deep side and Cross side) is also conducted. The default parameters for this approach include three deep layers with 256, 128, and 64 neurons, as well as two cross layers.

As stated, we need to select a proper position value for prediction. But due to resource limitation, it's impossible to evaluate the model with all possible positions. Therefore, we conduct an offline experiment to select proper position value. Similar to previous work [18] [19], we apply different position values, ranging from position 1 to position 10.

5.5 Baseline Models

To assess the effectiveness of our proposed model, we compare its results with a baseline model, noted for its strong performance.

- LightGBM - a gradient boosting framework that uses tree-based learning algorithms and is designed for distributed and efficient training by employing a histogram-based algorithm. Making it capable of achieving higher precision with lower computational costs.
- XGBoost - an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable. It employs an ensemble of decision trees to produce accurate and robust predictive models.
- DeepFM [8] - combines the factorization machines (FM) and deep neural networks to capture both low-level feature interactions and high-level feature representations.
- DCN - an initial model with default parameters and without position debiasing capability.

5.6 Preliminary Experiment

In this preliminary experiment, we conducted our experiments within the Google Colab Pro environment, utilizing a T4 GPU with a system RAM limit of approximately 50 GB. Given the substantial size of our dataset, each day's data encompasses around 500k interactions; the constrained system RAM posed limitations on the number of days available for training. Consequently, we opted to employ the baseline XGBoost model to explore the influence of varying training data durations on model performance. The dataset was partitioned into two sets: "Training using 5 Feb," where the training data consisted of interactions from February 5, 2023, with validation on February 6, 2023, and testing on February 7, 2023; and "Training using 4-5 Feb," encompassing data from February 4-5, 2023, with the same validation and testing dates. The results are presented in Table 7.

Table 7 Comparison of preliminary results of in training data size in term of MAP@K

Model	MAP@25	MAP@50
Training using 5 Feb	2.97%	2.39%
Training using 4-5 Feb	3.31%	2.65%

The results presented in Table 7 reveal a marginal improvement in performance with an increase in the volume of data during the training period, though not reaching statistical significance. Given the constraints imposed by hardware limitations in this experiment, we opted to proceed with training using data from February 12, 2023, while validating the model on February 13, 2023, and testing its performance on February 14, 2023.

Chapter 6

Experimental Results

In this section, we present a two-part evaluation of our proposed approach. The first part compares our model to a baseline model, while the second part conducts a sensitivity analysis of position numbers.

6.1 Performance comparison between proposed model and baseline model

The comparison results are shown in Table. 2, with the best core highlighted in bold. This table compares the performance results of all baseline models and the proposed method using MAP@K. It is found that the proposed model, Position-aware DCN (Cross side), achieves the best performance among all values of k (25, 50, 75, 100) at 25.98 %, 23.11 %, 21.53 %, and 20.70 %, respectively with the default parameters of DCN.

Table 8 MAP@K for different models (position = 1)

Model	MAP@25	MAP@50	MAP@75	MAP@100
XGBoost	8.93%	7.85%	6.70%	5.46%
LightGBM	10.77%	9.47%	7.98%	7.16%
DeepFM	17.75%	16.98%	16.42%	15.36%
DCN	18.00%	17.05%	16.51%	15.79%
Position-aware XGBoost	9.51%	9.20%	8.83%	8.77%
Position-aware LightGBM	14.97%	13.02%	11.84%	11.01%
Position-aware DeepFM	18.5%	18.12%	17.83%	17.65%
Position-aware DCN (Deep side)	20.43%	20.18%	19.86%	19.52%
Position-aware DCN (Cross side)	**25.98%**	**23.11%**	**21.53%**	**20.70%**
Position-aware DCN (Both side)	19.49%	19.08%	17.65%	16.06%

6.2 Sensitivity to position number in prediction

We conducted the test to examine the effect of assigned position number during prediction. As shown in Fig. 20 to Fig. 23, we can see that MAP@K values vary as we assign different position values. In addition, the Cross-side model achieves the highest on MAP@K among all values of k (25, 50, 75, 100) at position constant of 8 at 26.30 %, 23.45 %, 21.78 %, and 21.02 %.

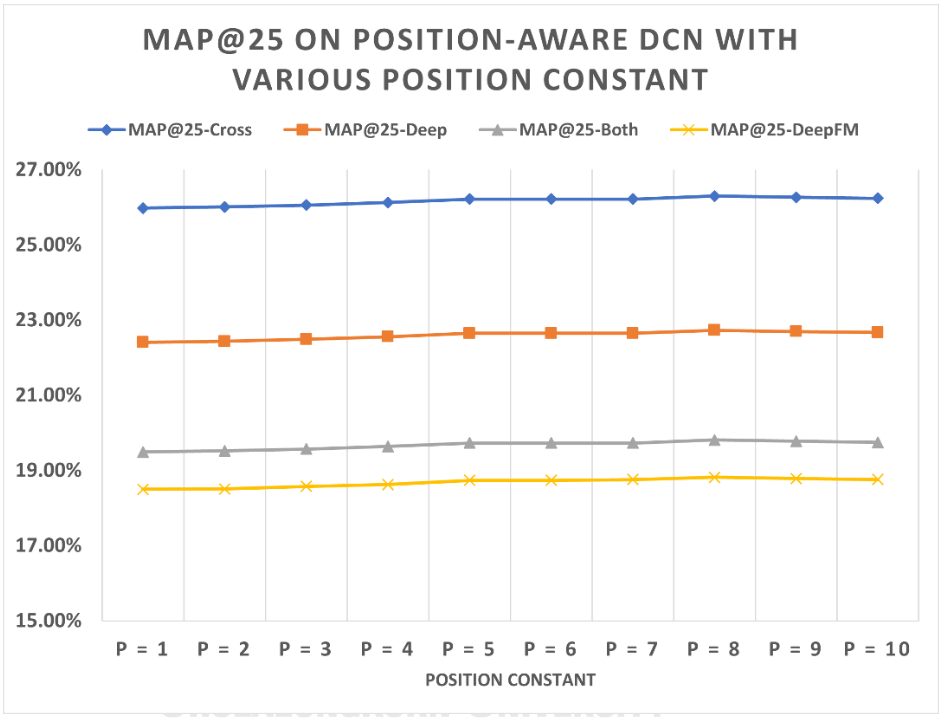


Figure 20 MAP@25 on Position-aware models with various position constant

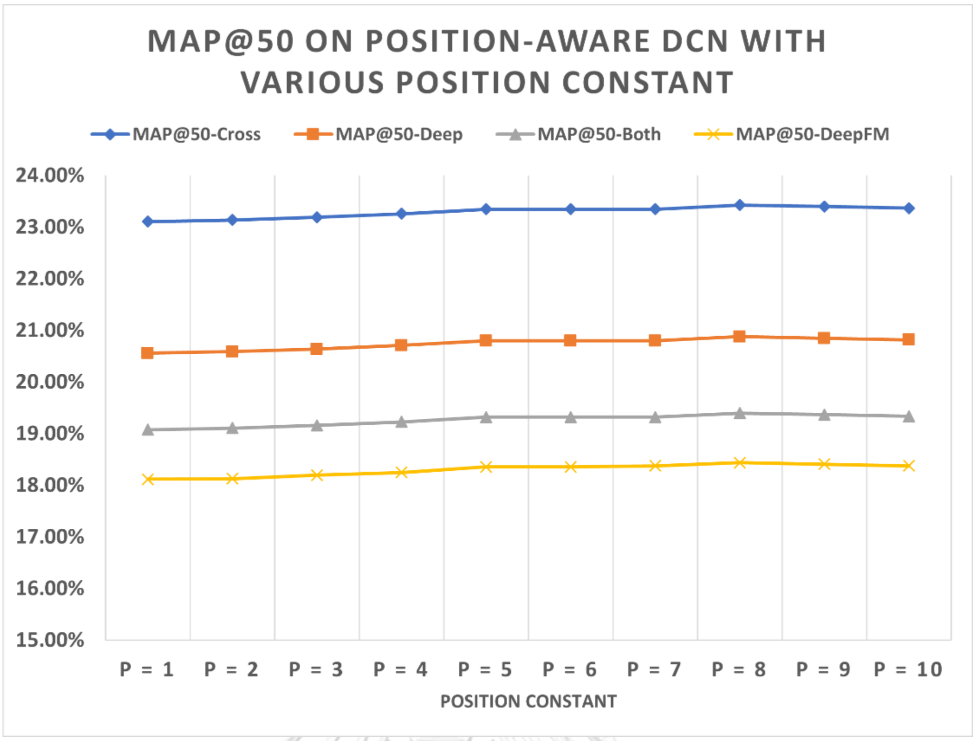


Figure 21 MAP@50 on Position-aware models with various position constant

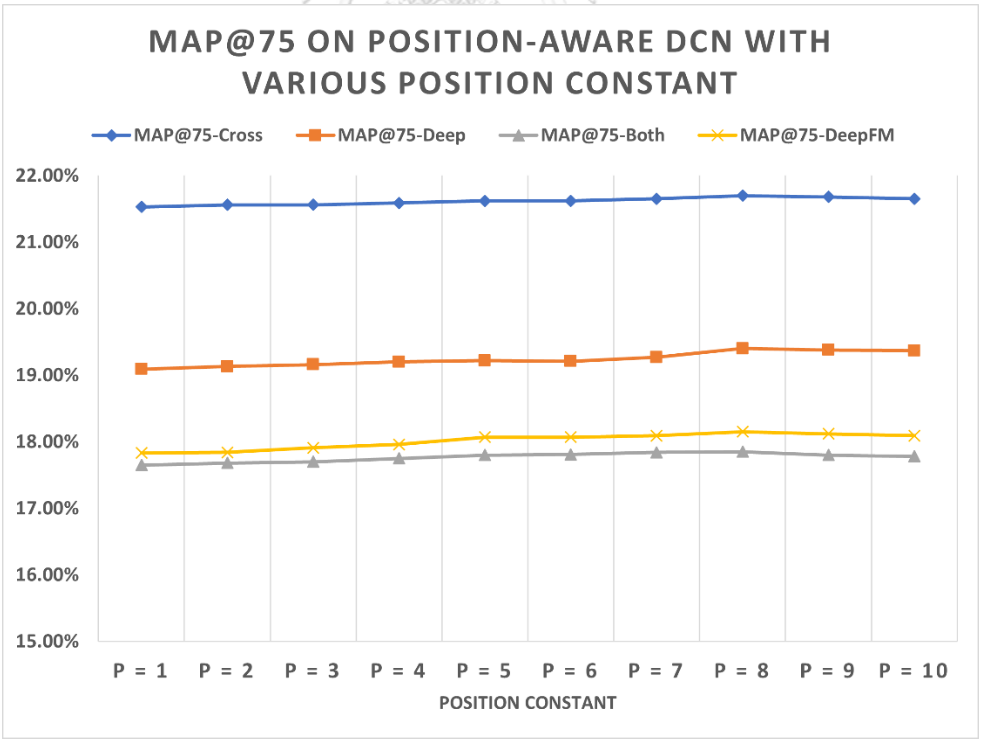


Figure 22 MAP@75 on Position-aware models with various position constant

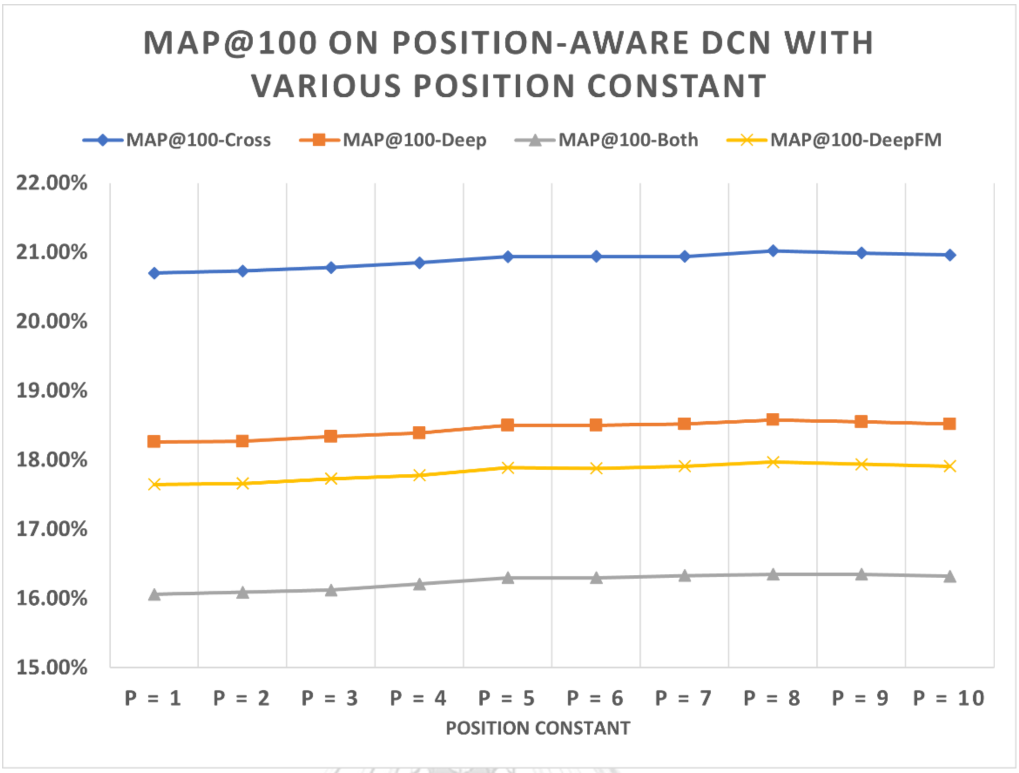


Figure 23 MAP@100 on Position-aware models with various position constant

The proposed model has outperformed the baseline model in predictive performance and click-through rate, particularly when applied as a dense feature. By treating positional features numerically (Cross-side), the model captures the inherent ordinality and potential non-linear relationships between position and user engagement, refining its predictive capabilities and understanding the implicit hierarchical significance within the data. This numerical approach enables the model to effectively comprehend and leverage the subtle nuances and gradients embedded within positional data, facilitating a more subtle prediction of user interactions and preferences. Also, the performance from different assigned position values on prediction in Fig. 20 to 23 indicates that the assigned position does not significantly impact the performance. Its primary purpose is to negate the position's impact on the prediction only.

On the other hand, positional features implemented as both a sparse and dense feature (Both sides) yield worse performance than those applied individually because

Deep & Cross Networks are designed to capture both low-level feature interactions (cross network) and high-level feature interactions (deep network) separately. Using the same feature in both representations increases the model's complexity unnecessarily. Thus, leads to a loss of information and performance degradation.

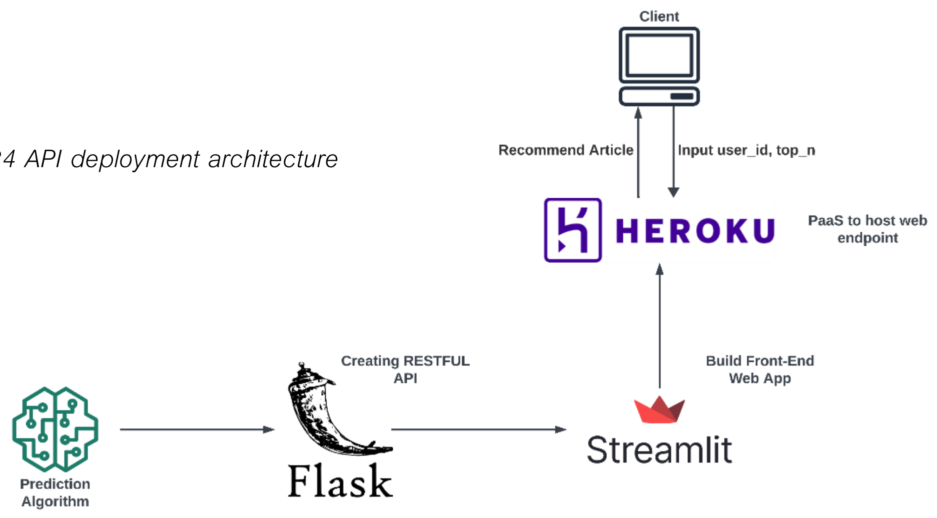
Chapter 7

Application Programming Interface Design and Developing

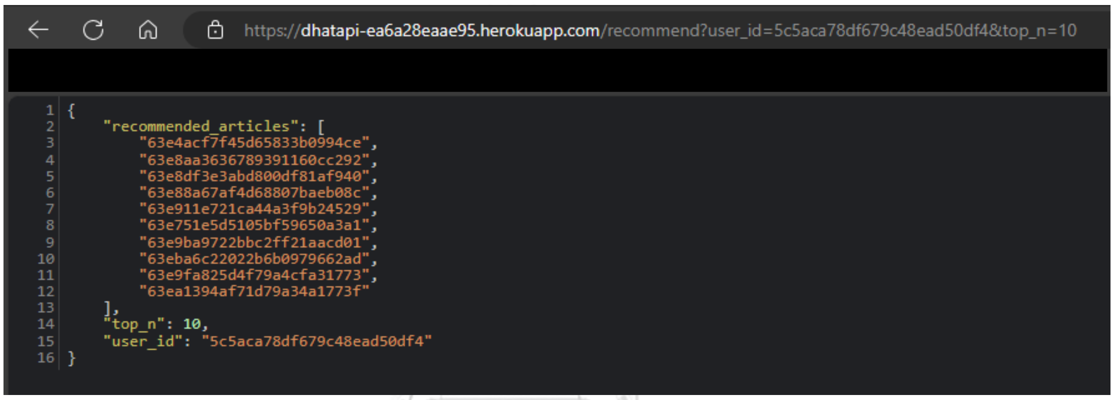
Deploying a recommender system into production involves various strategies, and considerations such as user, concurrent user, and load are crucial factors in this process. Firstly, the system must efficiently handle individual user interactions, ensuring a seamless experience for each user. Concurrent user handling is vital as the system must scale gracefully when multiple users simultaneously access recommendations. Additionally, addressing the overall load on the system, especially during peak usage periods, is essential for maintaining optimal performance.

In addressing these concerns, deploying the recommender system through an Application Programming Interface (API) provides a standardized and efficient solution. APIs enable seamless communication between different software components, ensuring scalability, ease of integration, and streamlined maintenance in a production environment. Furthermore, the author has utilized the API to create a front-end web interface, allowing users to interact with the recommender system easily. This additional step demonstrates the versatility of API deployment, culminating in a comprehensive demo that showcases the practicality and user-friendliness of deploying recommender systems through APIs. The deployment architecture is displayed in Fig. 24.

Figure 24 API deployment architecture



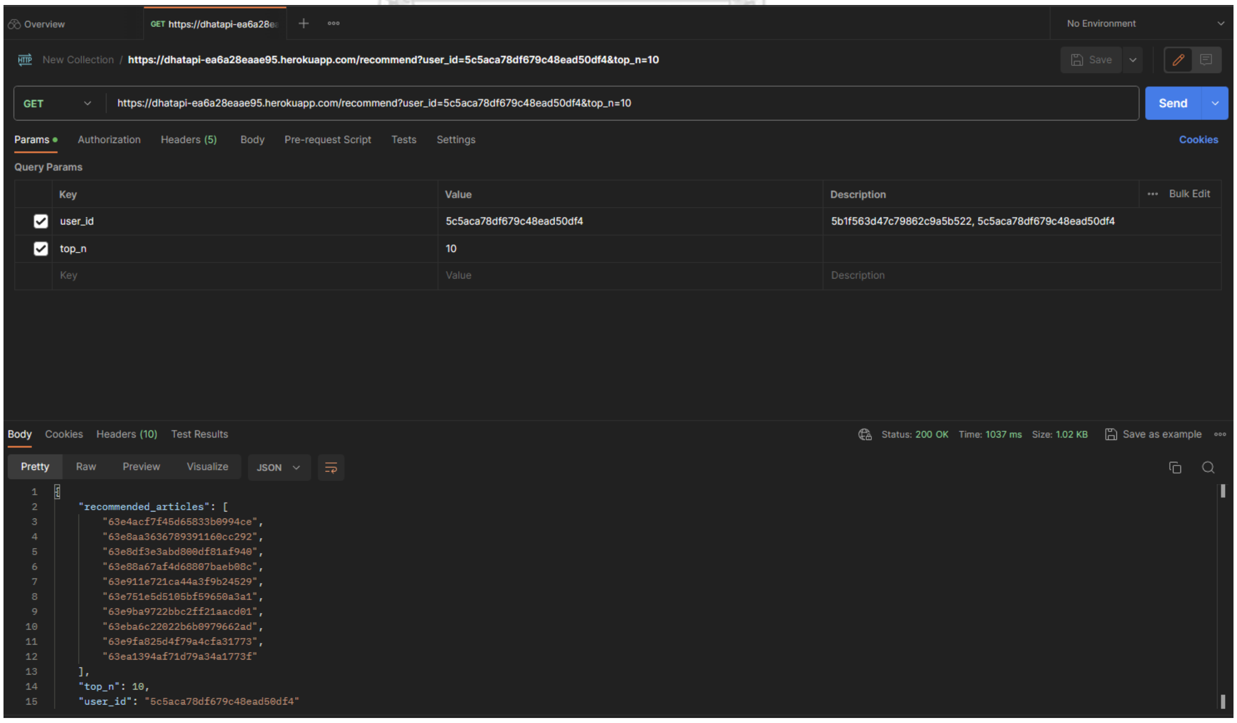
The recommender system deployment begins with the initiation of prediction algorithms accessed through a REST API implemented using Flask. Developers can interact with this API by providing parameters such as “user_id” and “top_n” to receive a list of recommended articles as output. This API serves as a versatile interface, allowing seamless integration into other projects demonstrating the flexibility and interoperability of the recommender system. The API Get request results are displayed in Fig. 25 and Fig. 26.



https://dhatapi-ea6a28eaae95.herokuapp.com/recommend?user_id=5c5aca78df679c48ead50df4&top_n=10

```
{
    "recommended_articles": [
        "63e4acf7f45d65833b0994ce",
        "63e8aa3636789391160cc292",
        "63e8df3e3abd800df81af940",
        "63e88a67af4d68807baeb08c",
        "63e911e721ca44a3f9b24529",
        "63e751e5d5105bf59650a3a1",
        "63e9ba9722bbc2ff21aacd01",
        "63eba6c22022b6b0979662ad",
        "63e9fa825d4f79a4cfa31773",
        "63ea1394af71d79a34a1773f"
    ],
    "top_n": 10,
    "user_id": "5c5aca78df679c48ead50df4"
}
```

Figure 25 API Calling with query string on web browser



GET https://dhatapi-ea6a28eaae95.herokuapp.com/recommend?user_id=5c5aca78df679c48ead50df4&top_n=10

	Key	Value	Description
✓	user_id	5c5aca78df679c48ead50df4	5b1f563d47c79862c9a5b522, 5c5aca78df679c48ead50df4
✓	top_n	10	
	Key	Value	Description

Status: 200 OK Time: 1037 ms Size: 1.02 KB

```
{
    "recommended_articles": [
        "63e4acf7f45d65833b0994ce",
        "63e8aa3636789391160cc292",
        "63e8df3e3abd800df81af940",
        "63e88a67af4d68807baeb08c",
        "63e911e721ca44a3f9b24529",
        "63e751e5d5105bf59650a3a1",
        "63e9ba9722bbc2ff21aacd01",
        "63eba6c22022b6b0979662ad",
        "63e9fa825d4f79a4cfa31773",
        "63ea1394af71d79a34a1773f"
    ],
    "top_n": 10,
    "user_id": "5c5aca78df679c48ead50df4"
```

Figure 26 API calling using Postman API

Users can conveniently access and visualize their recommended articles through a front-end web application implemented with Streamlit. This user-friendly interface enhances the user experience by providing an intuitive platform for exploring and interacting with the recommendations. The user's main interface is displayed in Fig. 27.

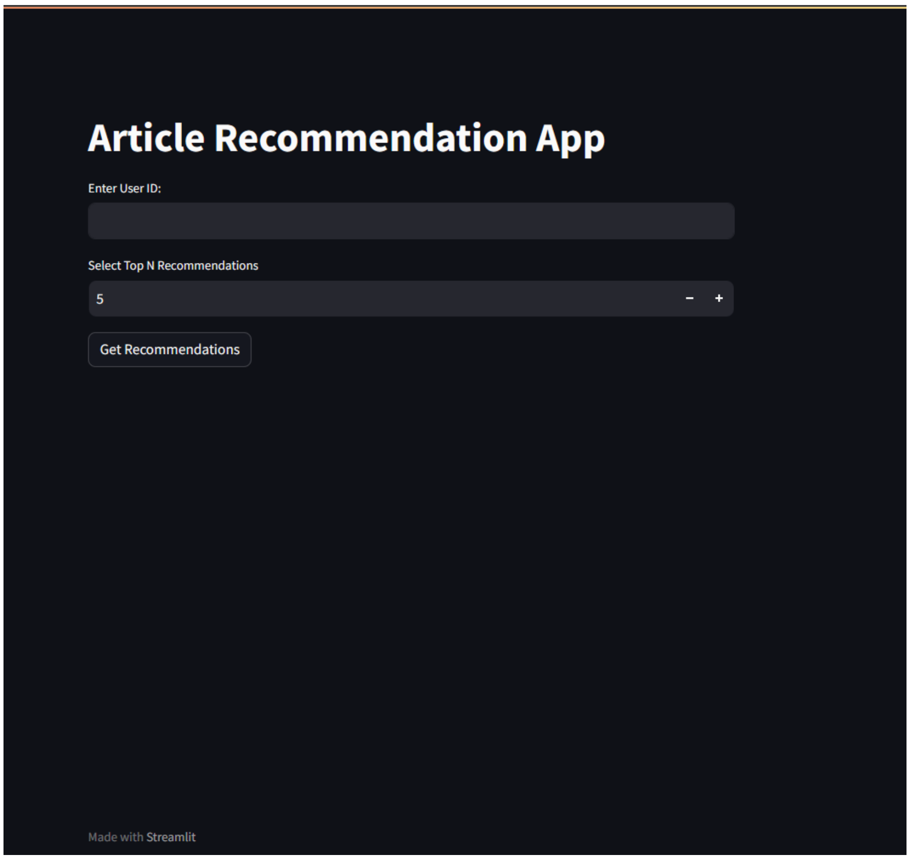
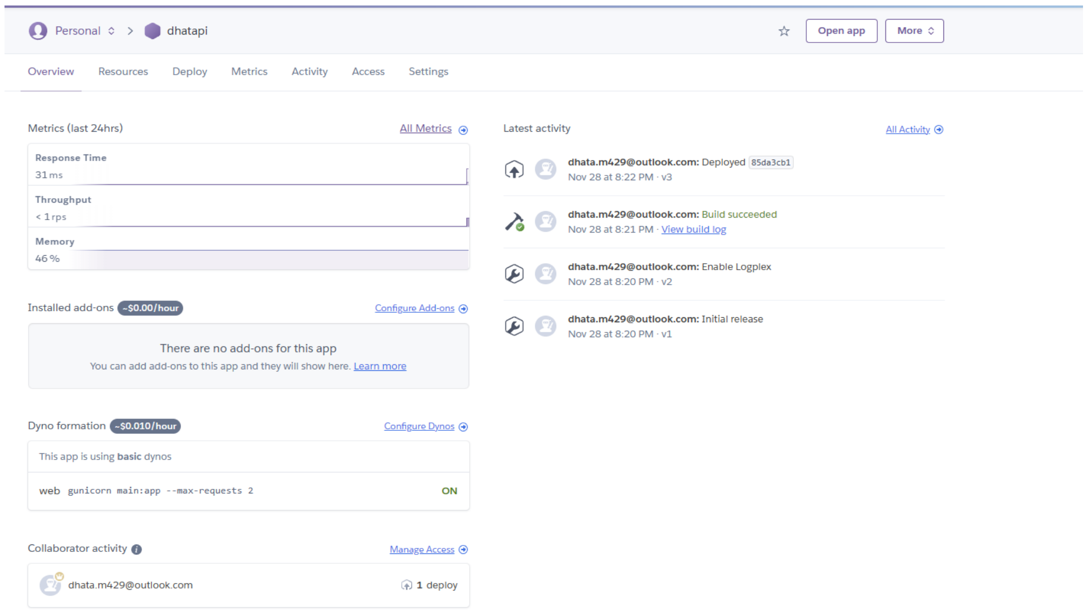


Figure 27 User's main interface

All these components are deployed on Heroku, a Platform as a Service (PaaS) provider. Heroku streamlines the deployment process, offering a scalable and efficient environment for hosting applications. By utilizing Heroku, the recommender system becomes easily accessible to both developers and end-users, ensuring a smooth and robust deployment experience. The dashboard of Heroku is displayed on Fig. 28.



Personal > dhatapi

Open app | More

Overview | Resources | Deploy | Metrics | Activity | Access | Settings

Metrics (last 24hrs) — All Metrics

Response Time: 31 ms

Throughput: < 1 rps

Memory: 46 %

Installed add-ons ~$0.00/hour — Configure Add-ons

There are no add-ons for this app

You can add add-ons to this app and they will show here. Learn more

Dyno formation ~$0.010/hour — Configure Dynos

This app is using **basic** dynos

```
web  gunicorn main:app --max-requests 2                ON
```

Collaborator activity — Manage Access

dhata.m429@outlook.com — 1 deploy

Latest activity — All Activity

dhata.m429@outlook.com: Deployed `85da3cb1`
Nov 28 at 8:22 PM · v3

dhata.m429@outlook.com: Build succeeded
Nov 28 at 8:21 PM · View build log

dhata.m429@outlook.com: Enable Logplex
Nov 28 at 8:20 PM · v2

dhata.m429@outlook.com: Initial release
Nov 28 at 8:20 PM · v1

Figure 28 Heroku Dashboard

7.1 Designing UML Diagram

In this step, the author designed UML diagrams to be used in the planning and readiness preparation for developing the API and web app of the recommender system. This involves simulating an analysis that details the structure of the system to be developed. Such modeling allows stakeholders to communicate and comprehend the system's components, ensuring a shared understanding in a standardized visual language known as the Standard Modeling Language. This research will focus on designing a standard deployment diagram to illustrate the system's deployment structure.

7.1.1 Deployment Diagram

Deployment diagram is designed to illustrate the physical architecture of a system in terms of its installation and operation. It depicts the layout of the system or components built on each node, showcasing the physical arrangement. This includes representing the relationships between various programs in the system. The diagram is crucial for planning the development process.

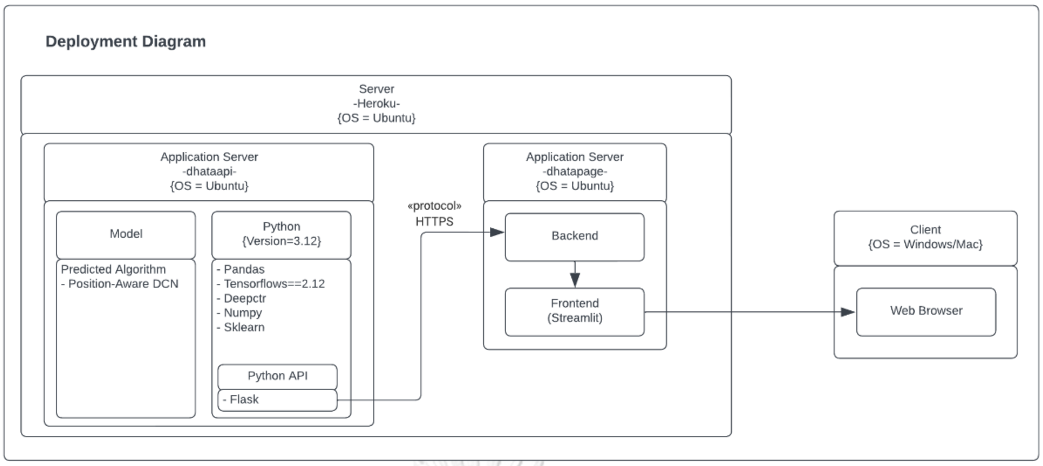


Figure 29 Deployment diagram of the proposed algorithm

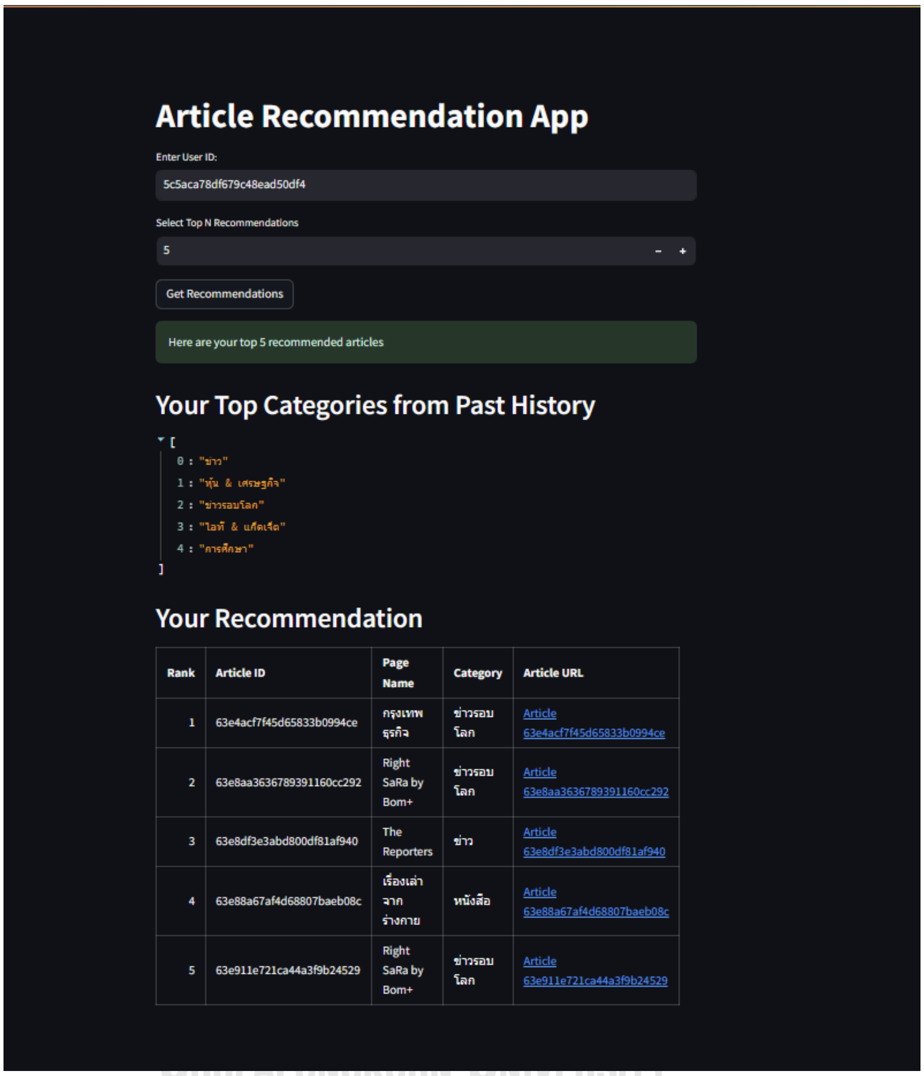
7.2 Designing Test Cases

Designing test cases is essential for deploying a recommender system, ensuring its functionality and reliability in real-world scenarios. The framework of test cases allows systematic evaluation of critical functionalities. In these test cases, the primary focus is on user interaction to assess the system's ability to cater to user needs accurately. The test cases are provided in Table 9.

Table 9 Test cases description

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Status
TC001	User input with valid user_id credentials (with default top_n)	1. Enter valid user_id, 2. Click “Get Recommendation”	User sees a list of recommended articles.	User presented with a list of recommended articles.	Complete
TC002	User input with invalid user_id credentials (with default top_n)	1. Enter invalid user_id, 2. Click “Get Recommendation”	User receives an error message.	User presented with an error message	Complete
TC003	User input with valid top_n credentials	1. Enter valid user_id and valid top_n. 2. Click “Get Recommendation”	User sees a list of recommended articles.	User presented with a list of recommended articles.	Complete
TC004	User input with invalid top_n credentials	1. Enter valid user_id but invalid top_n (top_N > 100). 2. Click “Get Recommendation”	User receives an error message.	User presented with an error message	Complete

The results of each test case are displayed in Fig. 30 to Fig. 33.



Article Recommendation App

Enter User ID: 5c5aca78df679c48ead50df4

Select Top N Recommendations: 5

Here are your top 5 recommended articles

Your Top Categories from Past History

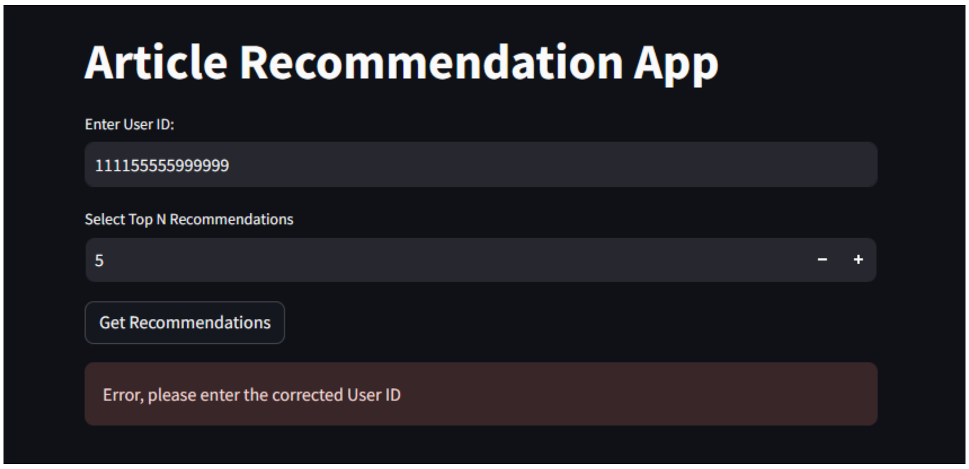
```
[
  0 : "ข่าว"
  1 : "หุ้น & เศรษฐกิจ"
  2 : "ข่าวรอบโลก"
  3 : "ไอที & แก็ดเจ็ต"
  4 : "การศึกษา"
]
```

Your Recommendation

Rank	Article ID	Page Name	Category	Article URL
1	63e4acf7f45d65833b0994ce	กรุงเทพธุรกิจ	ข่าวรอบโลก	Article 63e4acf7f45d65833b0994ce
2	63e8aa3636789391160cc292	Right SaRa by Bom+	ข่าวรอบโลก	Article 63e8aa3636789391160cc292
3	63e8df3e3abd800df81af940	The Reporters	ข่าว	Article 63e8df3e3abd800df81af940
4	63e88a67af4d68807baeb08c	เรื่องเล่าจากร่างกาย	หนังสือ	Article 63e88a67af4d68807baeb08c
5	63e911e721ca44a3f9b24529	Right SaRa by Bom+	ข่าวรอบโลก	Article 63e911e721ca44a3f9b24529

Figure 30 Test result of test case TC001

From Fig. 30, it can be observed that the recommended results are displayed correctly after user input with valid user ID and default top_N.


Article Recommendation App

Enter User ID:

111155555999999

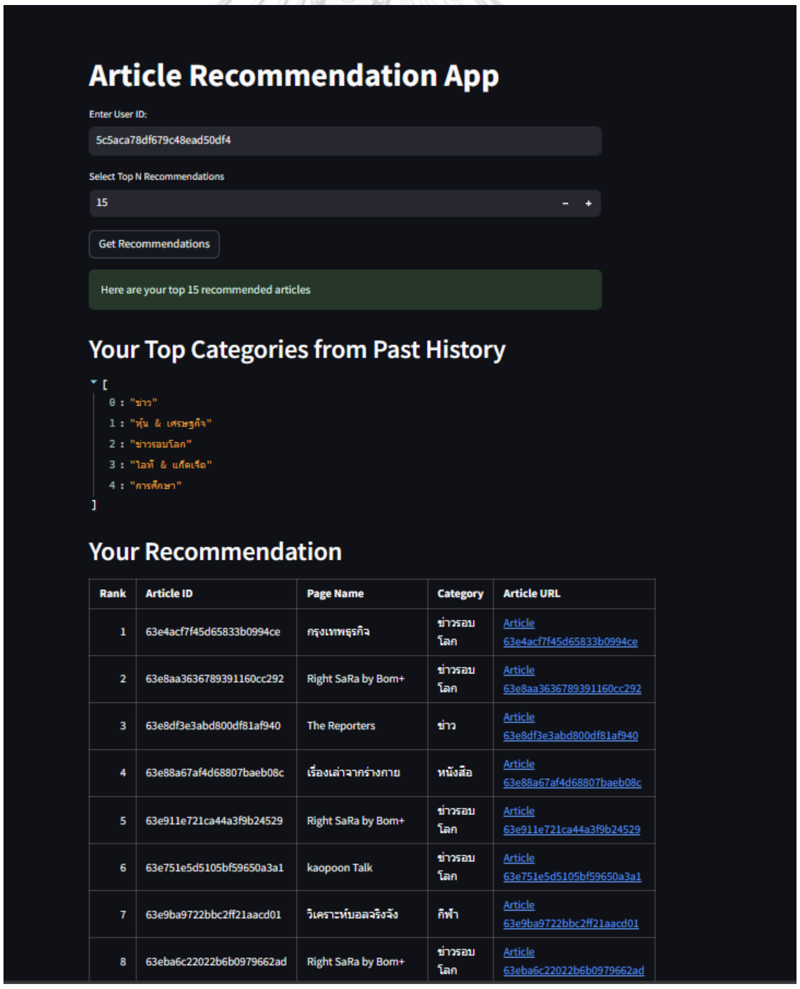
Select Top N Recommendations

5

Error, please enter the corrected User ID

Figure 31 Test result of test case TC002

From Fig. 31, it can be observed that the system shown error messages when the user ID is invalid.


Article Recommendation App

Enter User ID:

5c5aca78df679c48ead50df4

Select Top N Recommendations

15

Here are your top 15 recommended articles

Your Top Categories from Past History

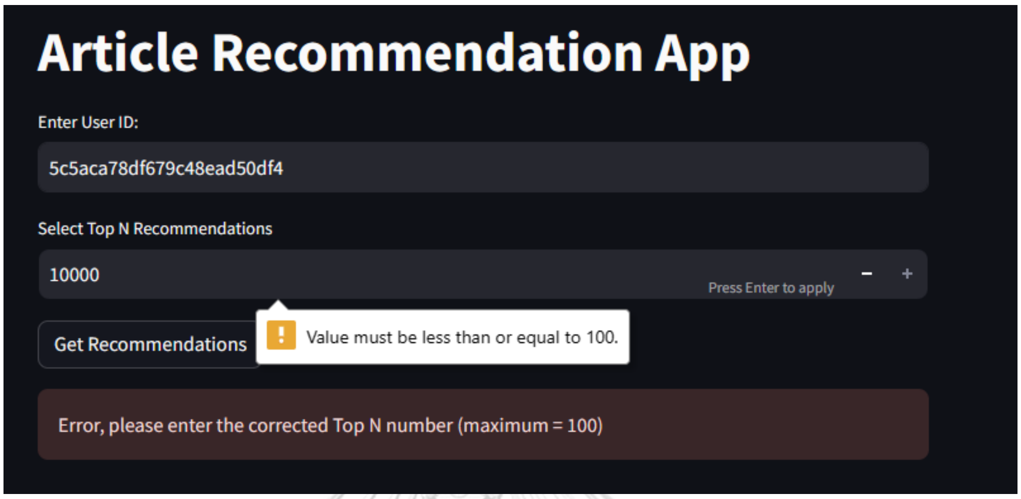
```
[
  0 : "ข่าว"
  1 : "หุ้น & เศรษฐกิจ"
  2 : "ข่าวรอบโลก"
  3 : "ไอที & แก็ดเจ็ต"
  4 : "การศึกษา"
]
```

Your Recommendation

Rank	Article ID	Page Name	Category	Article URL
1	63e4acf7f45d65833b0994ce	กรุงเทพธุรกิจ	ข่าวรอบโลก	Article 63e4acf7f45d65833b0994ce
2	63e8aa3636789391160cc292	Right SaRa by Bom+	ข่าวรอบโลก	Article 63e8aa3636789391160cc292
3	63e8df3e3abd800df81af940	The Reporters	ข่าว	Article 63e8df3e3abd800df81af940
4	63e88a67af4d68807baeb08c	เรื่องเล่าจากร่างกาย	หนังสือ	Article 63e88a67af4d68807baeb08c
5	63e911e721ca44a3f9b24529	Right SaRa by Bom+	ข่าวรอบโลก	Article 63e911e721ca44a3f9b24529
6	63e751e5d5105bf59650a3a1	kaopoon Talk	ข่าวรอบโลก	Article 63e751e5d5105bf59650a3a1
7	63e9ba9722bbc2ff21aacd01	วิเคราะห์บอลจริงจัง	กีฬา	Article 63e9ba9722bbc2ff21aacd01
8	63eba6c22022b6b0979662ad	Right SaRa by Bom+	ข่าวรอบโลก	Article 63eba6c22022b6b0979662ad

Figure 32 Test result of test case TC003

From Fig. 32, it can be observed that the recommended results are displayed correctly after user input with valid user ID and user desired top_N.



Article Recommendation App

Enter User ID:

5c5aca78df679c48ead50df4

Select Top N Recommendations

10000

Press Enter to apply

Value must be less than or equal to 100.

Error, please enter the corrected Top N number (maximum = 100)

Figure 33 Test result of test case TC004

From Fig. 33, it can be observed that the system shows error messages when the user ID is valid but out of range top_N number.

Chapter 8

Conclusion

In this research, we propose the positional debias application by using position value as a numerical feature toward the Cross side and a categorical feature toward the Deep side within the Deep & Cross Network, which we call "Position-aware DCN." Each reveals unique insights into user behavior and interaction patterns. This approach enhanced the model's predictive precision and showed a robust, straightforward framework to comprehend and mitigate positional bias. This research underscores the critical role of positional features in refining recommendation algorithms and establishes a foundation for future investigations into bias mitigation in algorithmic predictions.

Additionally, we extend our contributions by conducting an online deployment of the proposed model as an API. This deployment validates the practical applicability of our "Position-aware DCN" and demonstrates its simplicity and advantages through an overview and deployment diagram. By showcasing the system's functionality and ease of integration as an API, we aim to contribute to the broader understanding of deploying advanced recommender systems in real-world settings. This deployment methodology exemplifies the potential for widespread adoption, emphasizing the model's practicality and ease of implementation in diverse applications.

REFERENCES

1. Sage, D. *Social Media Users*. 2023 [cited 2023 5]; Available from: https://www.demandsage.com/social-media-users/.
2. Bobadilla, J.a.S., Francisco and Hernando, Antonio and others, *Collaborative filtering adapted to recommender systems of e-learning.* Knowledge-Based Systems, 2009. **22**: p. 261--265.
3. Leskovec, J.Y.a.Y.W.a.A.P.a.P.E.a.C.R.a.J., *Hierarchical Temporal Convolutional Networks for Dynamic Recommender Systems*. 2019. p. 2236-2246.
4. Koc, L., *Wide & Deep Learning for Recommender Systems.* CoRR, 2016. abs/1606.07792.
5. Wang, R.a.F., B. and Fu, G. and Wang, M., *Deep & Cross Network for Ad Click Predictions.* Proceedings of the ADKDD'17, 2017.
6. Richardson, M.a.D., Ewa and Ragno, Robert, *Predicting Clicks: Estimating the Click-through Rate for New Ads*. 2007: p. 521--530.
7. Chua, X.H.a.L.L.a.H.Z.a.L.N.a.X.H.a.T.-S., *Neural Collaborative Filtering.* 2017.
8. Guo, H., Tang, R., Ye, Y., Li, Z., He, X., & Mamitsuka, H., *DeepFM: A factorization-machine based neural network for CTR prediction.* 2017.
9. Huang, J.a.H., Ke and Tang, Qingtao and Chen, Mingjian and Qi, Yi and Cheng, Jia and Lei, Jun, *Deep Position-wise Interaction Network for CTR Prediction.* Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information, 2021.
10. Huang, J.a.T., Xingyuan and Wang, Zhe and Jia, Shaolin and Bai, Yin and Liu, Zhiwei and Cheng, Jia and Lei, Jun and Zhang, Yan, *Deep Presentation Bias Integrated Framework for CTR Prediction.* 2022.
11. Guo, H.a.Y., Jinkai and Liu, Qing and Tang, Ruiming and Zhang, Yuzhou, *PAL: A Position-Bias Aware Learning Framework for CTR Prediction in Live Recommender Systems.* Association for Computing Machinery, 2019: p. 452–456.
12. Yan, Z. *How to Measure and Mitigate Position Bias*. [cited 2022; Available from:

https://eugeneyan.com/writing/position-bias/.

13. Wang, X.a.G., Nadav and Bendersky, Michael and Metzler, Donald and Najork, Marc, *Position Bias Estimation for Unbiased Learning to Rank in Personal Search.* Association for Computing Machinery, 2018.

14. Aouichaoui, A.R.N., et al., *Comparison of Group-Contribution and Machine Learning-based Property Prediction Models with Uncertainty Quantification*, in *Computer Aided Chemical Engineering*, M. Türkay and R. Gani, Editors. 2021, Elsevier. p. 755-760.

15. Xin, S.a.L., Zhao and Zou, Pengcheng and Long, Cheng and Zhang, Jie and Bu, Jiajun and Zhou, Jingren, *ATNN: Adversarial Two-Tower Neural Network for New Item's Popularity Prediction in E-commerce.* 2021 IEEE 37th International Conference on Data Engineering (ICDE), 2021: p. 2499-2510.

16. A. Krizhevsky, I.S., and G. E. Hinton, *Imagenet classification with deep convolutional neural networks,.* Advancesin Neural Information Processing Systems. **25**.

17. H. Pratiwi, A.P.W., S. Susliansyah et al.,, *Sigmoid acti-vation function in selecting the best model of artificial neuralnetworks.* Journal of Physics: Conference Series. **1471 no. 1**.

18. Zhang, B.L.a.R.T.a.Y.C.a.J.Y.a.H.G.a.Y., *Feature Generation by Convolutional Neural Network for Click-Through Rate Prediction.* The World Wide Web Conference, 2019.

19. Gai, G.Z.a.C.S.a.X.Z.a.Y.F.a.H.Z.a.X.M.a.Y.Y.a.J.J.a.H.L.a.K., *Deep Interest Network for Click-Through Rate Prediction.* 2018.

VITA

NAME	Dhata Muangrux
DATE OF BIRTH	26 March 1998
PLACE OF BIRTH	Bangkok, Thailand
INSTITUTIONS ATTENDED	Chulalongkorn University
HOME ADDRESS	134/239 soi 1/15 Burasiri Sanambinnam village, Sanambinnam road, Tha Sai, Nonthaburi 11000