Automatic Personalized Health Insurance Recommendation Based on Utility and User Feedback (Supplementary Material)

Hao Liu
The Hong Kong University of Science and Technology
hliubs@cse.ust.hk

Raymond Chi-Wing Wong
The Hong Kong University of Science and Technology
raywong@cse.ust.hk

***Abstract*—Health insurance is a fundamental type of insurance that is essential to every person. Due to the increasing requests for personalized health insurance recommendation, traditional ways of obtaining recommended health insurance products from experts or agents are insufficient, and thus automatic personalized health insurance recommendation is needed. However, existing approaches to addressing this problem either need heavy input from users to obtain the utility of health insurance products for recommendation, or directly rely on user feedback and neglect the utility in recommendation. In this paper, we propose an automatic personalized health insurance recommendation system that incorporates both utility-based recommendation and user feedback based recommendation. To achieve that, we propose a two-component architecture in our system, where the first component is for daily operations to provide utility-based personalized recommendation for users with simple input of health risks, and the second component is to refine the first component by learning dynamic user preferences from user feedback data and combine the learnt preferences into utility-based recommendation. On a complex real-world insurance dataset, our proposed system achieves 2x more effectiveness than the baseline approaches for improving the recommendation quality.**

***Index Terms*—Recommendation, Insurance**

I. Introduction

Health insurance is a fundamental type of insurance and accounts for a significant marketing share. In Hong Kong, the marketing share of health insurance is 34.4% in 2021, which is the highest among all types of insurance [1]. Nearly every person has the basic need of a health insurance to cover essential health treatment and prevent unexpected high medical cost. Due to the massive number of health insurance products in the market across different providers, it is difficult to select a best suitable product considering one's personalized needs. There is thus an increasing need for personalized health insurance recommendation. However, traditionally, to give recommendation to a customer, rich knowledge on health insurance terms is required for evaluating the health insurance products in the market with respect to the customer's basic information and health conditions. Therefore, customers need to manually communicate with insurance experts or agents for health insurance recommendation, which could be time-consuming and biased.

Recently, some automatic personalized health insurance recommendation systems were studied [2]–[4]. They could be classified into *utility-based recommendation* and *user feedback based recommendation*. The *utility-based recommendation systems*, followed by most of the existing studies [2], [4], are mainly based on how good service the insurance product offers to users, where products are represented by some numerical attributes (e.g., out-patient coverage and dental coverage). To capture users' preferences, these systems require users to specify which attributes they care the most and then decide the weights of the attributes to compute the utility of the insurance products. However, these systems need heavy effort from users to specify their preferences. On the other hand, the *user feedback based recommendation systems* exploit users' feedback revealed in users' browsing history (e.g., users' click sessions on insurance web pages) [3]. The problem is thus modelled as how to learn the user's preferences from user sessions. However, these systems solely consider user's feedback without using the insurance product's information itself, which could lead to a case where a recommended insurance product has lower utility than one that is not recommended. Moreover, none of the above systems explains why the recommendations are made, but users need to understand the clear reason before deciding to purchase an insurance product.

In this work, we propose an automatic personalized health insurance recommendation system that incorporates both utility-based recommendation and user feedback based recommendation. We design a novel architecture in our system consisting of two components. The first component is used in daily operations by users to obtain the utility-based automatic recommendation based on their personalized needs but without heavy input from users. The second component is used as a "refiner" to refine the first component such that the user feedback information is combined into our recommendation system for consideration, and the utility of the recommended insurance products is preserved.

Our automatic personalized health insurance recommendation system has the following challenges. The first challenge is how to process and model the complex and unstructured health insurance data. In a real project, we need to process the information of over 150 health insurance products with complicated terms and constraints. The second challenge is how to effectively capture users' personalized needs based on their health conditions given a variety of complicated health insurance products. The third challenge is how to combine the

user feedback information into utility-based recommendation such that users' dynamic preferences reflected in the user feedback can be well considered and the utility of recommended products can be preserved.

To address these challenges, in our system, we first include a data processing approach that processes all the health insurance terms into a list of *scoring attributes*. In this way, each health insurance product can be modeled by *structured data* and can be easily applied in a utility-based recommendation model. We also model the health-related needs of a user for health insurance with concrete *health risks* (e.g., lung cancer risk level) and thus instead of manually specifying his/her own preferences towards many attributes, the user only needs to indicate his/her health risks to obtain the recommended health insurance products.

In the first component of our system, we leverage a linear scoring model to achieve utility-based recommendation, where the utility (i.e., score) of each product is computed based on the scoring attributes and a set of weights. To enable our recommendation system, the initial weights in the linear scoring model are set up by domain experts based on the relevance between scoring attributes and health risks. In this way, users' personalized health-related needs are well encoded into our recommendation system. Also, the linear scoring model is a highly explainable model for users to have a clear comprehension of the rationale behind the recommendation.

The second component of our system handles the task of updating the weights of the linear scoring model based on the user feedback data automatically collected from the user interface over a period of time. With the user feedback data (i.e., users' click sessions) in this period, we can learn the dynamic user preferences by adapting a state-of-the-art effective click model called *GraphCM* [5]. Then, we propose a *rewarding score* approach to combine the dynamic user preferences in this period with the current linear scoring model such that a new score is obtained for each product. Finally, the weights are updated with a linear regression of the new scores, and with the updated weights, the products that receive some dynamic user preferences will be likely to obtain a higher utility than before. By updating the weights in our recommendation system only, we still retain a utility-based recommendation, while the dynamic user preferences from user feedback data are also encoded into the system and taken into consideration for recommendation.

Our contributions are summarized as follows.

- We present an approach to process the complicated health insurance data and model users' personalized health needs, and we use this approach to handle a complex dataset.
- We propose an automatic personalized health insurance recommendation system with a two-component architecture that combines utility-based recommendation with user feedback based recommendation.
- We include a linear scoring model to handle the utility-based recommendation that well encodes users' personalized health need and offers explainable results.
- We learn the dynamic user preferences from user feedback data by adapting a state-of-the-art effective click model and we propose a rewarding score approach to handle the weight update task such that both utility-based recommendation and user feedback based recommendation are achieved.
- We evaluate our automatic personalized health insurance recommendation system to verify its effectiveness. Our system is verified to be 2x more effective than the baseline approaches in improving the recommendation quality.

In the following, we first show how we process and model our insurance data and users' personalized health needs, and also formalize our recommendation problem in Section II. Next, we show the overview of our system in Section III, and then present the details of the two components of our system in Section IV and Section V, respectively. We then review the related work in Section VI. The experimental evaluation of our system is shown in Section VII. We discuss some potential issues in Section VIII and conclude our work in Section IX.

II. Data Processing and Modeling

In this section, we first show how we collect and process the raw data of health insurance in Section II-A, and then we discuss how we handle users' personalized health conditions in Section II-B. Finally, we formalize our processed dataset and problem definition of personalized health insurance recommendation in Section II-C.

A. Health Insurance Data

There exist numerous health insurance products across various providers in the market. In a collaboration project with a local company, we collected the raw information of over 150 health insurance products across more than 10 major health insurance providers. For each health insurance product, we focus on its *insurance terms*, which are the services provided for the policyholder wrt claiming and other possible aspects. An example of insurance terms is *payout for heart disease is 150%*. Clearly, the products offering possibly better services to a user are more worth recommending.

Since the insurance terms are described in *unstructured* text, we transform them into *structured* data that can be applied in a multi-criteria recommendation scheme. Specifically, for each possible aspect of insurance terms (e.g., *payout for heart disease*), we create a corresponding scoring attribute to measure how good an insurance product is wrt this aspect. To achieve that, we first extract the raw value for this aspect (e.g., 150%), and then decide a score in $[0, 1]$ according to the raw value (by a piecewise function, e.g., score is 1 if *the payout for heart disease* $\geq 150\%$ and 0 otherwise), where a larger score is more preferred.

As such, each health insurance product is represented by a list of numerical scoring attributes, each with value in $[0, 1]$. In our dataset, the total number of scoring attributes is around 200 covering all aspects of health insurance terms we collected.

TABLE I: Example of Health Insurance Dataset

Product	t_1 (Payout for Cancer)	t_2 (Payout for Heart Disease)
p_1	0.8	0.5
p_2	0.6	0.5
p_3	0.2	1

B. Personalized Health Conditions

The goal of our system is to give users personalized recommendation of health insurance products based on their own health conditions. Considering a large number of scoring attributes of each product, it remains a challenge how to achieve personalized recommendation.

According to previous studies in health insurance, *health risk* of certain diseases is an important factor for users to consider when buying a health insurance product [6], [7]. Therefore, we model the personalized health needs of different users by their health risks concerning some diseases. In particular, a user could have one or multiple health risks or no risk at all. In our system, we collect a list of 44 common health risks, each related to a specific disease. We also create one special risk called "no risk". Since users are generally aware of what types of health risks in terms of explicit diseases they are faced with, they just need to select one or multiple health risks from the list (or "no risk") in our system to obtain the personalized health insurance recommendation.

Based on general laws and regulations in the world, users' health conditions are highly sensitive data. Thus, we do not "bind" a user with his/her health conditions. That is, our system will never keep any information that links a user with specific health risks. Instead, the health risks are provided by the user each time a recommendation is requested, which could avoid the potential privacy issue.

Besides, there could be correlations among different health risks. For example, the risk of diabetes might possibly be correlated with the risk of heart diseases. The correlations could be revealed in two ways. One is the association (i.e., the patterns of different risks) observed from users, and the other is the intrinsic relevancy of two diseases based on domain knowledge. In Section V-B, we describe how the two types of correlations are handled in our system.

C. Processed Data and Recommendation Task

After processing the health insurance data and modeling users' personalized health conditions with health risks, we formalize the dataset and recommendation task as follows.

Our health insurance dataset P consists of n health insurance products: $P = \{p_1, p_2, ..., p_n\}$. We have a list of m scoring attributes, denoted by $t_1, t_2, ..., t_m$, each corresponding to a health insurance term. For the i-th product p_i in P ($i \in [1, n]$), p_i has a value in $[0, 1]$ for each scoring attribute t_j ($j \in [1, m]$), denoted by $v_{i,j}$. We show a dataset in Table I with 3 health insurance products and 2 scoring attributes as an example, where the product p_1 scores 0.8 for attribute *payout for cancer* and scores 0.5 for attribute *payout for heart disease*.

We are also given a set $\mathcal{R}$ of all possible health risks: $\mathcal{R} = \{r_1, r_2, ..., r_{|\mathcal{R}|}\}$, each representing a health risk of a specific disease. In addition to the risks in $\mathcal{R}$, we also have a special risk, denoted by r_0, to represent "no risk". The header row of Table II illustrates an example of 3 risks, namely, *cancer risk*, *heart disease risk* and *no risk*. We represent the health condition of a user by a non-empty set R of risks, where $R \subseteq \mathcal{R}$ or $R = \{r_0\}$. Given a user with health condition (or risk set) R, we would like to give the user a ranked list of all feasible products in P (i.e., a recommendation list) concerning the health condition R of the user, where a product ranks higher in the list if it is more strongly recommended.

Moreover, we form a $|\mathcal{R}| \times |\mathcal{R}|$ matrix $\mathcal{I}$ based on domain knowledge to represent the intrinsic relevancy between each pair of health risks, where $\mathcal{I}_{i,j}$ for $1 \leq i, j \leq \mathcal{R}$ is a real number in $[0, 1]$ representing how strong the relevancy between risk r_i and r_j is (the stronger relevancy, the larger $\mathcal{I}_{i,j}$).

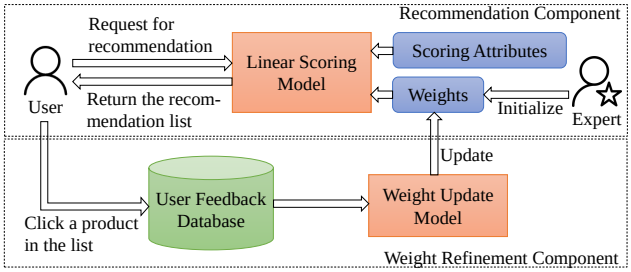

Fig. 1: System Overview

III. System Overview

In this section, we first present an overview of our automatic personalized health insurance recommendation system. As illustrated in Figure 1, our system consists of two components, namely, the *recommendation component* and the *weight refinement component*.

The recommendation component handles automatic personalized recommendation with a linear scoring model which can compute the score of each product based on the scoring attributes and the weights of the model. Initially, the weights of the linear scoring model are set by domain experts (or based on the domain knowledge obtained from Large Language Models when no domain experts are available, as we discuss in Section VIII-A). When a user requests for recommendation with his/her risk set, our system returns a ranked list of products to the user for recommendation.

The weight refinement component addresses the problem of weight updating such that the utility-based recommendation and user feedback based recommendation are combined in our system. When a user clicks a product in the recommendation list, the system will record this click behavior instantly and store this click information in a user feedback database. Then, after a period of time (e.g., a month), we take all the user click information in this period from the database and then use the weight update model to compute the updated weights. In this way, the user feedback information in this period will be combined into the system to contribute to the most up-to-date weights in the linear scoring model, resulting in the updated recommendation. Clearly, although the core of the

TABLE II: Example of Weights in Linear Scoring Model

Scoring Attributes	r_0 (No Risk)	r_1 (Cancer Risk)	r_2 (Heart Disease Risk)
t_1	0.5	0.7	0.4
t_2	0.5	0.3	0.6

updated recommendation is still utility-based, it has already taken the user feedback information into consideration.

In the next two sections, we present the details of the two components, the recommendation component (in Section IV) and the weight refinement component (in Section V)

IV. Recommendation Component

The most important desideratum in our recommendation component is that the recommendation results must be transparent to the user, which is strictly required in the insurance industry [8], [9]. We thus apply a *linear scoring model*, which is an explainable and effective approach for multi-criteria decision-making and utility-based recommendation [10], [11].

In the linear scoring model, the final ranked list for recommendation will be based on the descending order of the scores of all feasible products, where the score of each product is computed to be the weighted sum of all scoring attributes. The weight for each scoring attribute in this model represents the importance of this attribute. By demonstrating the score of each attribute, the corresponding weight and how the total score is computed (in a well-designed user interface described in our supplementary material [12]), the user could understand clearly the rationale behind the recommendation results.

In this work, to achieve personalized recommendation according to users' health conditions (in the form of health risks), different importance could be given to the scoring attributes under different health conditions. Thus, the weights in the linear scoring model differ for varied risks. Based on this intuition, our linear scoring model is formalized as follows.

Consider a risk r_k for $k \in [0, |\mathcal{R}|]$. Note that r_k could be "no risk" (when $k = 0$). We first define the score of a product p_i wrt r_k, denoted by $s(p_i, r_k)$, as follows.

$$s(p_i, r_k) = \sum_{j=1}^{m} w_{j,k} v_{i,j} \tag{1}$$

where $w_{j,k}$ for $j \in [1, m]$ and $k \in [0, |\mathcal{R}|]$ denotes the weight of scoring attribute t_j wrt risk r_k. Clearly, each risk r_k is related to a set of m weights (i.e., $\{w_{j,k} | j \in [1, m]\}$), each corresponding to the importance of a scoring attribute wrt this risk. Let W denote the set of all weights involved in the above formulation, i.e., $W = \{w_{j,k} | j \in [1, m], k \in [0, |\mathcal{R}|]\}$. Clearly, $|W| = m \cdot (|\mathcal{R}| + 1)$. Table II shows 6 weights in our running example, where the weights of t_1 and t_2 wrt risk r_1 are 0.7 and 0.3, respectively.

Next, we formalize the score of a product for a user. Consider a user with risk set R. If the user only has one risk (or "no risk"), then using $s(p_i, r_k)$ in Equation 1 to compute scores is sufficient. However, if the user has multiple risks (i.e., $|R| > 1$), one straightforward way is to maintain different weights for each possible risk set, which will lead to a considerably large set of weights in our model. Instead, we use another approach of taking the *average* score of a product p_i wrt a risk in R (among the scores wrt all risks in R). Then, the score of a product for a user can be computed based on the same weight set W. Specifically, we define the score of a product p_i wrt risk set R, denoted by $s(p_i, R)$, as follows.

$$s(p_i, R) = \frac{1}{|R|} \sum_{r_k \in R} s(p_i, r_k) = \frac{1}{|R|} \sum_{r_k \in R} \sum_{j=1}^{m} w_{j,k} v_{i,j} \tag{2}$$

Finally, for the user with risk set R, we compute the score of each product $p_i \in P$ wrt R. Our recommendation system thus returns the user a ranked list of all feasible products sorted by their scores (note that the scores are also presented to the user so that the user knows why certain products rank higher or lower in our recommendation). To exemplify our model, consider our running example in Table I and Table II. Assume that all products are feasible for all users. Given a user U_1 with risk set $R_1 = \{r_0\}$ (i.e., the user has no health risk), the scores of each product for user U_1 (wrt R_1) is computed to be:

- $s(p_1, \{r_0\}) = w_{1,0} v_{1,1} + w_{2,0} v_{1,2} = 0.5 \cdot 0.8 + 0.5 \cdot 0.5 = 0.65$
- $s(p_2, \{r_0\}) = w_{1,0} v_{2,1} + w_{2,0} v_{2,2} = 0.5 \cdot 0.6 + 0.5 \cdot 0.5 = 0.55$
- $s(p_3, \{r_0\}) = w_{1,0} v_{3,1} + w_{2,0} v_{3,2} = 0.5 \cdot 0.2 + 0.5 \cdot 1 = 0.6$

Thus, the recommendation list for user U_1 is [p_1 (0.65), p_3 (0.6), p_2 (0.55)].

In our system, the set of weights W is initially determined by domain experts. Since most insurance terms are highly related to specific diseases, determining the weights does not require extremely high-demanding knowledge, but rather the basic knowledge in health insurance is sufficient. For instance, considering the risk *cancer risk*, higher weights can be assigned to terms like *payout for cancer*, and the weights for other terms can be decreased accordingly. In case of no domain experts available, Large Language Models could be applied to find the domain knowledge. The details are discussed in Section VIII-A.

V. Weight Refinement Component

We formed our linear scoring model to obtain *utility-based* recommendation based on the scoring attributes and a set W of weights. In this section, we present our weight refinement component, which updates W to include the dynamic user preferences captured by a user feedback dataset in a time period. We show the details of user feedback dataset in Section V-A, and then present how the weight refinement component handles the weight update task in Section V-B.

A. User Feedback Dataset

In our recommendation system, we present a ranked list of health insurance products ordered by how strongly we recommend a product to a user. However, users may have their own preferences to different products and their preferences might change over time, and it is thus vital to take users' preferences into account for recommendation. In order not to spoil the user experience, we would not include additional steps for users to give *explicit* feedback for our recommendation results. Thus, following existing online insurance recommendation

TABLE III: Example of User Feedback Dataset

Request	Session ID	Risk Set	Product List	Score List	Click List
q_1	1	$\{r_0\}$	$[p_1, p_3, p_2]$	$[0.65, 0.6, 0.55]$	$[0, 0, 1]$
q_2	2	$\{r_0\}$	$[p_1, p_3, p_2]$	$[0.65, 0.6, 0.55]$	$[0, 1, 0]$
q_3	3	$\{r_1, r_2\}$	$[p_1, p_3, p_2]$	$[0.665, 0.56, 0.555]$	$[0, 1, 0]$

systems [3], [13], [14], we take advantage of user click sessions, which is a common data source to *implicitly* capture user preferences widely applied in web search [5], [15].

A user click session (collected automatically when users perform web search) is a web session data containing a list of queries, a list of returned web pages for each query and users' browsing history of which web pages are clicked. It is natural that if a user has stronger preference to a web page, then s/he will be more likely to click this web page.

We also collect user click sessions to form our user feedback dataset. Specifically, in a user session, when a user starts a recommendation request, we record the recommendation list of health insurance products and which products in this list are clicked by the user. With the user feedback collector in our system (embedded in the user interface), user click sessions are collected in real-time and stored in a user feedback database.

Formally, our user feedback dataset F is a set of N requests. Each request $q \in F$ is associated with the ID of the session that contains request q, and the selected risk set in q. In addition, q has three *ordered* lists of the same size M_q, namely, the product list $L_q^P = [P_{q,1}, P_{q,2}, ..., P_{q,M_q}]$, the score list $L_q^S = [S_{q,1}, S_{q,2}, ..., S_{q,M_q}]$, and the click list $L_q^C = [C_{q,1}, C_{q,2}, ..., C_{q,M_q}]$. The l-th member of L_q^P (i.e., $P_{q,l}$) denotes the l-th ranked product in the recommendation list of request q. The l-th member of L_q^S (i.e., $S_{q,l}$) denotes the score of $P_{q,l}$ (wrt the risk set of q). The l-th member of L_q^C (i.e., $C_{q,l}$) is a *click variable* which equals 1 if the user clicks product $P_{q,l}$ in the recommendation list and 0 otherwise.

Table III shows an example of a user feedback dataset containing 3 requests, q_1, q_2 and q_3 (note that the timestamps are omitted for simplicity). For the first request q_1 in a session of ID 1, the user with risk set $\{r_0\}$ obtains his/her recommendation list of 3 products. It is clear that s/he clicked the 3rd-ranked product (i.e., p_2).

Now, we formalize the weight update task. Since our goal is to find the refined weights of the linear scoring model based on the user feedback within a time interval τ (e.g., the past one month), we form the user feedback dataset in τ, says F_τ, containing all the requests of which the timestamps are within τ. Let W be the original set of weights used in the linear scoring model for recommendation in the time interval τ. Given dataset F_τ, we would like to find the updated set of weights W' such that the user preferences in F_τ are captured and combined into the updated weights. Specifically, with the updated set of weights W', the product that receive some dynamic user preferences based on dataset F_τ should have a high chance to obtain a higher score than with the original set of weights W.

B. Weight Update Model

In this section, we present how we handle the weight update task from a user feedback dataset F_τ by adapting a state-of-the-art *click model* called GraphCM [5]. A click model considers a problem called *click prediction* which takes a dataset of user click sessions in web search as input. Given a new user click session containing a list of queries each with a list of returned web pages, the click model predicts whether a user will click each returned web page or not (by the output of click probability for each returned web page).

Our weight update task has similar input as click prediction based on how we formalize our user feedback dataset. Although our goal is different, we achieve it by observing that the click probability could represent users' preferences to the web pages (i.e., the health insurance products in our context).

In the following, we first introduce how we learn the click probabilities of health insurance products (representing the users' preferences) from dataset F_τ by adapting the click model, GraphCM. Then, we show how we update the weights by applying click probability learning.

1) Learning Click Probabilities: GraphCM resolves the click prediction problem by a homogeneous Graph Neural Network that models the dependencies among sessions, requests and products. A key observation for GraphCM is that, apart from the (intra-session) dependencies within each session, it effectively models the (inter-session) dependencies across different sessions. Inter-session dependencies are more significant in health insurance recommendation since we need to capture the overall user preferences that could be dynamic over a long time span to refine our recommendation instead of capturing the information of single sessions.

Specifically, GraphCM first initializes two homogeneous graphs, namely the request graph G_R (with nodes each for a request) and the product graph G_P (with nodes each for a product). In both graphs, edges are created between nodes with both intra-session and inter-session connections.

After G_R and G_P are constructed, each request and each product are first embedded into feature vectors. Then, GraphCM includes three modules to capture the dependency between a request q and the l-th ranked product $P_{q,l}$ in the product list of request q. The first module extracts the *context information* of q and generates a context representation of q by first sampling a set of K neighbors of q in G_R, then using a Graph Attention Network (GAT) to aggregate the information between q and all neighbors of q and finally generating the context representation of q with a Gated Recurrent Unit (GRU) on all requests in the same session of q. Similarly, the second module generates a context representation of $P_{q,l}$ by sampling K neighbors of $P_{q,l}$ in G_P, aggregating the information between $P_{q,l}$ and those neighbors with GAT and applying GRU on all products in the recommendation list of q. The third module encodes the *interaction* between q and $P_{q,l}$ by an element-wise product of their feature vectors. Besides, to fully utilize the neighborhood of $P_{q,l}$, this module also captures the interaction between q and each sampled neighbor of $P_{q,l}$ in G_P (in the same element-wise product form), and all the interactions are aggregated by GAT into the *neighbor interaction information* between q and $P_{q,l}$. The above three modules together estimate how likely a user will click $P_{q,l}$ in

TABLE IV: Example of Prediction Set of Simulated Requests

Request	Support	Risk Set	Product List	Score List	Click Probabilities
q'_1	2/3	$\{r_0\}$	$[p_1, p_3, p_2]$	$[0.65, 0.6, 0.55]$	$[0.1, 0, 0.9]$
q'_2	1/3	$\{r_1, r_2\}$	$[p_1, p_3, p_2]$	$[0.665, 0.56, 0.555]$	$[0.1, 0.9, 0]$
$q'_{2,1}$	1/6	$\{r_1\}$	$[p_1, p_3, p_2]$	$[0.665, 0.56, 0.555]$	$[0.1, 0.9, 0]$
$q'_{2,2}$	1/6	$\{r_2\}$	$[p_1, p_3, p_2]$	$[0.665, 0.56, 0.555]$	$[0.1, 0.9, 0]$

request q (i.e., the click probability).

Since learning click probabilities is a challenging task, there exists no simple, explainable but effective approach, and thus we apply GraphCM that could effectively produce accurate results. However, GraphCM is a complex deep learning model that is not easily explainable as required in the insurance industry [8], [9]. We thus discuss some insights of how the explainability of GraphCM could be improved with its Attention network in Section VIII-B.

2) Updating Weights: Now, we introduce how we tackle our weight update task by applying click probability learning. Given the user feedback dataset F_τ and the original set of weights W in the recommendation model, we would like to obtain the updated set of weights W' based on users' preferences that are represented by the click probabilities. Importantly, if a product is found to receive strong user preference (i.e., have a high click probability), then this product should have a higher score with W'.

One way of modeling the weight update task is to (a) first find user preferred *attributes* from the learned user preferences towards products, (b) then assign larger values to more preferable attributes as the "rewarding weights" and (c) finally combine the "rewarding weights" with the original weights directly to form the updated weights. However, to perform the above Step (a), a complex deep learning model needs to be exploited [16], which is not explainable.

As regulated by authorities, explainability is of great importance in the insurance industries [8], [9], and thus we form our explainable intuition for the weight update task as follows. We directly give a *rewarding score* to a product p_i if we find that a user has strong preference on p_i. Then, we form an "expected" score of p_i (wrt this user) combining the original score of p_i with the rewarding score of p_i. After obtaining the expected scores for all products, we aim to find the updated set of weights W' such that the new scores for all products computed based on W' are as close as the expected scores. Since our scoring model is linear, the above problem can be effectively solved by linear regression, where the obtained regression coefficients are exactly the updated weights to be found. Since linear regression is explainable and transparent, the explainability is satisfied.

Moreover, for understandable recommendation to users, we model the weights separately for various risks, which actually have correlations with each other (either from *observed* patterns in the user feedback dataset, or from their *intrinsic* relevancy). We thus consider all the observed patterns of different risks instead of only single risk when obtaining user preferences. We also update the weights for some risks *not* observed in the dataset that have intrinsic relevancy with an observed risk, which could address the *cold start* problem here.

Based on the above intuitions, we handle the weight updating task in the following steps. (1) We train an adapted GraphCM model $\mathcal{M}$ where the input is the dataset F_τ such that model $\mathcal{M}$ could capture the user feedback information in F_τ. (2) We generate a concise prediction set F' of *simulated* requests according to the observed patterns in F_τ. (3) We predict the click probabilities for the product list in each request q' in F' with $\mathcal{M}$, which represent the preferences of users with the health condition in q'. (4) We give a *rewarding score* to a product p_i wrt an observed risk r_k if users with risk r_k show preferences to product p_i, and the same to a non-observed risk r_o if r_o and r_k have intrinsic relevancy. (5) We combine the rewarding score with the original score computed from W to form the expected score of p_i wrt r_k. (6) We compute the updated weights with a linear regression based on the expected scores of all products. Since Step (1) is straightforward, we formalize the details of the remaining steps.

In Step (2), to capture the risk patterns observed in F_τ into a prediction set F', we directly find all the health conditions (each is a set/pattern of multiple risks or a single risk) appearing in F_τ and insert them into a set $\mathcal{R}_\tau$. Then, for each health condition R' in $\mathcal{R}_\tau$, we issue a simulated request q' with risk set R' (in a new session) to obtain the ranked list of all products. The *support* of q', denoted by $sup(q')$, is defined to be the proportion of requests in F_τ with the risk set of q' (i.e., R'), which will be later used to decide how much each risk in R' contributes to the final update of weights. Continuing our running example, the prediction set constructed from our user feedback dataset (in Table III) is shown as the black marked part in Table IV, which contains 2 simulated requests q'_1 and q'_2 with risk set $\{r_0\}$ (support $= 2/3$) and risk set $\{r_1, r_2\}$ (support $= 1/3$), respectively.

In Step (3), for each simulated request q' in F' with the product list $L^P_{q'}$ and the score list $L^S_{q'}$ (note that the size of both lists is the total number of products n), we predict the click probability of each product $P'_{q',l}$ in $L^P_{q'}$ (for $l \in [1, n]$), which represents the preference on product $P'_{q',l}$. Clearly, all products in P will be assigned a preference in request q', and we thus denote $prf(p_i, q')$ to be the preference on product p_i in request q'. To exemplify, in Table IV, the click probabilities are obtained in the form of a list for each simulated request. Thus, we can see that for request q'_1, $prf(p_1, q'_1) = 0.1$.

In Step (4), we aim to find the preference to each product of users with *one* specific risk, so that the weight update problem will be solved in a simpler form. First, for a request q' in F' with multiple risks, we split q' into several sub-requests each with a single risk in R'. The support of q' will be evenly allocated to each sub-request to ensure that each risk in R' contributes equally to the result. We denote F'' to be the prediction set after the above splitting. In our running example, request q'_2 is split into two sub-requests $q'_{2,1}$ and $q'_{2,2}$ with equal support ($= 0.5 \cdot 1/3 = 1/6$). The obtained F'' after splitting thus contains 3 requests, q'_1, $q'_{2,1}$ and $q'_{2,2}$. Then, we could obtain the *rewarding score* of a product p_i wrt a risk r_k, denoted by $rew(p_i, r_k)$, to be the weighted sum of the preferences of p_i in all requests in F'' with risk set $\{r_k\}$ and

their supports. Formally,

$$rew(p_i, r_k) = \sum_{q'' \in F'' \text{ with risk set } \{r_k\}} sup(q'') \cdot prf(p_i, q'') \quad (3)$$

Consider risk r_0 in q'_1 from Table IV. The rewarding scores of p_1, p_2 and p_3 wrt r_0 are 0.067, 0.6 and 0, respectively.

So far, we only include the observed health conditions in dataset F_τ. Then, the weights related to the non-observed health risks could not be updated, which could induce the so-called *cold start* issue. To alleviate this issue, we exploit the intrinsic relevancy among different risks (modelled in matrix $\mathcal{I}$) by also assigning some rewarding scores to the non-observed risks that are intrinsically relevant to the observed risks. We first insert all the observed health risks (i.e., those appearing in F'') into a set $\mathcal{R}''$. For each non-observed risk r_o (i.e., $r_o \in \mathcal{R} \setminus \mathcal{R}''$), we compute $rew(p_i, r_o)$ for each product p_i as follows, based on the rewarding score of p_i wrt r_k and the intrinsic relevancy between r_o and r_k in the matrix $\mathcal{I}$.

$$rew(p_i, r_o) = \sum_{r_k \in \mathcal{R}''} \mathcal{I}_{o,k} \cdot rew(p_i, r_k) \quad (4)$$

Then, we insert into set $\mathcal{R}''$ all the non-observed risks wrt which the rewarding score of any product is non-zero, so that $\mathcal{R}''$ includes all the concerned health risks for weight updating.

In Step (5), we form the "expected" score of p_i wrt r_k by combining the original score computed from W (i.e., $s(p_i, r_k)$ in Equation 1) and the rewarding score $rew(p_i, r_k)$. Denoting the expected score of p_i wrt r_k by $exp(p_i, r_k)$, we could have various ways for this combination. Inspired by [5], we explore the following three functions, namely, the *weighted addition* (Equation 5), the *weighted multiplication* (Equation 6) and the *exponential multiplication* (Equation 7),

$$exp(p_i, r_k) = s(p_i, r_k) + \alpha \cdot rew(p_i, r_k) \quad (5)$$

$$exp(p_i, r_k) = s(p_i, r_k) \cdot (1 + \alpha \cdot rew(p_i, r_k)) \quad (6)$$

$$exp(p_i, r_k) = s(p_i, r_k) \cdot (1 + rew(p_i, r_k))^\alpha \quad (7)$$

where α is a non-negative real number called the *updating rate* to control how much the rewarding scores will affect the expected scores. Continuing the example, if we consider the exponential addition and α is set to 0.3, the expected scores of p_1, p_2 and p_3 wrt r_0 will be 0.6701, 0.73 and 0.6, respectively.

In Step (6), finally, we would like to find the updated weight set $W' = \{w'_{j,k} | j \in [1, m], k \in [0, |\mathcal{R}|]\}$, where $w'_{j,k}$ is the updated weight of scoring attribute t_j wrt risk r_k. Consider a risk r_k in set $\mathcal{R}''$. We form a linear regression to find the updated weights $w'_{j,k}$ for $j \in [1, m]$ such that the following sum-of-squared error (a commonly used form for linear regression) is minimized: $\sum_{i=1}^{n} (s'(p_i, r_k) - exp(p_i, r_k))^2$, where $s'(p_i, r_k) = \sum_{j=1}^{m} w'_{j,k} v_{i,j}$ denotes the updated score of p_i wrt r_k (applying the same form as Equation 1).

After solving the linear regression for each $r_k \in \mathcal{R}''$, we have found all the updated weights in W', and then we replace W with W' in our linear scoring model for recommendation. In our example considering risk r_0, we can form a linear regression to find the new weights $w'_{1,0}$ and $w'_{2,0}$ for attribute t_1 and t_2, respectively, which are related to risk r_0. With the expected scores computed in Step (5), it can be verified that the linear regression results are $w'_{1,0} = 0.562$ and $w'_{2,0} = 0.438$ (which will replace the original weights $w_{1,0} = w_{2,0} = 0.5$). Then, the updated scores of p_1, p_2 and p_3 wrt r_0 will be 0.669, 0.556 and 0.551, respectively. We can observe that p_2 (which ranks the 3rd in request q'_1 under original weights) ranks the 2nd under the new weights wrt to r_0, since p_2 receives large user preferences from user with risk r_0.

One can see that our updated weights W' are based on the expected scores that combine the original scores with the rewarding scores. Thus, W' integrates both the new user preferences encoded into rewarding scores, and the utility-based information "inherited" from the original scores. In our system, we could set the updating rate α in a controllable range to ensure that the weight updating is reasonably affected by the new user preferences. Thus, our system achieves a balanced combination of utility-based recommendation and user feedback based recommendation.

VI. Related Work

A. Recommendation in Insurance

Traditional recommendation systems exploit users' ratings with two main technologies, namely content-based filtering [17] and collaborative filtering [18], based on the similarity of items and the similarity of users, respectively. However, those technologies are more suitable for recommendation of movies, music, books, etc, which are based on users' taste or favor, and they could not be directly employed for the insurance domain. This is because (1) users lack knowledge of insurance to make proper ratings, (2) there hardly exists close similarity among users or items in insurance domain due to its complexity, and (3) decisions on health insurance should be made more carefully than based on taste or favor [4].

In the insurance domain, some recommendation systems have been proposed recently [19], [20]. [19] makes recommendation with a similar idea of collaborative filtering but finds similar users by their demographic data instead of their taste. [20] exploits a Bayesian Network which demonstrates good recommendation performance in insurance. However, both approaches rely on the previous insurance purchase records of users, which are sensitive data within a company, and thus those approaches are limited to inner-company recommendation. In this work, we need to give recommendation to a user among cross-company insurance products.

The approaches based on multi-criteria decision making have been a recent trend for insurance domain (particularly applied in health insurances) [2], [4], which adopts the idea of utility-based recommendation [10], [11]. Utility-based recommendation aims to find a suitable utility function for recommendation, which could well encode the complex attributes of health insurances and user preferences. However, these existing systems require users to specify their preferences manually.

Another recent approach utilizes users' click sessions to automatically capture users' dynamic implicit preferences for health insurance recommendation [3]. This work, however,

relies solely on the user feedback and does not consider any utility. Moreover, this work only exploits some simple approaches (e.g., long short-term memory), but we explore some more effective approaches to handle users' click sessions.

Our work generally follows the scheme of utility-based recommendation (but we include an expert-assistant data-processing that avoids heavy input from users), and we also exploit the user feedback in our recommendation model (but we ensure that user feedback is not excessively influencing).

B. Click Models

Click models are the popular approaches to the problem of finding users' implicit preferences reflected in the click behavior of users' click sessions in web search. Recent state-of-the-art click models apply Graph Neural Network that shows powerful capability of solving this problem [5], [15]. One branch of click models focuses on the intra-session information, which better handles the recommendation based on users' instant feelings. A state-of-the-art model in this branch called *CACM* [15] captures the session context information by a session-flow graph to well model the context-aware relevance of the searched results. However, in the health insurance domain, users' preferences over a long time span are more important. Thus, we are more interested in another branch that captures both intra-session and inter-session information. We thus select an effective state-of-the-art model GraphCM [5] to adapt in this work, as we introduced in detail in Section V-B1.

VII. Experimental Evaluation

A. Setup

1) Datasets: For the health insurance dataset, we maintain 164 products collected from 19 major insurance providers in Hong Kong, and each product has 203 scoring attributes after data processing. For the user feedback dataset, we collected all the user click sessions in 2022 containing 3,673 valid user requests. The number of user clicks among those requests is 2,203 (note that a user may not click any product in the recommendation list). Following [5], we split the user feedback dataset into training, validation and test set with proportions 80%, 10% and 10%, respectively.

2) Baselines: To learn dynamic user preferences from user click sessions (represented by the click probabilities), the existing health insurance recommendation system [3] uses two models, a feed forward neural network (FFNN) model (which encodes each recommendation list as a feature vector and returns the probabilities), and a long short-term memory (LSTM) model (which takes sequences of recommended products as input and returns the probabilities). We include FFNN and LSTM as well as another advanced click model CACM [15] (which focuses on intra-session information) as baselines to verify the effectiveness of exploiting GraphCM in our system.

3) Measurement: We measure how accurate user preferences are captured from user click sessions (i.e., the accuracy of click prediction) using two metrics defined in [5], namely, the *log-likelihood* (denoted by LL, smaller value preferred) and the *perplexity* (denoted by PPL, larger value preferred).

TABLE V: Results of Click Prediction

Model	LL	PPL	Training Time
FFNN	0.5281	1.1731	**322s**
LSTM	0.4735	1.2403	541s
CACM	0.4337	1.3502	664s
GraphCM (Our System)	**0.3374**	**1.5060**	723s

TABLE VI: Recommendation Quality by Weight Updating

Model	$avgNDCG$	MRR@20
Initial System (before Weight Updating)	0.2480	0.1385
FFNN	0.2446	0.1344
LSTM	0.2467	0.1367
CACM	0.2552	0.1463
GraphCM (Our System)	**0.2620**	**0.1550**

Also, to measure how good a recommendation list is to a user, we apply two commonly used metrics, the average *Normalized Discounted Cumulative Gain* [21] over all the recommendation lists in a dataset (denoted by $avgNDCG$, larger value preferred), and the *Mean Reciprocal Rank* at 20 [22] (denoted by MRR@20, larger value preferred).

4) Implementation: In our system, we run the first component (handling the recommendation task) in a product environment. Our second component (handling the weight refinement task) is implemented in Python in a machine with 3.6GHz CPU, 32GB memory and Nvidia GTX1060 GPU. The code and sampled datasets are available in our supplementary materials [12]. We also vary our updating rate α to study how it influences the recommendation quality by affecting the weight refinement in our second component.

B. Evaluation Results

1) Click Prediction: We first show the click prediction results that compare our system using GraphCM and the baseline models. As shown in Table V, all baseline models obtain much less accurate prediction than GraphCM, with much higher LL and much lower PPL. This is because GraphCM has a more powerful structure to capture user's preferences from click sessions, which fully explores the intra-session and inter-session information. CACM outperforms the other two baselines since it uses an advanced graph neural network that is effective to capture the intra-session information (but fails to capture the inter-session information).

Regarding the training, although GraphCM needs longer time for training (723s), the training process is done in our weight refinement component which is not frequently executed (e.g., on a monthly basis), and thus the execution time of weight refinement including training is still acceptable.

2) Recommendation Quality by Weight Updating: We compare the recommendation quality (measured by $avgNDCG$ and MRR@20) of each model after weight updating with the initial system before weight updating. For each model, we test with a large range of various alpha values (from 0.1 to 1000) and report the obtained optimal metric. As presented in Table VI, our system obtains the best improved recommendation quality of both metrics (i.e., 5.6% and 11.9% improvement of $avgNDCG$ and MRR@20, respectively, compared with the initial system). This is because our system accurately

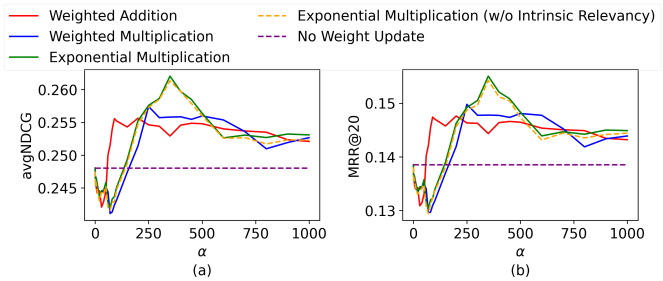

Fig. 2: Ablation and Variants Studies

captures user preferences from user click sessions and gives high rewarding scores to products that are more preferable. Then, after the corresponding weight refinement (with the effective linear regression), those preferable products will have a higher rank in the updated recommendation lists, which leads to a better recommendation quality. All the baselines fail to improve the recommendation quality due to their inaccuracy of capturing user preferences from user click sessions. In particular, the recommendation quality even degrades after weight updating with the FFNN and LSTM model.

3) Ablation and Variants Studies: We also conduct ablation and variants to verify the effectiveness of our system design. In those studies, we vary the updating rate α from 0.1 to 1000.

Our System without User Feedback. When the user feedback is not considered, our system does not perform any weight update, and thus the system always operates on the initial weights. As shown in Figure 2(a) and (b), for most of the α values and both metrics, the result of no weight update shows lower recommendation quality compared with our system after weight updating no matter which variant is used.

Our System without Utility. We also test our system without considering utility, which directly assigns the rewarding scores to the expected scores (without combining with the original scores) in Step (5) of our weight updating approach. We observe two critical issues for this variant. One issue is that each weight update will only be determined by the current user feedback dataset being processed, but the previous user preferences encoded in the original scores are lost. The other issue is that the updated weights are likely to contain unreasonable results. In the experiment results, we find many cases where heart disease related attributes receive much larger weights than cancer related attributes for a user with cancer risk, which is unacceptable in our system.

Our System without Intrinsic Relevancy. Without considering the intrinsic relevancy among risks, in Step (4) of weight updating, we only assign rewarding scores to products wrt observed risks in the dataset. As illustrated in Figure 2(a) and (b), when we fix the forming expected score function to exponential multiplication, our system without intrinsic relevancy slightly degrades in recommendation quality.

Variants of Expected-score-forming Functions. We also study the effect of using different functions of forming the expected scores in Step (5) of weight updating. Figure 2(a) and (b) show very similar trends on the two metrics. When α is varied from 0.1 to 1000, the recommendation quality first drops, then increases significantly and finally decreases to a stabilized value for all the three variants. The reason is that when α is too small, the linear regression could be affected by imbalanced scores (since only a few expected scores are raised while most of them are nearly unchanged) and thus inaccuracy could be caused, and when α is too large, some less preferable products could also obtain more rewarding scores and thus the capability of raising up the ranks of more preferable products will be degraded. As such, setting α in a reasonable middle range could optimize the recommendation quality. Among all the variants, the exponential multiplication function gives the best recommendation quality (i.e., $avgNDCG = 0.262$ and $MRR@20 = 0.155$).

4) Summary: In our experimental evaluation, we verified that our system using GraphCM could accurately capture the user preferences (represented by click probabilities) with much lower LL (0.3374) and much higher PPL (1.5060) than the best baseline (i.e., CACM with $LL = 0.4337$ and $PPL = 1.3502$) and with reasonable training time (723s). Our system effectively improves the recommendation quality of the initial system with $avgNDCG$ ($MRR@20$) increasing by 5.6% (11.9%), while the best baseline only have 2.9% (5.6%) improvement, and thus, our system is around 2x more effective in improving the recommendation quality than baselines.

VIII. Discussions

A. Domain Expert Involvement

In some components of our system, domain experts in health insurance are involved, which is necessary in the insurance industry. This is because the decision-making in insurance relies heavily on the domain knowledge and is highly regulated by authorities. Such systems need careful expert audit to ensure that the users obtain accountable recommendations that are aligned with industry regulations, especially for the health insurance. Therefore, expert involvement is common in existing online systems of health insurance recommendation [2], [4].

In our system, we have minimized expert input, since the major efforts of expert input lie in deciding the intrinsic relevancy among health risks and initializing the weights representing the importance of each scoring attributes wrt each risk. However, in case domain experts are barely available, Large Language Models (LLMs) [23], [24] could be applied to find the needed domain knowledge in a question-answering form, which has been a popular approach [25] due to the recent industrial advances of LLMs like GPT-3.5 [24].

In particular, we would like to quantify how strong the relevancy between two risks (e.g., *heart disease risk* and *lung cancer risk*) is, or how important a scoring attribute (e.g., *payout for cancer*) is considering a health risk (e.g., *lung cancer risk*). Taking the latter case as an instance, we form the following question to an LLM (i.e., GPT-3.5): *give a score between 0 and 5 indicating the importance of insurance term "payout for cancer" considering a user with lung cancer risk*. In most cases, the answers from GPT-3.5 with detailed reasoning are consistent with the answers from our domain experts.

B. Explainability of GraphCM

In Section V-B, to tackle the problem of learning click probabilities effectively and accurately, we apply a deep learning model GraphCM [5]. However, this model might not satisfy the explainability requirement. Learning click probabilities in an explainable manner is non-trivial and unexplored in the literature (to the best of our knowledge). Thus, developing an explainable click model is left as a future work beyond this paper.

Here, we discuss some insights on how the explainability of GraphCM could be improved. GraphCM incorporates Graph Attention Networks (GATs) as major building blocks, and recent studies have explored to explain attention-based neural networks with attention scores [26], [27]. Specifically, the nodes or features with higher attention scores are more influential to decide the result. Thus, we explain GraphCM as follows.

For the GATs on the graph of requests, the graph of products and the request-product interaction, when predicting the click probability of a new requested q, we find the request nodes, the product nodes and the pairs of interacting request-product that have the highest attention scores in the 3 GATs, respectively. They implicate the information of the requests and products that are most relevant with q and the most influential request-product pairs which affects the predicted click probabilities.

IX. Conclusion

In this paper, we propose an automatic personalized health insurance recommendation system that incorporates both utility-based and user feedback based recommendation with an effective two-component architecture. We run our system on a complex insurance dataset in a real project, which shows superior results compared with baselines. Some future directions include how to model the scoring function for each insurance term from raw values and how to design a fair scoring scheme to obtain recommendation without biases.

Acknowledgment

We are grateful to the anonymous reviewers for their constructive comments on this paper. The research is supported by PRP/026/21FX with the support of company "Simply Solution HK Limited".

References

[1] "Market performance of general insurance business in 2021," 2021. [Online]. Available: https://www.ia.org.hk/english/infocenter/statistics/files/GB_Market_Overview_2021_Eng_FINAL.pdf

[2] A. Abbas, K. Bilal, L. Zhang, and S. U. Khan, "A cloud based health insurance plan recommendation system: A user centered approach," *Future Generation Computer Systems*, vol. 43, pp. 99–109, 2015.

[3] S. Borg Bruun, "Learning dynamic insurance recommendations from users' click sessions," in *Fifteenth ACM Conference on Recommender Systems*, 2021, pp. 860–863.

[4] A. Rani *et al.*, "Multi criteria decision making (mcdm) based preference elicitation framework for life insurance recommendation system," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 2, pp. 1848–1858, 2021.

[5] J. Lin, W. Liu, X. Dai, W. Zhang, S. Li, R. Tang, X. He, J. Hao, and Y. Yu, "A graph-enhanced click model for web search," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 1259–1268.

[6] A. J. Barnes, Y. Hanoch, and T. Rice, "Determinants of coverage decisions in health insurance marketplaces: Consumers' decision-making abilities and the amount of information in their choice environment," *Health Services Research*, vol. 50, no. 1, pp. 58–80, 2015.

[7] A. Asgary, K. Willis, A. A. Taghvaei, and M. Rafeian, "Estimating rural households' willingness to pay for health insurance," *The European Journal of Health Economics, formerly: HEPAC*, vol. 5, no. 3, pp. 209–215, 2004.

[8] "Where the us is going on ai regulation of insurance," 2022. [Online]. Available: https://www.eversheds-sutherland.com/documents/sectors/tmt/global-tech-week/US-AI-regulation-of-insurance.pdf

[9] "Us insurance regulators' perspectives on artificial intelligence," 2021. [Online]. Available: https://www.mayerbrown.com/en/perspectives-events/publications/2022/10/us-insurance-regulators-perspectives-on-artificial-intelligence

[10] W. Wang, R. C.-W. Wong, and M. Xie, "Interactive search for one of the top-k," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 1920–1932.

[11] Y. Zheng, "Utility-based multi-criteria recommender systems," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 2019, pp. 2529–2531.

[12] L. Hao and R. C.-W. Wong, "Automatic personalized health insurance recommendation based on utility and user feedback (supplementary material)," 2023. [Online]. Available: https://github.com/satansin/HealthInsuranceRecommendation

[13] Y. Li, Y. Zhang, L. Gan, G. Hong, Z. Zhou, and Q. Li, "Revman: Revenue-aware multi-task online insurance recommendation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 1, 2021, pp. 303–310.

[14] Y. Li, Y. Zhang, W. Wu, Z. Zhou, and Q. Li, "Posgen: Personalized opening sentence generation for online insurance sales," in *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, 2022, pp. 1874–1879.

[15] J. Chen, J. Mao, Y. Liu, M. Zhang, and S. Ma, "A context-aware click model for web search," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 88–96.

[16] Y. Xian, T. Zhao, J. Li, J. Chan, A. Kan, J. Ma, X. L. Dong, C. Faloutsos, G. Karypis, S. Muthukrishnan *et al.*, "Ex3: Explainable attribute-aware item-set recommendations," in *Proceedings of the 15th ACM Conference on Recommender Systems*, 2021, pp. 484–494.

[17] C. C. Aggarwal, "Content-based recommender systems," in *Recommender systems*. Springer, 2016, pp. 139–166.

[18] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The adaptive web*. Springer, 2007, pp. 291–324.

[19] W. Xu, J. Wang, Z. Zhao, C. Sun, and J. Ma, "A novel intelligence recommendation model for insurance products with consumer segmentation," *Journal of Systems Science and Information*, vol. 2, no. 1, pp. 16–28, 2014. [Online]. Available: https://doi.org/10.1515/JSSI-2014-0016

[20] M. Qazi, G. M. Fung, K. J. Meissner, and E. R. Fontes, "An insurance recommendation system using bayesian networks," in *Proceedings of the eleventh ACM conference on recommender systems*, 2017, pp. 274–278.

[21] K. Järvelin and J. Kekäläinen, "Ir evaluation methods for retrieving highly relevant documents," in *ACM SIGIR Forum*, vol. 51, no. 2. ACM New York, NY, USA, 2017, pp. 243–250.

[22] T. Chen and R. C.-W. Wong, "Improving representation learning for session-based recommendation," in *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, 2022, pp. 854–863.

[23] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[24] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[25] K. Singhal, T. Tu, J. Gottweis, R. Sayres, E. Wulczyn, L. Hou, K. Clark, S. Pfohl, H. Cole-Lewis, D. Neal *et al.*, "Towards expert-level medical question answering with large language models," *arXiv preprint arXiv:2305.09617*, 2023.

[26] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[27] D. Neil, J. Briody, A. Lacoste, A. Sim, P. Creed, and A. Saffari, "Interpretable graph convolutional neural networks for inference on noisy knowledge graphs," *arXiv preprint arXiv:1812.00279*, 2018.

(a) (b) (c) (d)

Fig. 3: UI Flows of our Automatic Personalized Health Insurance Recommendation System

APPENDIX A
DEPLOYMENT AND UI

We show some representative UI flows of our deployed recommendation system for health insurance. In Figure 3(a), the user inputs the demographic attributes (e.g., age and gender) and the returned recommended insurance products will be filtered by those demographic attributes. Note that, for sensitive attributes (e.g., age), instead of the detailed value, we only need a value range (e.g., age between 16 and 20) for the input so that our system could better protect the privacy of users. In Figure 3(b), our system presents a list of health risks in terms of common illnesses for the user to select. In Figure 3(c), our system shows a list of recommended insurance products to the user sorted by their scores. The basic information and the score of each product is presented in the list. For privacy, we hide the product names in the figure. In Figure 3(d), after the user clicks a recommended product (e.g., the first product in the list), our system shows a hierarchical report of how the total score of the product is computed (based on our linear scoring model) so that the user could understand why this product is recommended. In the hierarchical report, each health risk of the user has a subtotal score, which averages to the total score. Under each risk type, each insurance term (i.e., a scoring attribute) has a term score and its raw value for this term. For example, the term "1st claim payout" score equal to 3 with its raw value equal to 1. Note that, although we use normalized scores in our recommendation model, the real scores shown to the user are scaled based on the units in the insurance domain. The report also presents the weights of scoring attributes, which are scaled and represented by the product of two values *relevance* and *importance* that are easier for users to understand. Moreover, since an insurance product has many terms, we organize those terms into groups in the hierarchical report (e.g., a group contains all insurance terms related to cancer) for users to explore those terms more easily.