Appendix

This appendix provides the supplementary research material including detailed descriptions of data and menu generation artificial intelligence.

1 Menu generation artificial intelligence

1.1 Data

In this work, we used food and diet data. Food data include the nutrients and ingredients of 1,726 foods and consist of 20 nutrition columns and 298 ingredient columns. Table 1 shows a sample row of the food data.

Table 1: Sample row of the foods dataset

Food	**Energy(kcal)**	**Carbohydrate(g)**	...	**Cabbage**	**Pumpkin**
Kimchi	6.250	1.100	...	1	0

Food data consist of the nutrition, ingredients, and group information. The nutrition part shows the nutrients in a standard serving of each food. The ingredients information signifies whether the specific food contains a specific ingredient. For example, kimchi contains cabbage but not pumpkin, so the cabbage column is indicated as 1, and the pumpkin column is indicated as 0. The group information indicates the type of food such as soup, main dish, and side dish. In this dataset, foods are grouped according to the list of dishes in Korean cuisine.

The diet dataset comprises a standard diet from one of the government Centers for Children's Food Service Management Korea. This center was established by the Ministry of Food and Drug Safety to support hygiene and nutrition management for daycare centers and kindergartens. The diet provided in this dataset was designed by professional nutritionists and disclosed to the public. These data are used as a reference for planning diets in kindergartens. Each row is a collection of food items, and the entire dataset consisted of 220 rows consumed from February to November 2019 (excluding weekends and holidays). Table 2 shows a sample of the diet dataset. As shown in Table 2, the 1st and 2nd cells are morning snacks, the 3rd to 7th are lunch, the 8th and 9th are afternoon snacks, and the 10th to 14th are dinner, representing the standard daily diet of a Korean nursery and does not include breakfast.[1] Each diet plan is described in order of main dish-soup-side dish. Not all 14 cells are always filled, some are empty for diet plans consisting of fewer kinds of nutritionally complete foods.

[1]Due to the lack of space, we illustrate our sample as multiple rows in Table 2. However, the real sample in our dataset is unfolded into one row.

Table 2: Sample row of the foods dataset

Date	**Menu**				
July 2	**morning 1**	**morning 2**			
	Carrot stick	milk			
	lunch 1	**lunch 2**	**lunch 3**	**lunch 4**	**lunch 5**
	Jjajangbap	Miso soup	Dumplings	Kimchi	(empty)
	afternoon 1	**afternoon 2**			
	White Bread	Yogurt			
	dinner 1	**dinner 2**	**dinner 3**	**dinner 4**	**dinner 5**
	White Rice	Fish cake soup	Boiled pork slices	Cucumber salad	Kimchi

1.2 Artificial Intelligence

In this section, we describe the methods that underlie the two AI solutions of Generative Adversarial Nets and Reinforcement Learning.

1.2.1 Generative Adversarial Nets

Generative Adversarial Nets (GAN) is a framework for training the generator model through its competition with the discriminator model. In GAN, the generator model (G) and the discriminator model (D) are initially trained with real data. The purpose of G is to estimate the real data distribution, and the purpose of D is to distinguish between real and generated data. G generates data that mimic the real data based on the estimated data distribution, and D returns the probability that inputted data were derived from real data. Therefore, the purpose of G is to ensure that D recognizes the data generated by G as real data (not generated). It is proven that the output of D converges to 0.5 when G and D are ideal (i.e., when G and D are well-trained)[1].[2]

Entering each food directly into the GAN is not effective. Because GAN uses only real data and only 752 have been used in real data among 1,726 foods, then the generated diet will consist of only these 752 items of food. To utilize all the 1,726 foods, we categorized foods and transformed the diet data from the level of individual food to the level of food category. This way, foods that were not used in the daily diet data can appear in the generated diet if other foods in the same category have been used. First, we categorized the foods that make up the diet using the Affinity propagation clustering algorithm [2]. Each set of foods produced through this categorization is called a food cluster, and each cluster is numbered. Categorization was based on the ingredients that make up the food and created 154 clusters. For example, a list of foods belonging to Cluster 6 includes dried-shrimp potato stew, dried-shrimp chard stew, dried-shrimp miso stew, dried-shrimp consommé, dried-shrimp daikon stew, and tofu

[2]D has nothing but tossing a coin to judge whether the inputted data is real or not as G succeeds to perfectly deceive D.

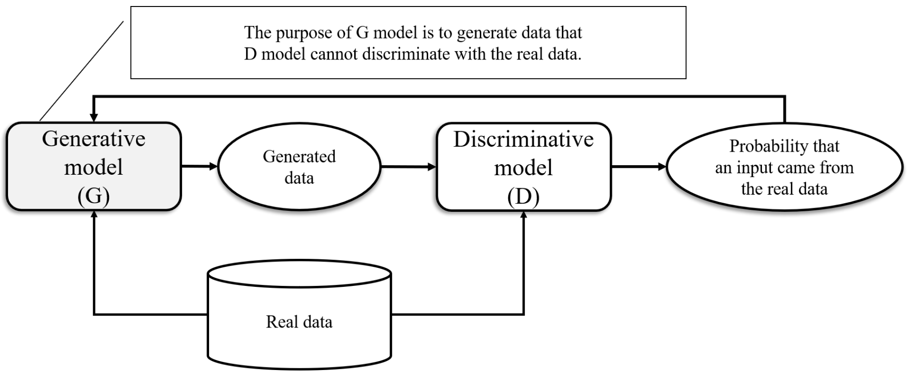
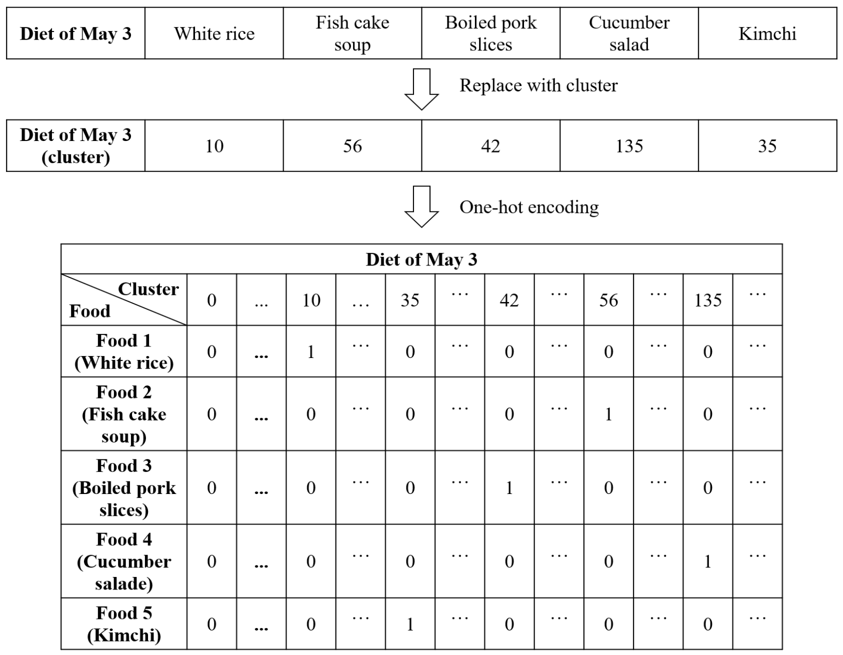

The purpose of G model is to generate data that
D model cannot discriminate with the real data.
Generative model (G)
Generated data
Discriminative model (D)
Probability that an input came from the real data
Real data

Figure 1: The framework of Generative Adversarial Nets

stew. We can see that most of Cluster 6 consists of stews with dried shrimp. After categorization, we replaced each food in the diet data with the index number of the relevant cluster. For example, dried-shrimp potato stew belongs to Cluster 6, so this dish is replaced by a '6'. Finally, we encode the clusters of diet data into binary format of their relevant foods. The cell where the food item (row) belongs to the cluster number (column) is indicated by a '1,' and the remaining cells are zeros. Figure 2 shows an example of this transformation process, changing diet data into the data that can be used for GAN. We used GAN to train a diet-generating model with the real data.

The generator model learned through the GAN process generates diets of food clusters in the form shown in Table 2. We should revert these cluster-level results back to original food-level form. For this, we return the encoded data to cluster number data through a reverse of the transformation process described earlier. Then, each cell is replaced with a randomly selected food in the relevant cluster. Figure 3 shows an example of this restoration process.



Diet of May 3	White rice	Fish cake soup	Boiled pork slices	Cucumber salad	Kimchi

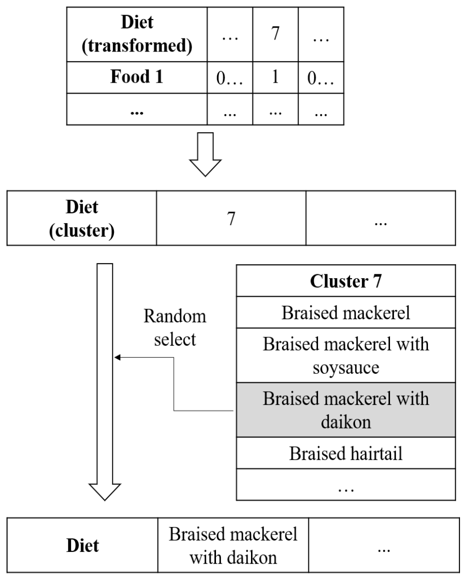
Replace with cluster

Diet of May 3 (cluster)	10	56	42	135	35

One-hot encoding

Diet of May 3											
Cluster / Food	0	...	10	…	35	…	42	…	56	…	135	…
Food 1 (White rice)	0	...	1	…	0	…	0	…	0	…	0	…
Food 2 (Fish cake soup)	0	...	0	…	0	…	0	…	1	…	0	…
Food 3 (Boiled pork slices)	0	...	0	…	0	…	1	…	0	…	0	…
Food 4 (Cucumber salade)	0	...	0	…	0	…	0	…	0	…	1	…
Food 5 (Kimchi)	0	...	0	…	1	…	0	…	0	…	0	…

Figure 2: Data transformation process from food to cluster



Diet (transformed)	…	7	…
Food 1	0…	1	0…
...	...	...	...

Diet (cluster)	7	...

Cluster 7
Braised mackerel
Braised mackerel with soysauce
Braised mackerel with daikon
Braised hairtail
…

Diet	Braised mackerel with daikon	...

Figure 3: Data restoration process from cluster to food

1.2.2 Reinforcement Learning (RL)

Reinforcement learning (RL) evolved out of efforts to model animal behavior such as mice trying to find their way through a maze to a reward [3]. RL aims to develop an agent to learn the behavior (i.e., policy) that reinforces the correct decision (i.e., action) through a framework of trial-and-error interactions in a dynamic environment [4] (See Figure 4). RL is a principled mathematical framework, where an agent learns correct decisions by experiencing consecutive situations and optimizes her sequential decisions. As a result, for the past few years RL has arisen as one of the most promising method in AI research that studies sequential decision-making over time.

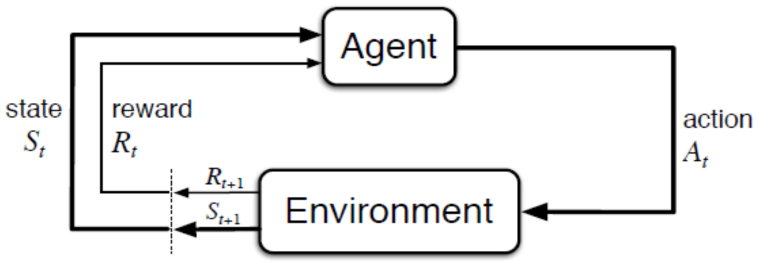


Figure 4: The agent - environment interaction in reinforcement learning [5]

The task of RL is usually described as a Markov Decision Process (MDP), a class of stochastic sequential decision processes in which each decision only depends on the previous decisions and their outcomes [5, 6]. A MDP consists of:

- a set of states $\mathcal{S}$
- a set of actions $\mathcal{A}$
- a reward function $\mathbb{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$
- a discount factor γ
- a transition function $\mathbb{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{P}$

where $\mathcal{S}$ is a set of observable states, and each state is an agent's observation of a situation. $\mathcal{A}$ is a set of feasible actions from which the agent selects to act. The agent behaves differently according to the current state. How an agent will behave depends on the policy π, which is the parameterized probability distribution of actions and defines the probability that an agent will select each action at the given state.

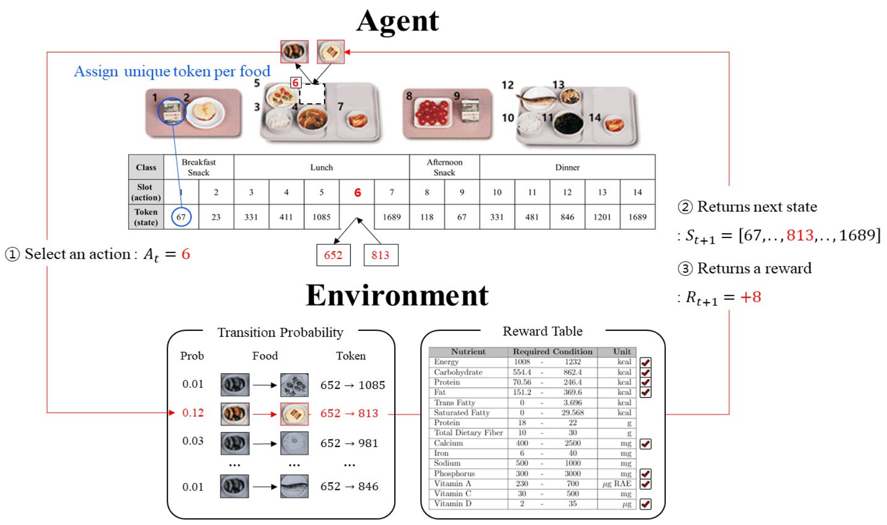
$\mathbb{R}$ is a reward function that maps the given state and action onto a reward, which is the only information that the agent learns from the environment. Equation 1 shows the numerical expression of reward function that is defined as the expected value of the reward. Given that the state is $S_t = s$ and the action is $A_t = a$ at time t, the agent receives a reward according to the reward function.

$$r(s_t, a_t) = \mathbb{E}[R_{t+1}|S_t = s, A_t = a] = R_{t+1} \tag{1}$$

The agent makes a decision (i.e., selects an action) as follows. In an MDP setting, successive decisions are made over time. Therefore, the agent needs to consider the future situation caused by the current decision. To do this, a discount factor γ is introduced. Total return to the agent at time t is determined as follows:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + ... \tag{2}$$

The environment provides not only a reward to the agent, but also a next state, as determined by the transition function $\mathbb{T}$. The transition function defines $P_{s,a}^{s'}$, the probability of states that the environment will return as a next state $S_{t+1} = s'$ when the agent takes action $A_t = a$ at the given state $S_t = s$. Because the environment underlies the transition function, it is called an environment model.



Class	Breakfast Snack		Lunch					Afternoon Snack		Dinner				
Slot (action)	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Token (state)	67	23	331	411	1085		1689	118	67	331	481	846	1201	1689

Nutrient	Required Condition			Unit	
Energy	1008	-	1232	kcal	✓
Carbohydrate	554.4	-	862.4	kcal	✓
Protein	70.56	-	246.4	kcal	✓
Fat	151.2	-	369.6	kcal	✓
Trans Fatty	0	-	3.696	kcal	
Saturated Fatty	0	-	29.568	kcal	
Protein	18	-	22	g	
Total Dietary Fiber	10	-	30	g	
Calcium	400	-	2500	mg	✓
Iron	6	-	40	mg	
Sodium	500	-	1000	mg	
Phosphorus	300	-	3000	mg	✓
Vitamin A	230	-	700	μg RAE	✓
Vitamin C	30	-	500	mg	
Vitamin D	2	-	35	μg	✓

Figure 5: An overview of interaction between agent and environment for diet design

In this work, we developed the RL-based AI solution that produces a daily diet plan (see Figure 5). Given that nutritionists sequentially adjust the foods while designing a diet, we modeled the diet design problem as the MDP. First, we defined the state to have an identical shape to a daily diet, similar to the real diet data in Table 2. The state is as a vector of length 14, in which each element represents a food item in the diet. Each element is filled with a unique token randomly assigned to each food. In short, the state is the vectorized format of a diet.

Second, the agent selects one element (i.e., the agent makes an action) among the 14 elements of diet vector and removes the food in element. Then, the environment returns a new food to fill in the blank element, and the state with the new food becomes the next state. If the nutritional value of the next state is superior to that of the current state, the environment returns a reward to the agent. The nutrition quality of state is calculated by summing the individual nutritional values of each food items in the diet. We set the nutrient requirements of diet design based on the dietary reference intakes for Korean children aged 3-5 years. The number of nutrient requirements achieved at the given state was defined as the reward value. Table 3 shows the list of nutrients and their requirements to be achieved by the RL-based AI.

Table 3: A reward table of nutrients and their required conditions

Nutrient	**Required**	**Condition**		**Unit**
Energy	1008	-	1232	kcal
Carbohydrate	554.4	-	862.4	kcal
Protein	70.56	-	246.4	kcal
Fat	151.2	-	369.6	kcal
Trans Fatty	0	-	3.696	kcal
Saturated Fatty	0	-	29.568	kcal
Protein	18	-	22	g
Total Dietary Fiber	10	-	30	g
Calcium	400	-	2500	mg
Iron	6	-	40	mg
Sodium	500	-	1000	mg
Phosphorus	300	-	3000	mg
Vitamin A	230	-	700	μg RAE
Vitamin C	30	-	500	mg
Vitamin D	2	-	35	μg

Third, the agent is trained via interactions with the environment. The agent repeats the interactions to learn a policy that maximizes the total return. For training, we used the deep Q-learning [7, 8]. The update algorithm of Q-learning (equation 3) aims to approximate the action-value function (equation 4) as accurately as possible.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \tag{3}$$

$$q_\pi \doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a] \tag{4}$$

This interaction is illustrated in Figure 5. If the agent judges arrival at a terminate state, the interaction ends. In this work, the terminate state is achievement of a recommended daily nutrient value that meets all the required conditions in Table 3. At this time, the agent obtains a total return as maximized as possible. Therefore, RL-based AI always provides a nutritionally improved diet for children.

List of Figures

List of Tables

References

[1] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial nets. In: Advances in neural information processing systems; 2014. p. 2672–2680.

[2] Frey BJ, Dueck D. Clustering by passing messages between data points. science. 2007;315(5814):972–976.

[3] Powell WB. Reinforcement learning and stochastic optimization. John Wiley & Sons; 2020.

[4] Kaelbling LP, Littman ML, Moore AW. Reinforcement learning: A survey. Journal of artificial intelligence research. 1996;4:237–285.

[5] Sutton RS, Barto AG, et al. Introduction to reinforcement learning. vol. 135. MIT press Cambridge; 1998.

[6] Puterman ML. Markov decision processes. Handbooks in operations research and management science. 1990;2:331–434.

[7] Watkins CJCH. Learning from Delayed Rewards. King's College. Cambridge, UK; 1989.

[8] Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, et al. Playing atari with deep reinforcement learning. arXiv preprint arXiv:13125602. 2013.