CSE6242: Final Report
Team 8 - NutriFive

1 Introduction

This report describes a new meal planning system. Three fundamental elements are considered in our system. Firstly, it provides versatile plans based on the requirements of users. We consider only those recipes which satisfy two different nature of constraints set by the user: the time constraint for each slot and the presence/absence of a particular ingredient. For example, the user can specify that lunch on Tuesday should take less than 20 minutes and should not include any dairy products. After filtering the recipes based on these constraints, we intelligently extract recipes from the database which maximise the nutritional value of the suggested recipes. The user can tweak these nutritional values, e.g. a user may specify a higher minimum protein intake for the week. Such flexibility makes the service more personalized. Visualization is the other critical element of our system. It can quantify and visualize the nutrition intake (calories, salt, protein, and more), which is especially helpful for those who suffer from medical ailments and need precise tracking. Moreover, the service automatically generates a shopping list based on the weekly recommendation, helping users plan groceries. The successful implementation of this system will significantly assist people living away from home, such as students, who often prioritize academics and extracurricular over their food habits, leading to detrimental effects on their health.

2 Problem Definition

We want to recommend recipes for the week based on users' preferences of nutritional content, time it takes to make the recipe, and ingredients.
More specifically, we have a database of recipes D obtained from [12]. Each recipe with id i is characterised by the time it takes to prepare t_i, the nutritional content $n_i \in R^d$ and a set of ingredients G_i. Here, d is the number of nutritional parameters we plan to track. The week is divided into K slots. We ask the user to input the available time a_k for each slot in the week. We allow the user to filter recipes according to presence and/or absence of ingredients for each slot. Let the filtered set for slot k be denoted by D_k. Also, let $N \in R^d$ denote the optimal nutrition values. By default, we set N based on recommendations in scientific studies [6].
Given these constraints, we want to find a set of recipe ids y_k for $k \in \{1, 2, ..., K\}$ such that:

- $t_{y_k} \leq a_k$ for $k \in \{1, 2, ..., K\}$ [Time constraint]
- $G_{y_k} \subseteq D_k$ for $k \in \{1, 2, ..., K\}$ [Ingredient constraint]
- $||\sum_{k=1}^{k=K} n_{y_k} - N||_2$ is minimised [Maximise nutritional content]

3 Related works

Humans make decisions on their daily diet based on various factors, including time availability, ingredient availability, price, nutrition [15][10], medical chronic disease such as diabetes [2], age [1], and many more. Junk food is an attractive solution that saves time but is detrimental to the consumer's health [4]. On the other hand, balanced meals often require diverse ingredients, which may be impractical for students on a budget (both in terms of time and money). Hence, it is essential to balance the practicality of the suggestions (time and availability of ingredients) with the nutritional value. This necessitates a non-trivial optimization over the space of available

recipes. In order to avoid repeated meals, Aljbawi et al. [3] developed a recommendation system to generate meal recipes based on GPT-2, a language model to text mine the information from recipes. Salvador et al. [13] reported a novel approach to predict possible ingredients from the image of recipes. Brunstrom et al. [5] conducted research on human digestion and came up with a meal size control method. Hong et al. [8] focused on food nutrition and built a NutriSonic web expert system for meal management. Similar work [9] was also conducted by Hsiao, who built SmartDiet to generate healthy meal plans for users. It also provided restaurant suggestions. However, although those previous works are impressive, they are too specialized for only a few people. The impact of our meal planning system gives more leeway to let users determine which factors they care about most and can be widely used by people at all levels.
Current meal planners offer sets of recipes which are balanced (eMeals, Eat This Much, Tastes Better from Scratch). [11] explores a surge of recipe discovery advancements with recent technological advancements. [14] considers user group preferences also as a constraint. [7] uses similar constraints as ours, but it requires a comprehensive knowledge of nutritional requirements. However, current solutions do not consider the time required to cook meals. This makes it unsuitable for students since they face a severe time crunch. In general, healthy meals tend to be cooked, which takes time, as opposed to junk food which can be readily consumed. Hence, our approach is the first to incorporate the time constraint into the optimization framework natively.

4 Proposed Method

4.1 Algorithm

Given a set of recipes D_k for slot $k \in \{1, 2, ..., K\}$, we need to pick a recipe with id $y_k \in S_k$ for slot k such that $\sum_{k=1}^{k=K} n_{y_k} \approx N$ i.e. we meet the nutritional constraints. Formally,

$$y_1, y_2, ..., y_K = \underset{x_1 \in D_1, ..., x_K \in D_K}{\operatorname{argmin}} || \sum_{k=1}^{k=K} n_{x_k} - N||_2 \tag{1}$$

$$Error(y_1, y_2, ..., y_K) = || \sum_{k=1}^{k=K} n_{y_k} - N||_2 \tag{2}$$

To find the best solution, we need to consider all possible $|D_1| \times |D_2| \times ...|D_k|$ combinations of the recipes and find the one whose aggregate nutritional vector is closest to N. We define *closesness* using the simple L2 norm. This approach is not scalable and infeasible for our current dataset.

To tackle this, we propose the following sampling-based approach:

1. For each slot, randomly sample a fixed fraction f_k of recipes from D_k (f_k can be fixed for all k). Call this set D'_k.
   E.g. if $f_1 = 0.1$ and $|D_k| = 1000$, then $|D'_k| = 100$.
2. Find optimal solution for these subsets i.e. try out all possible $D'_1 \times D'_2 \times ... \times D'_K$ combinations.
3. We can repeat this multiple times (say W times) with different subsets and pick the best solution with the least L2 norm.

There are two advantages of the above method over non-sampling strategies which other apps use and run in fixed time:

- We can control the trade-off between error and time by adjusting the fraction f_k; increasing f_k will decrease error but will take more time to compute. Hence, depending on the available compute resources, we can adjust the time it takes to run the algorithm.
- Multiple instances of the algorithm can be run in parallel. More instances decreases the error of the solution. In other words, we can spawn threads carrying out steps 1-3 above, collect all the solutions and report the best. The more threads we run, the lower the final error.

4.2 Visualization

We have implemented a web app that takes in user input, calls the API/algorithm, and returns a view of recipes for the week. This is better than other apps because we return data in a clean, organized manner where users can click on each recipe to find out all of its information.
NutriFive has 4 views: Constraint (Figure 1), Week (Figure 2), Recipe (Figure 3), Shopping List (Figure 4). Each view is decribed in the caption below the corresponding figure.

Figure 1: Constraint view: Users can add ingredients which should be included or excluded from the recipes for each slot. Moreover, the user also specifices an average time t_k for slot k (multiple of 5). To restrict the size of the subset of dataset, we consider all recipes in the 10-minute range $[t_k - 5, t_k + 5]$.

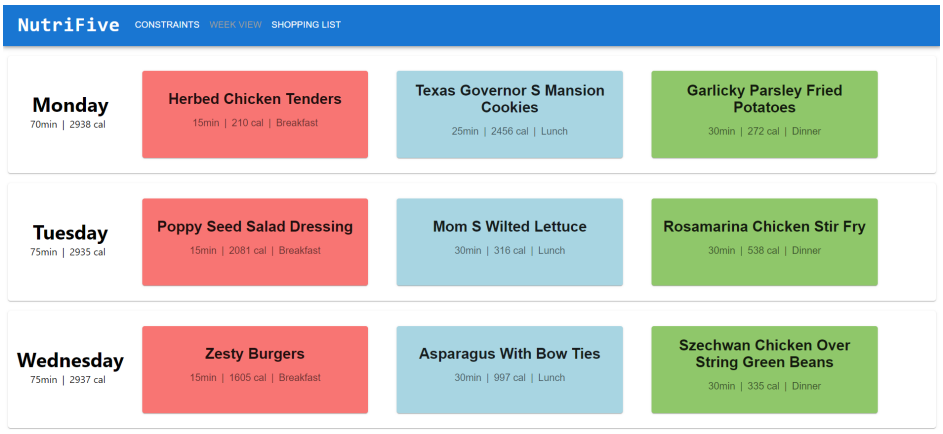

Figure 2: Week view: The suggested recipes are displayed in a grid of cards with 3 columns (3 meals per day) and 7 rows (1 row for each day). Name of the recipe, time taken and calories are displayed as a short summary. Users can click on the card to open up the recipe view.

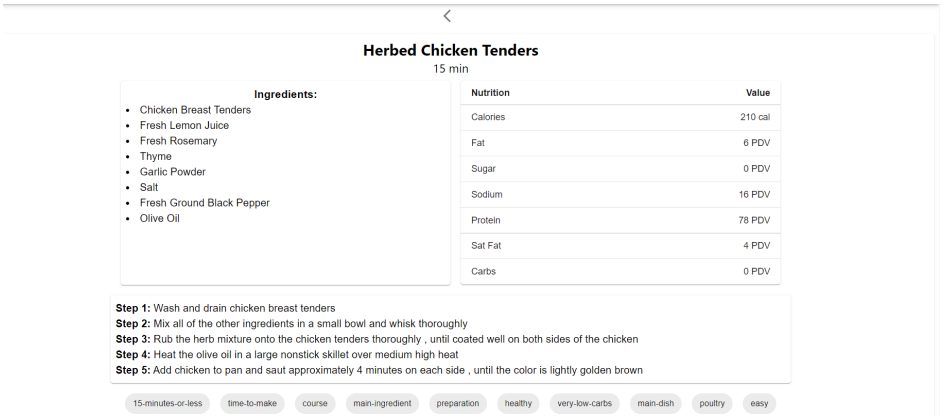

Figure 3: Recipe view: This displays the name, ingredients, nutritional value, steps and tags for the chosen recipe.

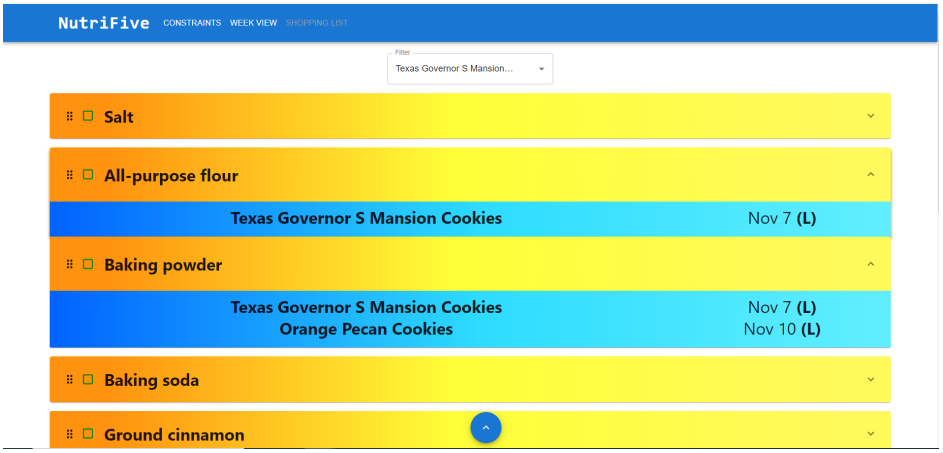

Figure 4: Shopping List view: We aggregate the ingredients of all the recipes suggested by the algorithm and display it as a list. Users can reorder the ingredients, check them off the list or filter ingredients according to recipes.

5 Experiments/Evaluation

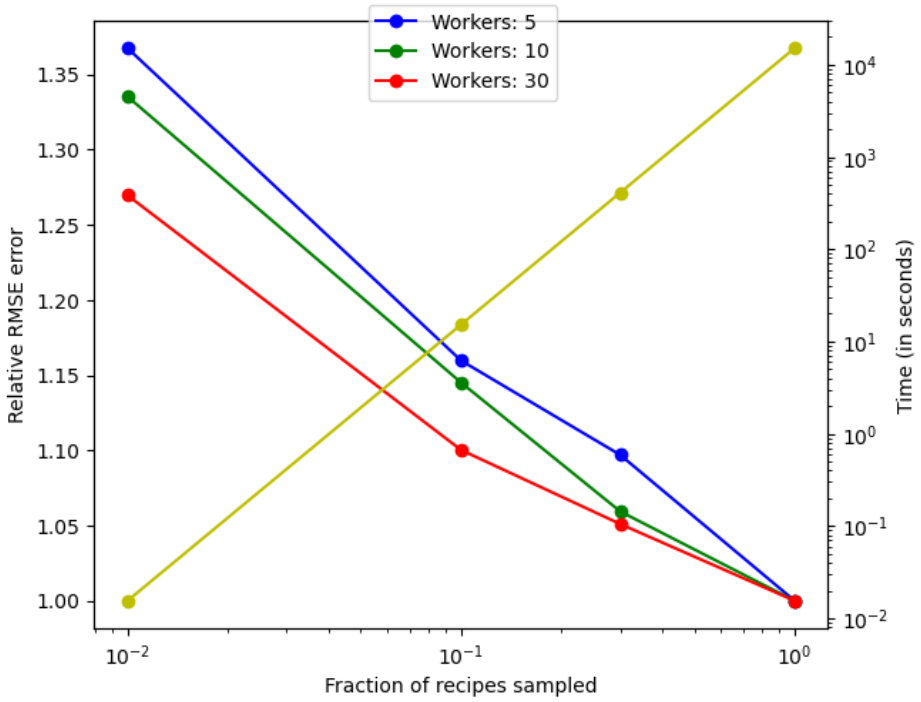

Figure 5: We plot the ratio of L2 error of computed solution to optimal solution for each set of workers. Firstly, as the number of workers increases, the ratio goes down, indicating that more workers decreases the error. This is expected since more workers will capture different random subsets of the dataset and by picking the best solution from the optimal solutions for each subset, we will get a more accurate solution. Secondly, as the fraction f increases, the error goes down. Note that the X-axis is on the log scale. Hence, depending on the computational constraints, we can appropriately choose the value of f to keep the error below a certain constraint. The yellow curve corresponds to time taken by each worker. Note that it's y-axis (on the left) is on log scale. The slope is 3, thereby validating our hypothesis that runtime is $O(n^k)$ where $k = 3$ and n varies linearly with f.

To test the validity of the algorithm, we vary two parameters: the number of workers W and sampling fraction f_k ($= f$ i.e. same for all slots). The target nutritional vector N is the same for all settings. On running the algorithm, we obtain a prediction $y_1, y_2, ..., y_K$ for each of the K slots. The error of this solution is computed using equation 2. It is divided by the L2 error of the optimal solution i.e. the solution when $f_k = 1$ or when all recipes in the database are used. We then plot this ratio as a function of f and W. Results given in figure 5.
Along with the error, we also plot the average time taken by each worker to compute the optimal solution. If k slots has n recipes each, time taken is approximately n^k since we have these many combinations. In our case, $k = 3$ since we find optimal combination for each day (3 meals). n varies linearly with the fraction f.

We distributed a prototype of our app among 10 potential end-users and gathered their feedback on the usefulness of features and ease of use through a survey. The results of the survey are presented

below:

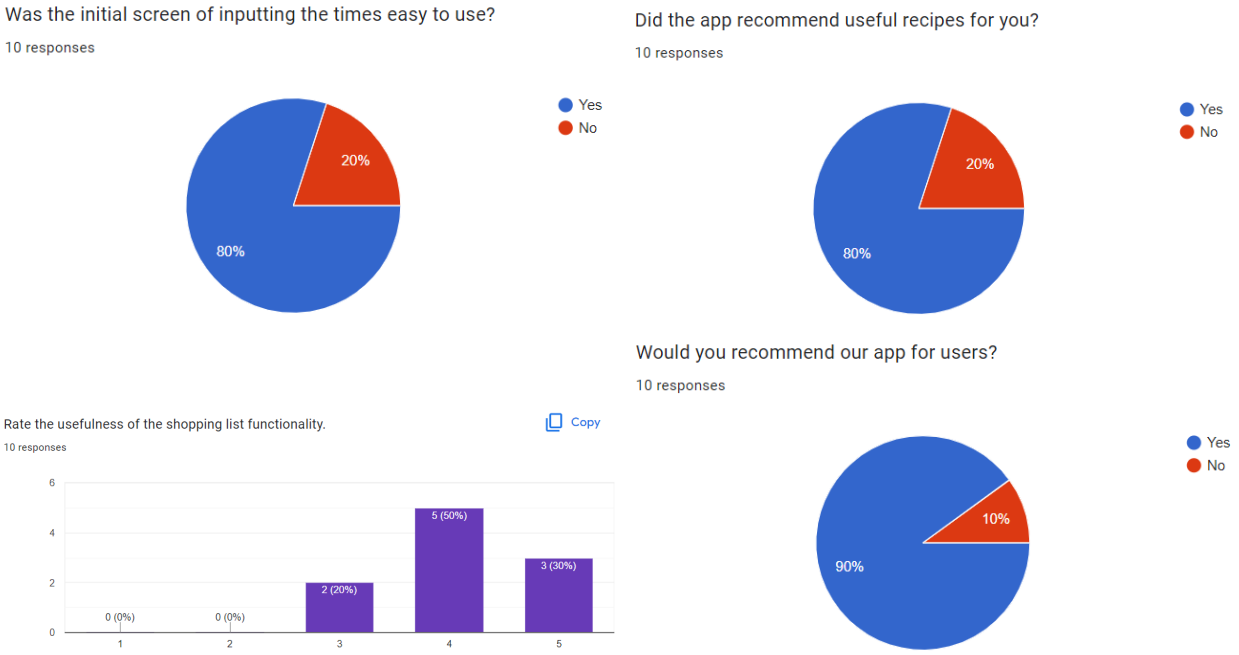

Figure 6: Results from our user survey among 10 students

Hence, the survey results indicate that users were happy with the implementation.

6 Conclusion

In this paper, we introduced a meal planning service which takes into account both time and ingredient constraints set by the user. Drawing from a large database of recipes, the service returns a weekly plan so as to maximise the nutritional value of the suggested recipes. The sampling based approach allows us to control the trade-off between accuracy of suggestions and time to run the algorithm. The majority of our user testers were satisfied with the implementation.
The work can be extended by improving the UI so as to allow the users to move around recipes, add custom recipes to the dataset and track the user's preferences over time so that the service can suggest personal recommendations. Another interesting extension is to incorporate a feasibility score for the recipe based on popularity of the ingredients.
All team members have contributed a similar amount of effort.

References

[1] Johan Aberg. Dealing with malnutrition: A meal planning system for elderly. In *AAAI Spring Symposium: Argumentation for Consumers of Healthcare*, 2006.

[2] Shadi Alian, Juan Li, and Vikram Pandey. A personalized recommendation system to support diabetes self-management for american indians. 6:73041–73051, 2018.

[3] Bushra Aljbawi. Health-aware food planner: A personalized recipe generation approach based on GPT-2. page 89, 2020.

[4] Rajveer Bhaskar. JUNK FOOD: IMPACT ON HEALTH. 2(3), 2012.

[5] J M Brunstrom. Mind over platter: pre-meal planning and the control of meal size in humans. 38:S9–S12, 2014.

[6] Johanna Dwyer. Nutritional requirements of adolescence. *Nutrition reviews*, 39(2):56–72, 1981.

[7] Manuel B. Garcia. Plan-cook-eat: A meal planner app with optimal macronutrient distribution of calories based on personal total daily energy expenditure. In *2019 IEEE 11th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, pages 1–5. IEEE, 2019.

[8] Soon-Myung Hong, Jee-Ye Cho, Jin-Hee Lee, Gon Kim, and Min-Chan Kim. NutriSonic web expert system for meal management and nutrition counseling with nutrient time-series analysis, e-food exchange and easy data transition. 2(2):121, 2008.

[9] Jen-Hao Hsiao and Henry Chang. SmartDiet: A personal diet consultant for healthy meal planning. In *2010 IEEE 23rd International Symposium on Computer-Based Medical Systems (CBMS)*, pages 421–425. IEEE, 2010.

[10] Abdallah Ismail. How many calories should i eat a day? 2018.

[11] Shuyang Li and Julian McAuley. Recipes for success: Data science in the home kitchen. 2020.

[12] Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni, and Julian McAuley. Generating personalized recipes from historical user preferences. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5976–5982, Hong Kong, China, November 2019. Association for Computational Linguistics.

[13] Amaia Salvador, Michal Drozdzal, Xavier Giro-i Nieto, and Adriana Romero. Inverse cooking: Recipe generation from food images. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10445–10454. IEEE, 2019.

[14] Thi Ngoc Trang Tran, Müslüm Atas, Alexander Felfernig, and Martin Stettinger. An overview of recommender systems in the healthy food domain. 50(3):501–526, 2018.

[15] Tsuguya Ueta, Masashi Iwakami, and Takayuki Ito. A recipe recommendation system based on automatic nutrition information extraction. In Hui Xiong and W. B. Lee, editors, *Knowledge Science, Engineering and Management*, volume 7091, pages 79–90. Springer Berlin Heidelberg, 2011. Series Title: Lecture Notes in Computer Science.