1.5em 0pt

Recommending healthy bundles in a food context

Master Thesis

Group: dpw103f18

Aalborg University
Department of Computer Science
Selma Lagerløfs Vej 300
DK-9220 Aalborg

Copyright © Aalborg University 2018

AALBORG UNIVERSITY
STUDENT REPORT

Department of Computer Science
Selma Lagerløfs Vej 300
DK-9220 Aalborg Ø
http://www.cs.aau.dk/

Title:
Recommending healthy bundles in a food context

Theme:
Recommender systems

Project Period:
Spring Semester 2018

Project Group:
dpw103f18

Participant(s):
Mark V. Nielsen
Bogi H. Eliasen
Kristian B. Dyhrberg

Supervisor(s):
Peter Dolog

Copies: 4

Page Numbers: 67

Date of Completion:
June 15, 2018

Abstract:

This thesis looks at how to recommend healthy bundles consisting of dinner recipes. The proposed approach to evaluate the healthiness of recipes is based on the proportions of ingredients. Furthermore, a bundling algorithm from previous work is adjusted to include the developed health measure along with a trained SVD model. Different versions of the algorithm is tested in an offline setting in order to evaluate which version performs best. The accuracy of the SVD model is also compared with a KNN algorithm to determine if KNN is a viable option to SVD. Due to the complexity of bundles in regards to presentation, a user study with 16 participants is also conducted. The user study tests three different layouts with the purpose of finding which is best perceived.

The content of this report is freely available, but publication (with reference) may only be pursued due to agreement with the author.

Preface

This master thesis was written on the 10. semester of Computer Science (IT) at Aalborg university in the spring of 2018.
The project group would like to thank the supervisor of this project, Peter Dolog for his constructive feedback and general help throughout the project.

Aalborg University, June 15, 2018

Summary

Recommender systems usually recommend single items or a list of individual items for a user. In this thesis we delve into bundle recommendation in the context of food. The thesis presents the problem of how to recommend healthy bundles to a user and what should be considered when presenting a bundle to a user. Different approaches have been done to evaluate the healthiness of food recipes. Many of them utilize a scoring function to evaluate how healthy the recipes are and these scoring functions are usually based on already established nutritional guidelines. Most of these scoring functions are based on the nutritional content in recipes, such as calories, fat, etc. and most only consider the healthiness of one meal. Therefore, an assumption can be made that these health scores are not that reliable for indicating a healthy eating habit. In this thesis we present a health scoring function which is not based on nutrition but the proportions of the ingredients themselves. We introduce two versions of our scoring function which is mainly based on the food pyramid in order to determine what is the best proportion of ingredients in a recipe. One version is better suited for calculating a weekly meal plan, whereas the other version is more suited for individual recipes. The scoring function is then adapted to calculate the proportions in a bundle of dinner recipes. By providing bundles with the best split of ingredient proportions to a user, it can be assumed that a user will eat more balanced. The algorithm for creating bundles is based on previous work. In the previous version of the algorithm only aspects such as diversity of a bundle is considered. Diversity is defined as how dissimilar the recipes are in the bundle. We adjust the algorithm to consider the healthiness of a bundle and use a trained SVD model for generating a personalized list of recipes used for bundle creation.

Both an SVD and a KNN algorithm is tested in regards to the implementation. An offline test for both SVD and KNN was therefore conducted in order to determine the accuracy of the two approaches. We also test the bundle algorithm in a offline setting in which we evaluate the performance by its precision, recall, diversity and health. Several different versions of the algorithm is tested, where each version is based on different weighted parameters.

We also look into how bundles can be presented to a user, since the complexity

of presenting bundles is greater than presenting a single item. We look into three different layouts for a bundle, a list, grid and pie layout. These three layouts are looked into as they are also researched in another paper regarding recommendations. We test which layouts are most enjoyable for users by conducting a user study with 16 participants. The participants were exposed to all three layouts and after the test a focus group interview was conducted. The interviews were done in order to both elaborate on the users thoughts regarding the layouts and what is important when recommending food items. The results of the user study gave an indication of what layout was preferred and what aspects should be presented in a recipe item.

Contents

Chapter 1

Introduction

Eating healthy is widely known to be essential to maintaining a fit and healthy physical condition. In precise technical terms, eating healthy means consuming appropriate amounts of both macro- and micro-nutrients [4]. Macro meaning energy contributing nutrients such as fat, carbohydrates and protein. Micro-nutrients on the other hand, represents essential vitamins and minerals [3]. Even though much research has been conducted in this field, in order to specify optimal amounts of the nutrients, these numbers change regularly as new research is conducted. The complexity involved in understanding a healthy diet is also apparent by the many different careers and educations in the field, such as clinical dietitian and nutritional consultants. As such, it is not expected that every person knows and understands everything regarding macro- and micro-nutrients as well as following the recommended intakes. Therefore, in order to reach out to as many people as possible, many different simplified nutritional guidelines have been made. These guidelines have different forms and content. Some use a pyramid or a pie-chart diagram in order to illustrate the optimal proportions regarding food. In Denmark the most well known nutritional guideline is the food pyramid which has existed since 1976 [2]. These simplifications differ from e.g, the World Health Organization (WHO) and the Food Standards Agency's (FSA) nutritional guidelines, in that they do not show the precise optimal amounts of macro and micro-nutrients but instead illustrate different food groups and their optimal proportions. This can be argued to be an easier way to illustrate a healthy diet, especially for home cooked meals were it can be difficult to determine the amount of nutrients. A report conducted from 2011 to 2013 by the Danish National Food Institute shows that there generally is a positive development regarding healthy diets in Denmark. However, still many areas need improvement e.g, only 3% of the Danish population have a satisfactory level of saturated fat, half of the children and 1/3 of adults eat to much sugar and the fiber contents are for the most on a bare minimum of the recommended [27, p. 130-135]. This thesis investigates recommender systems in the context of healthy meals and what considerations should be made for developing such a system.

1.1 Related work

Recommendation algorithms have been implemented within a wide range of systems and domains. Especially shopping and streaming services have benefited from research and progress of machine learning by providing a more personalized user experience [6]. However, in the case of nutritional food recommendations, there have been experiences on how recommendation algorithms affect the users behaviour / recommendations negatively. Due to the nature of recommenders, a negative pattern has been discovered where people who prefer unhealthy recipes are recommended more unhealthy recipes [9]. In regards to this problem and the general problem of overweight, some papers have investigated how to include a health score in food recommender systems.

Healthiness

Elsweiler et. al [12] presents a method for recommending healthy meal plans, under certain constraints. The health score in this paper is measured in relation to a persons recommended daily caloric intake and the caloric distribution of key macro nutrients in the meal plan. This approach can help control weight gain or weight loss but in regards to maintaining an overall healthy diet it does not consider many important nutrients (fibers, vitamins, etc). Another recent approach is to calculate the health score based on dietary guidelines from the World Health Organization (WHO) and the United Kingdom's Food Standards Agency (FSA) [9]. The approach based on WHO uses 7 out the 15 macro-nutrients listed in the guidelines to determine the healthiness of a meal [31]. The approach based on FSA gives recipes a score that only relates to 4 of the foods macro-nutrients (sugar, sodium, fat, saturated fat). This approach for the scores utilizes a so-called traffic light system consisting of the colors green, amber, and red [9]. However, both of these score can be said to be too simple to evaluate a diet in depth. Another aspect of these health scores is that they are based on macro-nutrient values, which can be difficult to understand for the average citizen. This is also recognized by different organizations which have developed more comprehensible models for recommending a healthy and diverse diet. In Denmark the food pyramid is the most well known. This model illustrates optimal proportions for different categories of food in order to more easily illustrate a healthy diet[18]. Out of the aforementioned health guidelines WHO and the danish food pyramid are used to measure a healthy diet over a day, a week or a longer period of time. Besides the FSA score, some work has been done on calculating healthiness of individual meals. The Healthy Meal Index (HMI) has been tested as a way to measure healthiness of meals served to kids [23]. By looking at the HMI Adequacy Score ranging from 0-65, it's possible to score individual meals based on the presence of 14 different foods needed for a healthy diet. The downside of the HMI is however that it does not consider proportions of the meal, which can end in misleading results. Another approach is the healthy eating plate which recommends proportions in a healthy meal [24]. This is however based on American standards, which have very different health guidelines from Denmark. As an example, potatoes are not counted as healthy vegetables in the healthy eating plate. Research has also been done to measure if diverse diets generally results in a healthy diet. Drescher et al. [11] measure the healthiness of a persons food basket by considering the healthiness of individual foods based on the nutritional guidelines from the German Nutrition society(DGE).

Bundle recommendations

Previous work has also been done on how to recommend bundles of food recipes to groups of people. The bundle recommendations were based on solving the problem of selecting dinner for groups of two or more people. Putting recipes together in a bundle, allowed the possibility of recommending weekly dinner plans, in which metrics such as diversity in a bundle was important to not recommend a weekly dinner plan with too similar recipes. Since a system recommending weekly dinner plans with similar recipes would make the users grow tiresome of the specific foods[20].

Interfaces

On another note, analyzing recommenders has also come a long way. From mostly looking at the algorithmic accuracy to viewing the recommender system as a whole [21] [19] [28]. Based on the framework presented by Pu et al. [28] research has been made to see the specific impact different interfaces have on the perception of recommender systems. The framework is for example used by Chen et al. [7] to investigate how three different interfaces in a movie recommender affects users confidence, enjoyability and overall perception of the interface's.

1.2 Problem statement

Based on the area of recommending healthy diets and the different nutritional guidelines presented in the related work, this thesis will focus on how to recommend healthy bundles of food recipes to a user. This is due to how a bundle can be used to represent a meal plan. In order to recommend healthy meal plans however, an appropriate method of determining the healthiness of a meal is required. When considering an implementation of a healthy meal recommender system, aspects such as the complexity of bundles in regards to presentation should also be considered as it could possibly affect the user experience of the system.

The problem statement of this master thesis is as follows:

- How can we recommend healthy meal plans to a user?

This question, leads to the interest of investigating the following sub-questions:

- How can we implement a health function to evaluate the healthiness of a meal plan?
- What considerations should be made in regards to presenting bundles of food items to a user?

These questions, and the knowledge needed to answer them will be investigated in the following sections.

Chapter 2

Background Methods

This chapter describes the different methods used in this project and is structured into three sections. The first section describes the different recommendation approaches used within this project. The second section describes how these recommendation approaches can be evaluated. Whereas the last section describes how to approach user experience in recommender systems.

2.1 Recommendation approaches

2.1.1 SVD

Singular-value decomposition (SVD), which originates from linear algebra, is a matrix factorization technique that factorizes a matrix into three other matrices, see figure 2.1. In the context of recommender systems the original matrix A is a $m \times n$ user/item matrix, where m is the number of user rows and n is the number of item columns. P is a $m \times r$ orthogonal matrix, Σ is a diagonal $r \times r$ matrix whose values are sorted in a decreasing order and Q^T is a n $\times$ r transposed matrix.

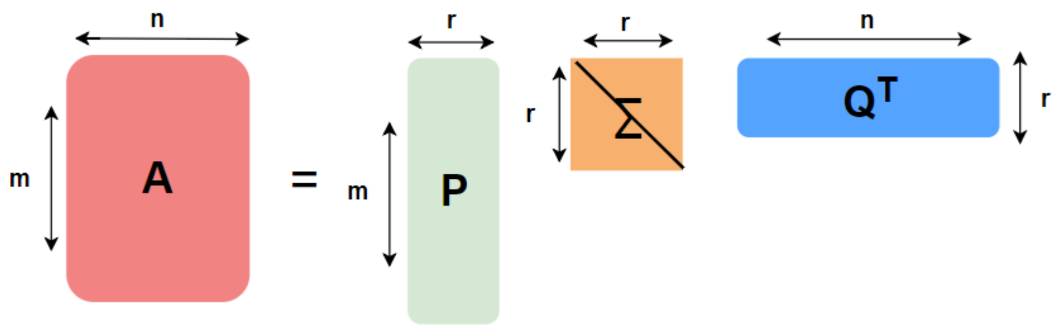


Figure 2.1: SVD - Matrix factorization

In recommendation systems, the SVD of a matrix discovers so-called latent features. These features could for example, in the context of movie recommendation, be thought of as the actors in the movies or its genre. However, it is not possible to know for certain what these latent features are in SVD. The latent features are represented in the matrices P and Q^T, where the values in P indicate how relevant a latent feature is for a particular user, and values in Q^T expresses how relevant the latent features are for a given item. The values in matrix Σ are the weights for each of the latent features, where a greater weight implies greater importance. As mentioned earlier, the values in Σ are sorted so the greatest weights are at the left topmost part of the diagonal matrix[13, p. 83]. An important property of SVD is that it can produce a best approximation matrix with a lower rank than its original matrix. This property is possible due to how the matrix Σ is sorted. The dimension reduction is done by setting the lowest values in Σ to 0, essentially removing them as they have no real importance due to their low values. Removing low values in Σ also results in a simplification of matrix P and Q^T. This is known as low-rank approximation, in which SVD has the property to always produce the best-rank *k* matrix of the original matrix, where *k* in this sense also indicates the cutoff in matrix Σ. See figure 2.2 which illustrates the approximation. Matrix B is in this case a matrix with rank k and is the best approximation to the original matrix A from figure 2.1 which had rank r [14].

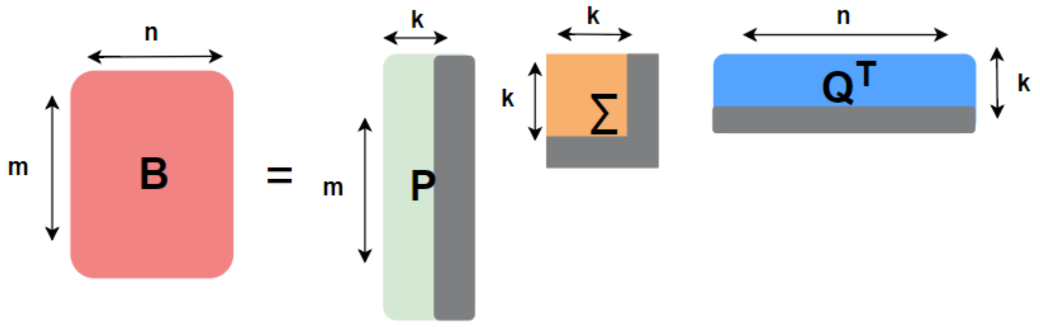


Figure 2.2: SVD - Low rank approximation

SVD in a recommendation context

One of the drawbacks of SVD is that the original matrix has to be dense, i.e. it must not miss any values, in order to compute SVD on it. This drawback is especially important when thinking of using SVD in the context of item recommendation, since user/item matrixes in this context are often sparse, as users usually never rate all of the items that are available. Therefore in this context, SVD can never be computed in the traditional sense. Different approaches have been used to overcome this obstacle, such as replacing the missing values in the matrix with zeros, this however skews the perception of how the user rates items due to zero implying a strong dislike for an item. Another approach, which is based on the best-rank k approximation property of SVD, states that the matrix which only contains the users and the items they have rated, can be used to find the best rank approximation of the matrix containing all users and all items. This reduced matrix still has missing values as users often don't rate the same items, hence SVD can't be computed traditionally. Instead of computing the SVD, this approach utilizes learning to find the latent features in matrices P and Q^T for the known rating values in the approximation matrix. Learning is possible, since users and items can be represented as vectors q_i and q_u respectively. The dot product between the two gives a predicted rating, see equation 2.1, which can be held up against the known rating value and the error can be calculated. As for the missing values in the matrix, they are disregarded, as it is not possible to measure the error between a missing value and a predicted rating.

$$\hat{r}_{ui} = {q_i}^{T} \cdot p_u \tag{2.1}$$

In order to determine how far off the predicted rating is compared to the actual rating, an error is computed, see equation 2.2. An optimization method is then used for minimizing the error between the predicted and actual rating. The SVD algorithm used in this project utilizes the Stochastic Gradient Descent (SGD) optimization procedure for minimizing the error[33].

$$e_{ui} \overset{\text{def}}{=} r_{ui} - {q_i}^{T} \cdot p_u \tag{2.2}$$

Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is as mentioned an optimization procedure, where optimization is the process of finding parameters that minimize a cost function. SGD differs from Gradient Descent (GD) in the sense that SGD selects a single random data point in the training set to approximate a gradient of the cost function, whereas GD uses the whole training set to get an exact gradient[30, p. 185]. Another difference between the two approaches is when a step is taken, namely when the update of the parameters is performed. In GD the update is performed after an epoch, i.e. a pass over the training set. However in SGD, a more iterative approach is used by performing the value update after each training sample. When applying SGD in SVD, it is used for finding the feature values p_u and q_i that minimize the sum of squared error function 2.3. In equation 2.3, k is used to denote the training set containing all the known rating values for each user/item pair.

$$min(p,q) \sum_{(u,i)\in k} (r_{ui} - {q_i}^{T} \cdot p_u)^2 \tag{2.3}$$

In order to avoid overfitting and ensure a more general model, a regularization term is introduced in the cost function, see equation 2.4. The regularization term penalizes the magnitude of feature values. The regularization parameter Lambda(λ) controls the extent of the regularization [13, p. 68].

$$min(p,q) \sum_{(u,i)\in k} (r_{ui} - {q_i}^{\mathrm{T}} \cdot p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2) \tag{2.4}$$

As mentioned earlier, SGD performs updates on the parameters in order to minimize the sum of squared errors, and in the case for SVD there are two update rules. An update rule is used for the user-feature value $\mathrm{p_u}$ and the item-feature value $\mathrm{q_i}$, see equations 2.5 and 2.6. When an update is performed, it is done simultaneously for both $\mathrm{q_i}$ and $\mathrm{p_u}$.

$$q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i) \tag{2.5}$$

$$p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u) \tag{2.6}$$

Gamma (γ) denotes the learning rate, i.e. how big the steps are when moving towards the minimum. The specified value of the learning rate is a balancing act. A too low learning rate would increase the number of iterations required to get to the best values, and a too high learning rate opens up for the possibility of overshooting the best value. A consequence of a high learning rate is therefore the possibility of never being able to converge. λ is the regularization parameter which is used for penalizing large values as they can have a too great contribution to the predicted rating. Therefore with a regularization parameter, the update value is decreased depending on how large the current feature value is[13, p. 84].

Illustration of SVD

This section will illustrate each step performed in learning a SVD model for predicting ratings for a user. At first the values of the learning rate, epochs, etc. have to be specified. The algorithm then takes a rating matrix as input, consider for example table 2.1.

	User/Item matrix			
	Pizza	Chicken Salad	Fish Soup	Beef Stew
User-1	4	2		3
User-2			1	2
User-3	3	3	2	3
User-4	2	4	4	

Table 2.1: User/item rating matrix

First the matrices P and Q^T are initialized with arbitrary values, for the sake of this illustration, the value 0.1 is used for initialization, see table 2.2 and 2.3.

	Matrix P		
	Feature$_1$	Feature$_2$	Feature$_3$
User-1	0.1	0.1	0.1
User-2	0.1	0.1	0.1
User-3	0.1	0.1	0.1
User-4	0.1	0.1	0.1

Table 2.2: Matrix P initialized with the value 0.1

	Matrix Q^T			
	Pizza	Chicken Salad	Fish Soup	Beef Stew
Feature$_1$	0.1	0.1	0.1	0.1
Feature$_2$	0.1	0.1	0.1	0.1
Feature$_3$	0.1	0.1	0.1	0.1

Table 2.3: Matrix Q^T initialized with the value 0.1

When P and Q^T have been initialized, SGD can be used to learn the best feature values in both of the matrices. So for a given number of epochs, and for each known r_{ui} in the user/item matrix, the derivatives of the error function is calculated. Equation 2.7 shows the partial derivative in respect to the item and 2.8 in regards to the user.

$$\frac{\partial}{\partial q_{\mathrm{i}}} {e_{ui}}^2 = -2 \cdot e_{ui} \cdot p_u \tag{2.7}$$

$$\frac{\partial}{\partial p_{\mathrm{u}}} {e_{ui}}^2 = -2 \cdot e_{ui} \cdot q_i \tag{2.8}$$

When the derivatives have been computed, they are then used to update the values of p_u and q_i, see equation 2.9 and 2.10.

$$\begin{aligned} &q_i - \gamma \cdot (-2 \cdot e_{ui} \cdot p_u) = q_i + \gamma \cdot (e_{ui} \cdot p_u) \\ &q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u) \end{aligned} \tag{2.9}$$

$$\begin{aligned} &p_u - \gamma \cdot (-2 \cdot e_{ui} \cdot q_i) = p_u + \gamma \cdot (e_{ui} \cdot q_i) \\ &p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i) \end{aligned} \tag{2.10}$$

An example of updating feature$_1$ in the first iteration is illustrated below. The learning rate is set to 0.05 in the example. First a rating prediction is done for a user, for this particular example a rating prediction is done for the item *Pizza* by *user-3*. After the prediction, the partial derivatives of the squared error is computed followed by a simultaneous update to the vectors $\mathrm{p_u}$ and $\mathrm{q_i}$. The vectors $\mathrm{p_u}$ and $\mathrm{q_i}$ are updated with the values 0.1299 in the first iteration.

$$e_{ui} = 3 - \begin{bmatrix}0.1\\0.1\\0.1\end{bmatrix} \cdot \begin{bmatrix}0.1\\0.1\\0.1\end{bmatrix} = 2.97$$

$$\frac{\partial}{\partial q_{\mathrm{i}}} {e_{ui}}^2 = -2 \cdot 2.97 \cdot \begin{bmatrix}0.1\\0.1\\0.1\end{bmatrix}$$

$$\frac{\partial}{\partial p_{\mathrm{u}}} {e_{ui}}^2 = -2 \cdot 2.97 \cdot \begin{bmatrix}0.1\\0.1\\0.1\end{bmatrix}$$

$$q_i \leftarrow \begin{bmatrix}0.1\\0.1\\0.1\end{bmatrix} + 0.1 \cdot (2.97 \cdot \begin{bmatrix}0.1\\0.1\\0.1\end{bmatrix}) = \begin{bmatrix}0.1\\0.1\\0.1\end{bmatrix} + \begin{bmatrix}0.0297\\0.0297\\0.0297\end{bmatrix} = \begin{bmatrix}0.1297\\0.1297\\0.1297\end{bmatrix}$$

$$p_u \leftarrow \begin{bmatrix}0.1\\0.1\\0.1\end{bmatrix} + 0.1 \cdot (2.97 \cdot \begin{bmatrix}0.1\\0.1\\0.1\end{bmatrix}) = \begin{bmatrix}0.1\\0.1\\0.1\end{bmatrix} + \begin{bmatrix}0.0297\\0.0297\\0.0297\end{bmatrix} = \begin{bmatrix}0.1297\\0.1297\\0.1297\end{bmatrix}$$

When the algorithm is done learning all the feature values, i.e. it has run the specified amount of epochs, a prediction can be made for the missing entries. Since SVD is computationally a expensive procedure, in the sense that the model has to be recomputed when new data is available, another approach to predicting ratings is considered.

2.1.2 KNN

In this thesis a K-Nearest-Neighbours (KNN) algorithm is also used for predicting item ratings for a user. The algorithm is based on user-user collaborative filtering. Intuitively, the algorithm finds the k most similar users for a given user and then computes a predicted rating. A similarity function is used to find the similarity between user u and the rest of the users in the dataset. Each similarity value is stored and the algorithm then selects the k most similar users to u for predicting the rating value. A benefit of this algorithm is that it does not need to train a model in order to do predictions. However, a drawback to this approach is a high consumption of memory, relative to the dataset. Equation 2.11 is used for predicting a rating. Here *sim* denotes a similarity function, u is the selected user, and v is a user in the neighbourhood of u. The prediction function simply performs a summation of all the similarity values multiplied with a rating value for all the users in the neighbourhood, and finally averages over the similarity values. Using a similarity function, ensures that rating values from users that are more similar to u are given a greater weight in the rating prediction [17].

$$\hat{r_{ui}} = \frac{\sum\limits_{v \in N_i^k(u)} sim(u,v) \cdot r_{vi}}{\sum\limits_{v \in N_i^k(u)} sim(u,v)} \tag{2.11}$$

The algorithm is flexible in the sense that it can use different similarity functions, but in this thesis the algorithm is set up to use the *Cosine Similarity*, see equation 2.12.

$$cosine\_sim(u,v) = \frac{\sum\limits_{i \in I_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum\limits_{i \in I_{uv}} r_{ui}^2} \cdot \sqrt{\sum\limits_{i \in I_{uv}} r_{vi}^2}} \tag{2.12}$$

In cosine similarity the two users u and v are considered as two rating vectors ($\mathrm{r_u}$, $\mathrm{r_v}$) and the cosine angle of them depicts how similar they are. The cosine similarity output of 1 indicates the vectors are similar, i.e. they are pointing in the same direction, whereas closer to 0 indicates dissimilarity, see figure 2.3. Cosine similarity sums the multiplications of all the ratings of user u and v, where it treats the missing values as 0. It then divides with the square root of each users squared ratings multiplied. Treating the missing values as 0 ensures that only ratings users have in common are considered.

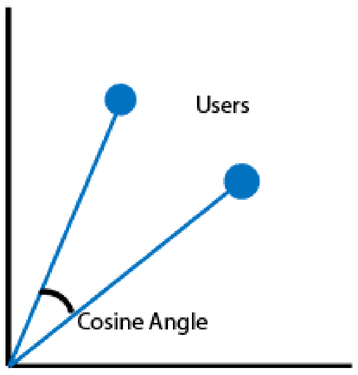


Figure 2.3: Cosine Angle of two vectors

Illustration of KNN

The section will briefly illustrate how KNN predicts an item rating for a user. The KNN illustration uses the same user-rating table as in the SVD example, see table 2.1. As the table is fairly small, the *k* for the algorithm is set to two. Meaning the neighbourhood of the selected user consists of 2 similar users, of which a rating prediction is made. This illustration will show the steps to predict a rating for the item *Fish Soup* for *user-1*. The first step of the algorithm, is to go over the dataset and compute the similarities in order to find the 2 most similar users to *user-1*. The similarities are calculated for each user using cosine similarity. An example calculation of cosine similarity between *user-1* and *user-3* is shown in equation 2.13.

$$cos\_sim(user1, user3) = \frac{(4 \cdot 3) + (2 \cdot 3) + (0 \cdot 2) + (3 \cdot 3)}{\sqrt{4^2 + 2^2 + 0^2 + 3^2} \cdot \sqrt{3^2 + 3^2 + 2^2 + 3^2}} = \frac{27}{29.9826} = 0.9005 \tag{2.13}$$

	User-2	User-3	User-4
User-1	0.4982	0.9005	0.4951

Table 2.4: Similarities computed for user-1

The rest of the computed similarities for the illustration is listed in table 2.4. Looking at table 2.4 the 2 most similar users to *user-1* are *user-2* and *user-3*. When the neighbourhood of *user-1* has been found, a rating prediction can be made by using equation 2.11. Equation 2.14 illustrates how the predicted rating for *Fish Soup* is calculated.

$$\hat{r}_{ui} = \frac{(0.4982 \cdot 1) + (0.9005 \cdot 2)}{(0.4982 + 0.9005)} = \frac{2.2992}{1.3987} = 1.64 \tag{2.14}$$

The final result of predicting the rating for *Fish Soup* by *user-1* using the KNN algorithm with *k* specified as 2, results in a predicted rating of 1.64.

2.1.3 Bundle Recommendation

This section will describe the approach for generating bundles of items, in which a bundle is defined as a collection of items. The approach to bundling food recipe items is an algorithm from previous work [20].

Bundle Algorithm

The algorithm uses two functions for generating a specified amount of bundles. The main function is *Bundle One-By-One* (BOBO), see algorithm 1, which controls how many bundles are to be made. BOBO then calls the second function

Pick_Bundle, see algorithm 2, which is used for generating one bundle at a time. The algorithm in terms of functionality can be summarized into three parts:

1. Generating a list of items which the bundle algorithm uses as a starting point. This part sorts the list of all available items based on popularity, meaning that the items first in the list are the most popular.

2. The process of selecting items to be added to a single bundle, using the function Pick_Bundle(). Here several scores are calculated for adding an item to the bundle. For example a score regarding the diversity of the bundle for adding that specific item is calculated. The item which gets the highest total score is selected and added to the bundle. This process continuously adds items until the size constraint of the bundle has been met.

3. Continuously generating multiple bundles until the number specified has been met, using the algorithm BOBO. This part also performs the action of removing items from the list of all items, so that the algorithm does not duplicate bundles.

Algorithm 1 Bundle One-By-One

```
Input: $I$, $C_{\text{size}}$, number of bundles $k$
Output: a set $k$ of bundles

$Bundles \leftarrow \emptyset$
$Pivots \leftarrow DescendingSort(I, opop)$
while $|Bundles| < k$ do
    $w \leftarrow Pivots[0]$
    $Pivots \leftarrow Pivots - \{w\}$
    $B \leftarrow PickBundle(w, I, C_{\text{size}})$
    $Pivots \leftarrow Pivots - B$
    $Bundles \leftarrow Bundles \cup B$
end
return $Bundles$
```

Algorithm 2 Pick_Bundle

```
Input: $w$, $C_{\text{size}}$, $I$
Output: a bundle $B$

$B \leftarrow \{w\}$
$activeList \leftarrow I - \{w\}$
while $B.Count < C_{size}$ do
    $i \leftarrow argmax_{\text{i} \in \text{active}} Score_{\text{u}}(B \cup \{i\})$
    $B \leftarrow B \cup \{i\}$
    $activeList \leftarrow active - i$
end
return $B$
```

Score metrics

In order to select which item is added to a bundle, several different score metrics are calculated for each case of adding an item. These scores are summed up to a total score and the item giving the bundle the highest total score is added. The different score metrics used are: Diversity, Popularity and Estimated Appreciation. In the subsections below, different equations are used to illustrated the intuition of the various functions. The notation used in the equations is listed in table 2.5.

I	Refers to a given set of items
i	Refers to an item $i \in I$
j	Refers to an item $j \in I$
P	Refers to a package/bundle P, where $P \subset I$
S	Refers to a set of items
u	Refers to a user

Table 2.5: Notation used in equations for bundle score metrics

Diversity

The purpose of this particular score is to determine how diverse the bundle of items are. An item which is not similar to the items already in a bundle would get a higher score and thus a higher chance of being added to the bundle. The function Intra-Package Diversity (IPD) is used for calculating the diversity of the bundle, see equation 2.15.

$$IPD(P) = \frac{\sum_{i,j \in P} 1 - sim(i,j)}{|P|^2} \tag{2.15}$$

i and j denote two items and P is a particular bundle. IPD is a simple function that calculates the pairwise distance between the items in the bundle and then averages with the number of items in the bundle. The similarity (sim) measurement currently used in the algorithm is Jaccard similarity, which considers each recipe as a set and the members of the set as ingredients. Jaccard similarity is formally defined as the intersection between two sets divided by the union of them, see equation 2.16. A threshold value τ is also specified, in which the value is set to 0.5. This means that if the output of jaccard similarity function is less than 0.5, then the two sets are not considered similar.

$$J(S_\mathrm{i}, S_\mathrm{j}) = \frac{|S_\mathrm{i} \cap S_\mathrm{j}|}{|S_\mathrm{i} \cup S_\mathrm{j}|} \tag{2.16}$$

Popularity

The popularity of an item is based on how often it has been rated by users (pop). The popularity score of a single item is calculated using the Overall Popularity (opop) function, see equation 2.17.

$$opop(i) = \frac{pop(i)}{max_{j \in I} pop(j)} \in [0, 1] \tag{2.17}$$

The opop function is then extended to calculate the overall popularity of the bundle instead of just a single item, see equation 2.18. The function sums up the overall popularity of all the items in the bundle and then takes the average.

$$opop(P) = \frac{\sum_{i \in P} opop(i)}{|P|} \in [0, 1] \tag{2.18}$$

Estimated Appreciation

The estimated appreciation (eapp) is a personalized score, used for calculating how interesting an unrated item is for a user. The function is based on an item-item collaborative filtering approach, see equation 2.19. Eapp is calculated for the selected item i, and j is an item already rated by the user u. Similarities are then computed for item i and the various items j and multiplied with the rating of j given by user u and finally the value is divided by the similarity.

$$eapp_\mathrm{u}(i) = \frac{\sum_{j \in S_i} rating_u(j) \times sim(i, j)}{\sum_{j \in S_i} sim(i, j)} \tag{2.19}$$

Equation 2.19 is also extended to estimate the interest a user would have in the bundle, see equation 2.20.

$$eapp_u(P) = \frac{\sum_{i \in P} eapp_u(i)}{|P|} \in [0, 1] \tag{2.20}$$

Metric weights

As each of the four metrics mentioned above has an influence on which items are added to a bundle, it is possible to introduce a weight for each of the metrics. The weights can therefore be used to specify different types of bundling algorithms. An algorithm focusing heavily on diversity would produce bundles with more diverse items, but might suffer in regards to personalization or popularity. The score function with weights can be seen in equation 2.21, which sums the scores from the four metrics into a total score for a bundle.

$$Score_{\mathrm{u}}(P) = C_{\mathrm{eapp}} \times eapp_{\mathrm{u}}(P) + C_{\mathrm{opop}} \times opop(P) + C_{\mathrm{div}} \times IPD(P) \tag{2.21}$$

The different weights for each metric is represented as C which is a simple coefficient, where a higher coefficient therefore denotes a higher importance for the algorithm.

2.2 Evaluation Metrics

This section will briefly describe the evaluation metrics used in this thesis. The metrics are used for two different cases, one being a recommender system predicting what rating a user u would give an unrated item i. The other is to find the most interesting item for a user, i.e. recommending items that a user would actually use[13, p. 42].

2.2.1 Mean Absolute Error

Mean Absolute Error (MAE) is used for evaluating rating predicting systems. Equation 2.22 illustrates the intuition of MAE. The term $\hat{y}$ denotes the predicted rating by the recommender system and y denotes the actual rating value given by a user. The result given by MAE can then be used to determine how accurate the recommender system is, in terms of rating predictions.

$$MAE = \frac{\sum\limits_{j=1 \in n} |y_j - \hat{y}_j|}{n} \tag{2.22}$$

2.2.2 Precision and Recall

Precision and Recall are two metrics used in the case of recommending useful items to a user. In this case there are four possible outcomes, see table 2.6. These outcomes occur when a system recommends an item and the user either uses the item(true-positive), i.e. its a good item, or the user won't use the item(false-positive), i.e. its a bad item. Two other outcomes are when the recommender system does not recommend a good item(false-negative) and when it does not recommend a bad item (true-negative). The precision of the recommender system is based on how many good items are recommended out of all recommended items, see equation 2.23. Recall on the other hand is based on how many good items were recommended out of all good items available, see equation 2.24.

	Recommended	Not recommended
Used	True-Positive (tp)	False-Negative (fn)
Not used	False-Positive (fp)	True-Negative (tn)

Table 2.6: Possible results of item predictions [13, p. 275]

$$Precision = \frac{\#tp}{\#tp + \#fp} \tag{2.23}$$

$$Recall = \frac{\#tp}{\#tp + \#fn} \tag{2.24}$$

2.3 User experience of recommender systems

Early on, recommender evaluation was typically focused on improving the algorithms in regards to e.g. accuracy and precision. These metrics are indeed important in order for a recommender to be successfull. However, around the year 2005 different articles were published which address the importance of viewing a recommender system as a whole[21]. The view here is that recommenders should be seen in terms of how they support and interact with users. This means that, among other things, the way the user interacts and receives information must be considered along with the users motivation and perception of the system. These are the three key areas identified in the framework designed by McNee et al. where each contains subcategories [21]. More recently, additional frameworks have appeared, which are more focused on the evaluation of recommender systems. Whereas McNee et al.'s framework is meant as a design guide. Among the newer frameworks are Pu et al. [28] and Knijnenburg et al [19]. These two frameworks are somewhat similar in the categories they identify as important in evaluating a recommender. Pu et al. identifies 4 higher level categories as: the perceived system qualities, users' beliefs, their subjective attitudes, and their behavioral intentions [28, p.158]. Knijnenburg et al identifies 6 areas, where most are similar to Pu et al. expanding with the areas of personal characteristics and situational context. In the user study performed in this project it was decided to follow the framework developed by Pu et al. as this framework is used in a paper related to this project[7]. The framework of Pu et al will therefore be further detailed in the following section 2.3.1.

User Perceived Quality	**User Beliefs**	**User Attitudes**	**Behavioral Intentions**
Recommendation Accuracy	Perceived Ease of Use	Overall Satisfaction	Use Intentions
Recommendation Novelty	Control	Confidence & Trust	Purchase Intentions
Recommendation Diversity	Transparency		
Interface Adequacy	Perceived Usefulness		
Explanation			
Information Sufficiency			
Interaction Adequacy			

Table 2.7: ResQue Framework[28]

2.3.1 ResQue framework

Pu et al. has developed the framework: Recommender systems' Quality of user experience(ResQue). This framework is based on existing work within the field of recommender evaluation, usability and user experience in general. The goal is to unify the work within user experience evaluation of recommender systems. The framework identifies four key evaluation areas of recommender systems where each have sub constructs describing them. In total 15 constructs are identified, see table 2.7.

Initially different candidate questions are identified to measure these constructs. These initial questions were reviewed in a pilot study and by experts in the field in order to prune and modify questions. The questions were evaluated in a two month study with 239 participants in order to do a statistical evaluation of the questions reliability and validity. For measuring the reliability of the questions on the same constructs, Cronbach's alpha is used. After several iterations of removing and combining constructs they reach Cronbach's alpha values of $\alpha > 0.5$ for all constructs. Where most achieve values of $\alpha > 0.7$ which indicates acceptable internal reliability. For measuring the validity of the questions, factor loading was used. Here they find a correlation values > 0.5 with the lowest being 0.699 which indicates a strong correlation between the questions and the constructs.
In the end the result is a well evaluated framework presenting 15 constructs and 32 questions for evaluating the important aspects of a recommender system. Additionally, they also present a short version of ResQue. Here they state that instead of applying all 32 questions, 1 from each construct can be used for a short version of the framework[28].

Chapter 3

Implementation

Based on the presented background methods, a bundle recommendation system was implemented. The purpose of the system is to recommend healthy bundles to users based on personal preference. This chapter first describes a web crawler used for expanding the existing dataset. Following this is a description of the implemented health score. Lastly, the implemented bundle and recommendation algorithms are presented. The code for all implementations, except the web application and database can be found in appendix E.

3.1 Ingredient crawler

Due to the scope of this project, a dataset of ingredients was needed to evaluate the healthiness of recipes and thereby bundles of recipes. From a previous implemented crawler, made by the authors of this project [20], a dataset was collected containing both users, ratings and recipes. In order to add nutritional information, a new crawler was constructed and fine tuned to crawl the site Epicurious.com. The scripting language Python was used to write the crawler using regular expressions to match ingredients to categories. Regular expressions are made available in python through a module called *re*. *re* makes it possible to define a set of rules for strings given as input based on the theoretical concepts of regular expression. To match a string such as "1/4 cup tomato" the following regular expression 3.1 is used.

$$matchObj = re.match(r'(.*)cup(.*)', ingredient) \tag{3.1}$$

To initialize the code above, the crawler loops through the HTML code containing ingredients for a given recipe. Each line in the HTML is then given as input to the expression as a variable *ingredient*. Each ingredient element is then compared to multiple regular expressions similar to the one above, but with different weight parameters such as cup, ounce and pound. Once a match is found, which in this case is for the parameter 'cup', the regular expression looks for a left and right side match group of the word 'cup'. This is expressed in python using parenthesis around the regular expression. To gain a match on any combination of characters or integers the signs '.*' is used as the parameters. The dot(.) stands for any character or integer and the star(*) stands for zero of more occurrences. As a result matchObj will be assigned two match groups which in this case will be '1/4' as the first group and the second being 'tomato'. This makes it possible to use the first element as the amount, and the second element as the ingredient name. If an ingredient is not within our saved list or the pattern doesn't match a regular expression, the ingredient will be saved in a separate file for failed matches. This file will then be used to manually update the ingredient list before a new iteration takes place.

With the script capable of collecting strings defined as ingredients in the HTML source code, classification was needed to transform a string into a pre-categorized ingredient. This was achieved with a predefined "accepted ingredient list". The list was based on data from the Danish Technical University (DTU) [32], where common danish ingredients were weighted and named. The list was translated into English and standard ingredient weights were manually defined, such that, e.g. 1 tomato weighs 75 grams. Additionally, different measurements such as cup and tablespoon were translated to grams. It was then possible to make continuous lookups, looping over each recipe and its ingredients. By taking the tomato example, the match groups (1/4) and (tomato) are identified. A cup of tomato is estimated to weigh 200 grams, so 200/4 = 50 grams which is added to the database as the amount of tomato within that particular recipe.

The result of the script was a list of ingredients along with the required amount for each recipe and attached category values based on the location within the food pyramid.

Another approach for categorizing ingredients would be to use an information retrieval algorithm, such as the Porter Stemming algorithm[25]. The algorithm works by indexing words based on predefined rules. The benefit of this algorithm would have been that ingredients such as Chickens, Chicken and Chicken wing, would all have been identified and indexed in a single category; *Chicken*, if the correct rules were in place. The rules are based on words(stems) and affix(suffix). The stems are the index word you wish to keep, whereas the suffix is something following the stem that can be changed or removed. A simple rule for removing the last $'s'$ on any word would be $s \rightarrow \epsilon$ such that $chickens \rightarrow chicken$. However our main problem is not in the indexing of ingredients but in the calculation of ingredient weights. Therefore, the algorithm was not implemented.

3.2 Health score

For developing a health score in this project it was decided to base it on the most well known nutritional guide in Denmark, the food pyramid.

3.2.1 Food pyramid

The Danish food pyramid was first seen in 1976. Since then many revisions have been made, where the latest is from 2011. The food pyramid illustrates how to combine foods in order to achieve a healthy diet[5].

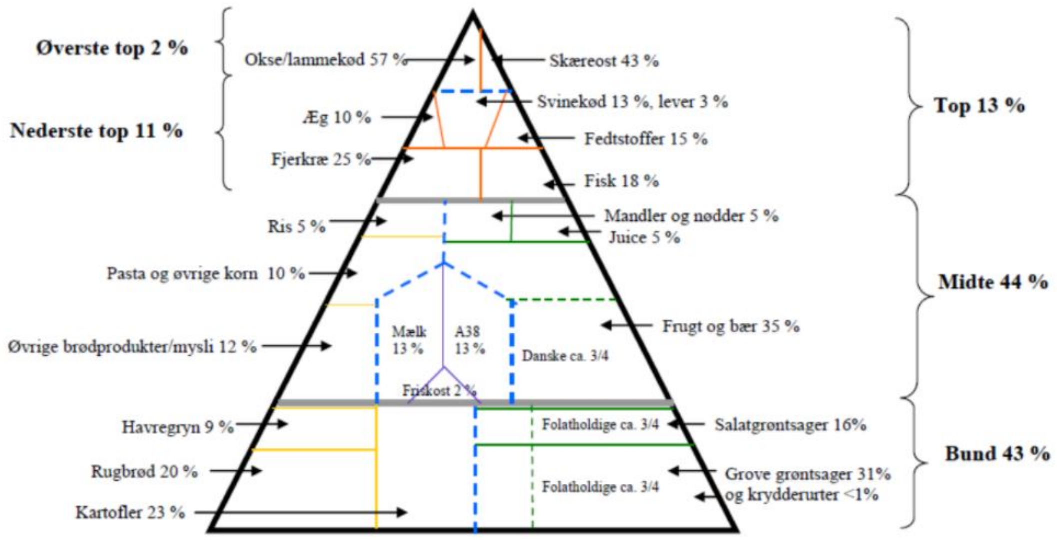


Figure 3.1: Detailed food pyramid [18]

During development of the health score, several question arose of how to interpret the food pyramid to a single score. When looking at the detailed food pyramid, 2 different ways of dividing it can be identified, see figure 3.1. The first one divides the pyramid into 4 sections: "Øverste top 2%" (Upper top), "Nederste top 11%"(Lower top), "Midte 44%" (Middle), and "Bund 43%" (Bottom). The second approach, is to divide the pyramid into the 23 different food categories. Both of these divisions were investigated. Using all 23 food groups for a health score could be useful when calculating a full weeks food plan. However, meals would almost never contain all 23 food groups making this approach unsuitable for individual meals. Therefore, as the scope of this project is to only consider dinner meals, this approach was not be followed. Now, the only thing remaining was that the food pyramids proportion guidelines is based on weekly intake. As such, it was decided to look at how the food pyramid could be adjusted to consider an individual meal. Here the healthy eating plate was found as a solution.

3.2.2 Healthy eating plate

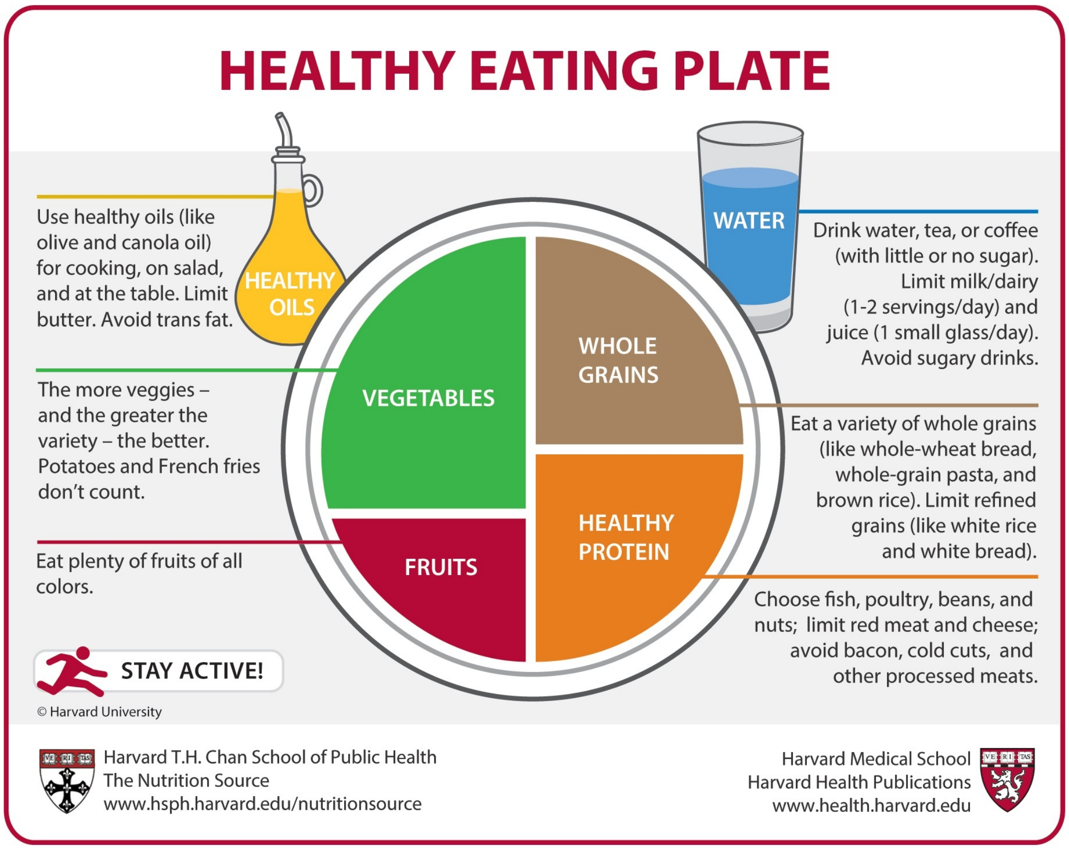

HEALTHY EATING PLATE
Use healthy oils (like olive and canola oil) for cooking, on salad, and at the table. Limit butter. Avoid trans fat.
HEALTHY OILS
WATER
Drink water, tea, or coffee (with little or no sugar). Limit milk/dairy (1-2 servings/day) and juice (1 small glass/day). Avoid sugary drinks.
The more veggies – and the greater the variety – the better. Potatoes and French fries don't count.
VEGETABLES
WHOLE GRAINS
Eat a variety of whole grains (like whole-wheat bread, whole-grain pasta, and brown rice). Limit refined grains (like white rice and white bread).
Eat plenty of fruits of all colors.
FRUITS
HEALTHY PROTEIN
Choose fish, poultry, beans, and nuts; limit red meat and cheese; avoid bacon, cold cuts, and other processed meats.
STAY ACTIVE!
© Harvard University
Harvard T.H. Chan School of Public Health
The Nutrition Source
www.hsph.harvard.edu/nutritionsource
Harvard Medical School
Harvard Health Publications
www.health.harvard.edu

Figure 3.2: Healthy Eating Plate [24]

The healthy eating plate was created at Harvard school of public health, and is meant as a guideline for eating balanced meals, see figure 3.2. The key points of the healthy eating plate are as follows[24]:

- Make most of your meal vegetables and fruits – 1/2 of your plate
- Go for whole grains – 1/4 of your plate
- Protein power – 1/4 of your plate
- Healthy plant oils – in moderation
- Drink water, coffee, or tea
- Stay active

These recommendations for a single meal, can be used to readjust the food pyramid to better represent a single meal. This was done by translating the healthy eating plate to the categories of the food pyramid. The bottom of the food pyramid can be seen as the equivalent to the vegetable section of the healthy eating plate. As such the proportions here are changed from 0.43 to 0.5 as in the healthy plate. The middle of the food pyramid can be seen as the whole grain section of the healthy eating plate, and thus the middle is readjusted from 0.44 to 0.25. The two top sections of the food pyramid then has to account for the last 25% of the healthy eating plate. This comes to readjusted values of top 1 from 0.02 to 0.0385 and top 2 from 0.11 and 0.2115, see table 3.1.

	Original food pyramid	Recalculated food pyramid
Top 1	0.02	0.0385
Top 2	0.11	0.2115
Middle	0.44	0.25
Bottom	0.43	0.5

Table 3.1: Recalculated food pyramid

Now, in order for this re-calibration to still confer with the food pyramid, the remaining meals of the day must then be assumed to contain more from the middle of the pyramid, less from the top and slightly less from the bottom. This can also fit with the danish food culture, where breakfast generally consists of either bread, cereals, milk-products or fruits as found by [10, p.21] and [8, p.23]. These foods are all considered as the middle category. For lunch the biggest food is rye-bread with toppings followed by fruits and vegetables on the side [8, p.25]. These foods belong with the middle and bottom of the pyramid. With a small amount belonging to the top, as the toppings typically are cold cuts of meat. For snacks throughout the day fruits are the biggest category followed by different types of bread and biscuits/cookies [8, p.28-29] which all belong to the middle category.

So in summation, throughout the day most snacks and meals for an average dane are coming from either the middle or bottom of the pyramid with a small section from the top, mostly in the form of cold cuts. The recalculation of the food pyramid in regards to dinner meals makes sense in having increased the amount of meat allowed and the decrease in fruit, breads and dairy products as this is typically covered through the day.
As the recalculated health score is only based on assumptions both the original proportions and the recalculated will be pursued throughout the implementation.

3.2.3 Health score function

With these considerations and assumptions made for the food pyramid, it still needed to be represented by a function in order for it to be usable. This function should take the different proportions of a recipe as input and then output a single

score bounded between 0 and 1 to indicate the healthiness of the recipe.
The health score is implemented by using two different linear functions for each of the four different sections of the food pyramid. Each section uses two functions, one for calculating below the optimal and one for above 3.3. Here is an example shown of the bottom score. The red graph scores proportions below the optimal and the blue scores proportions above. Choosing the correct function is ensured in the implementation. Additionally, the score is also rounded to zero in case of minus values. Pseudo code for the entire health score algorithm is shown in *Algorithm 3* below.

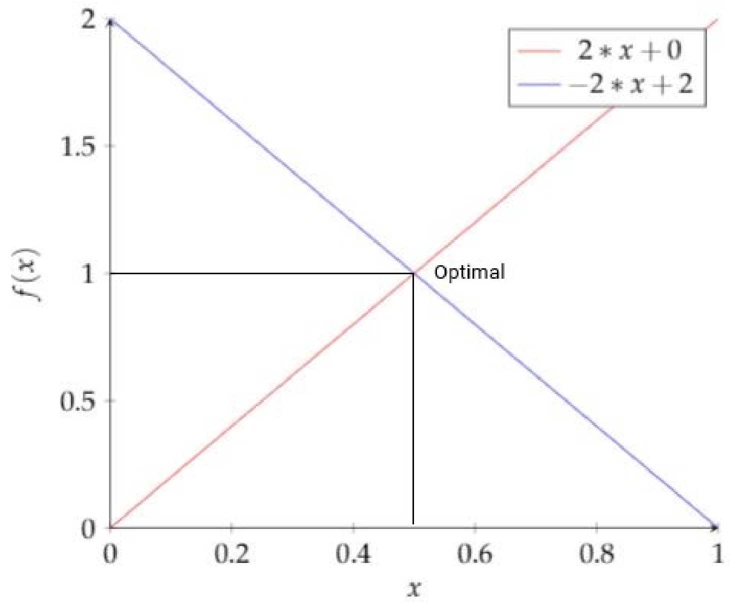


Figure 3.3: Health score graph for bottom proportion where 0.5 is optimal

The algorithm simply works by summing the 4 section scores and taking the mean. The algorithm it self calls 4 other functions to calculate the individual proportion scores. An example of such a function can be seen in *Algorithm 4* below. This algorithm simply checks which function to use for calculating the proportion score. Whether greater or smaller than the optimal value. Lastly it rounds to zero in case of a negative value. The scores for the remaining proportions (middle, top1 and top2) use the same algorithm as the bottom score, *Algorithm 4*, with changed functions to fit the optimal proportions appropriately. See appendix D for all health score functions.

Algorithm 3 CalculateHealthScore

```
Input: $Recipe$
Output: $HealthScore$

$HealthScore \leftarrow (\textbf{CalculateTop1Score}(Top1Proportion) +$
$\textbf{CalculateTop2Score}(Top2Proportion) +$
$\textbf{CalculateMiddleScore}(MiddleProportion) +$
$\textbf{CalculateBottomScore}(BottomProportion))/4;$
return $HealthScore$
```

Algorithm 4 CalculateBottomScore

```
Input: $BottomProportion$
Output: $BottomScore$

if $BottomProportion \geq OptimalProportion$ then
    $BottomScore \leftarrow -2 * BottomProportion + 2$
else
    $BottomScore \leftarrow 2 * BottomProportion$
end
if $BottomScore < 0$ then
    $BottomScore \leftarrow 0$
end
return $BottomScore$
```

3.3 Bundle Recommendation - adjustments

This section will describe what changes have been made to the bundle algorithm mentioned in chapter 4. The main adjustments to the bundle algorithm is that we introduce a health parameter to compute the healthiness of the bundles. Secondly, we use a trained SVD model to provide a list of top rated items for a given user instead of using the most popular items as in the previous version of the algorithm. So instead of using a sorted list of items based on popularity, the algorithm now uses a personalized list of items provided by a SVD recommender.

Health

One of the new adjustments that has been done to the algorithm is the inclusion of a health metric. This health metric is based on the approach in section 3.2.3 for calculating the healthiness of a recipe. This health property is based on the proportions of a recipe instead of its nutritional contents. The optimal proportions are based on the food pyramid. This is implemented by calculating the optimal proportions for a bundle instead of just continuously adding recipes with the best proportions to the bundle. This is due to how a bundle in this project represents a weekly dinner plan, hence it fits well with the recalculated health score as it is based on dinner meals. Since the health property is utilized for the bundle and not just a single recipe, the proportions of a bundle is defined as the sum of proportions of all the recipes in the bundle. The health function used in the bundling algorithm therefore tries to select the recipes whose proportions best fit with the recipes already in the bundle in order to make the proportions of the bundle as close to the optimal as possible, see algorithm 3. A high health score of a bundle thus denotes a bundle whose proportions are close to the optimal. The proportions is as mentioned based on the food pyramid, meaning it is based on the pyramids four parts: Top 1, Top 2, Middle, Bottom. The four proportion parts of a bundle is defined in the equations: 3.2, 3.3, 3.4, 3.5. In the equations P denotes a bundle and *amount* denotes the amount of ingredients.

$$Top1Prop(P) = \frac{\sum_{i \in P} amount \in Top1}{\sum_{i \in P} amount \in Top1, Top2, Middle, Bottom} \tag{3.2}$$

$$Top2Prop(P) = \frac{\sum_{i \in P} amount \in Top2}{\sum_{i \in P} amount \in Top1, Top2, Middle, Bottom} \tag{3.3}$$

$$MiddleProp(P) = \frac{\sum_{i \in P} amount \in Middle}{\sum_{i \in P} amount \in Top1, Top2, Middle, Bottom} \tag{3.4}$$

$$BottomProp(P) = \frac{\sum_{i \in P} amount \in Bottom}{\sum_{i \in P} amount \in Top1, Top2, Middle, Bottom} \tag{3.5}$$

With the proportion parts of a bundle described, a score for each part can be computed. The proportion score computed for each part is using algorithm 3 described in section 3.2.3. Finally when the proportion scores for each part is calculated, a final health score for a bundle is computed. The health score for a bundle is calculated using equation 3.6.

$$health(P) = \frac{Top1Score(P) + Top2Score(P) + MiddleScore(P) + BottomScore(P)}{4} \tag{3.6}$$

With the introduction of a health parameter, the bundle algorithm requires an update to the total score function. Equation 3.7 is the new score function used in the bundle algorithm which considers the healthiness of the bundles.

$$Score_{\mathrm{u}}(P) = C_{\mathrm{eapp}} \times eapp_{\mathrm{u}}(P) + C_{\mathrm{opop}} \times opop(P) + C_{\mathrm{div}} \times IPD(P) + C_{\mathrm{health}} \times health(P) \tag{3.7}$$

3.4 Recommender

The following section covers the design and implementation of the algorithms SVD and KNN used within this project to provide predictive ratings between users and dinner recipes. As an essential part of generating bundles, the rating scores for a given user and all dinner recipes had to be calculated. The reason being, that the bundle algorithm bases its bundle creation around the recipes which received the highest predicted rating for a given user.

In order to decrease the sparsity of the dataset, the design and implementation of the SVD algorithm in this project relies on previous discoveries of how ratings for recipes can be predicted using ingredients [20]. Here it was discovered that it is advantageous to use the fact that a few ingredients can represent many hundreds of recipes, see figure 3.4. Disassembling recipes to ingredients showed a slight decrease in accuracy. However, sparsity was greatly reduced, compared to relying on the recipes alone.

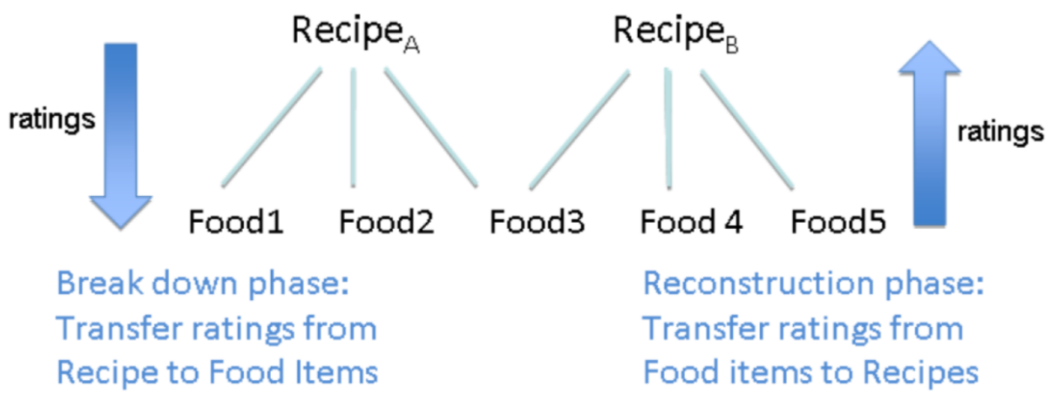


Figure 3.4: Recipe to ingredient model [20]

Users within the dataset had on average rated 2.34 recipes compared to an average of 15.59 ingredients. The total sparsity were therefore reduced from 99.76% down to 89.88% when looking at rated recipes compared to rated ingredients with dinner tag, see table 3.2.

Rated with dinner tag	Ingredients	Recipes
Total amount	154	971
Total Rated Amount	14049	2113
Users	901	901
Sparsity	89.88%	99.76%

Table 3.2: Sparsity of dinner recipes

Based on this knowledge and the pros and cons of this design approach, a function was constructed to handle the disassembling and reassembling of recipes in dataset. Recipes are disassembled to ingredients before they are given to the SVD algorithm as input. The ingredients are then assembled back to recipes when the SVD model has been trained.

The implementation was made using the programming language Python. Python is one of the most popular programming languages, for machine learning, with a lot of frameworks available [29, p. 13]. The SVD and KNN algorithm used within this project is implemented using the Surprise framework by Nicolas Hage[16]. The framework was chosen for the project as it was possible to reuse previous made code for predicting recipe ratings, as this framework was also used within the previous work.

It was however necessary to extend upon previous work and implementations as no online version of the previous implementation was tested. When implementing the SVD algorithm within an online system, where new users emerge and ratings are continuously provided by users, problems such as computation time, updating training data and unknown users arose. To tackle such problems KNN were decided as the algorithm to handle new users who had yet to be part of the trained model. If a new user requests a bundle the API calls to the recommender will automatically switch to KNN in order to locate a similar existing user within the system.

3.5 Web Application

Beside the implementation of the bundle algorithm, healthscore and predictive rating algorithms KNN and SVD, a web application has been designed and deployed. The web application was made in order to combine different aspects of a recommender into a functioning system and thereby be able to further understand the healthy food domain within the context of recommender systems. For the implementation of the web application the Angular 5 framework was used to provide the frontend where users can navigate and interact with the generated food bundles [15].

The Angular 5 framework facilitated rapid development and test of our combined solution as it provides the tools to build a modern and responsive web application using TypeScript and HTML. Besides Angular 5, bootstrap 4 is used, as the main styling reference throughout the application. Bootstrap 4 makes the application look modern with layout styles easy to recognize by users, from other applications around the web [26]. The web application layout can be found in Appendix A. The layout of the application follows a recognizable pattern seen from services like Netflix and Facebook. A menubar is in the top of the browser enabling you to navigate between pages, see A.1. In order to use the system, a user has to create an account and login using the details provided upon creation, see A.2. Once logged in, a new user is presented with a rating page and a random selected recipe, which has to be rated on a scale between 1 and 4 stars. 1 star symbolize a negative preference for the recipe and 4 stars symbolize a strong positive preference for the recipe. The user has to rate 10 random selected recipes in order to navigate to the next and final part of the web application, which is the bundle recommendations. Bundles can within the system be viewed in three different layouts, either as a Pie A.5, List A.3 or Grid A.4. Common for all of them, is that they follow the same 4 star rating system as used when a user first logged in. This is done to provide a certain familiarity in the way bundles are represented as the user now receives much more information compared to individual recipes. The three different layout styles are selected based on the paper [7].

The entire frontend is deployed through Google Firebase, which is part of the Google cloud solution. Firebase Hosting, enabled us to freely host our web application as it provides a deployment package for Angular projects. Besides Firebase Hosting, the application needed support for user profile creation and authentication. To avoid coding this our selves, it was decided to use Firebase Authentication. Through the Firebase Authentication API, our web application is capable of creating and authenticating user profiles and assign them with a unique user ID. The ID is then linked to the database where all data about recipes ratings and existing users are stored.

Besides Google Firebase, services such as Microsoft Azure and pythonEverywhere.com is part of the application architecture as well. This is due to different payment plans on the services, making some services more beneficial than others for certain tasks. The Microsoft Azure service was used to host a server with API's communicating with a database for retrieving and inserting data. The Microsoft Azure server was also used to employ a API using the bundle algorithm in order to provide the front-end with bundles. PythonEverywhere.com was used to host and deploy the entire SVD and KNN recommender implementation. As with the Microsoft Azure API's, the recommender implementation also rely on a REST architecture to ensure clear communication between the different services [22]. An overall application architecture can be seen in figure 3.5.

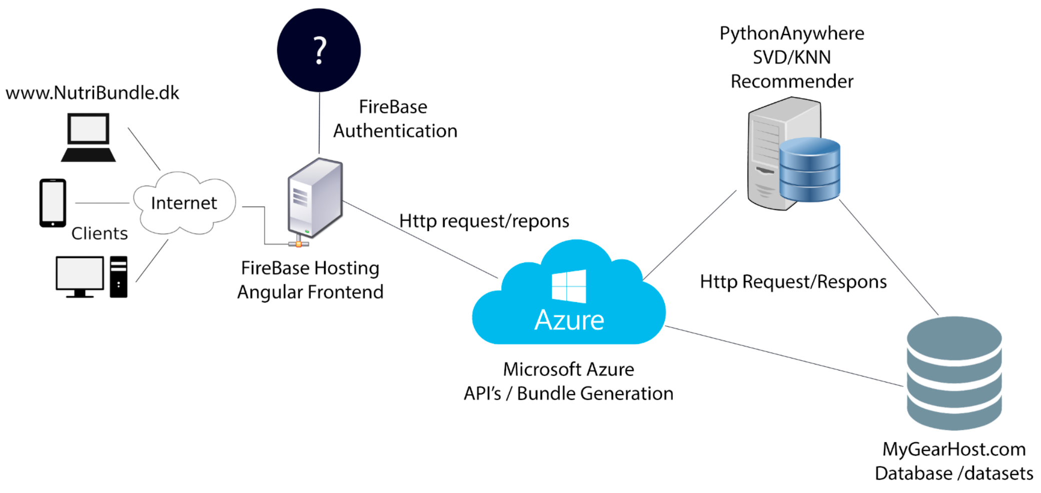


Figure 3.5: Application Architecture

Chapter 4

Evaluation

Besides the implementation of the entire food recommendation system, each part of the bundle recommendation is evaluated individually. This chapter will describe the offline evaluations done on SVD, KNN and the bundle algorithm. Finally, the conducted user study is described. Each part starts with a explanation of how the testing was done followed by the results.

4.1 Offline Testing

4.1.1 SVD/KNN

In order to test the accuracy of the chosen recommendation algorithms SVD and KNN, offline tests was conducted. The tests were conducted using 5-fold cross validation. 5-fold cross validation works by dividing the data into 5 equal sized folds, namely 20% each. Each of these folds are then used as a test fold on a model which is trained on the remaining 4 folds (80%). The benefit of this approach is the ability to use all data for both testing and training to ensure that a given accuracy is not acquired by randomness due to a given fold. Since the model used for our recommendation is based on ingredients and not recipes, the dataset had to be prepared before a 5-fold cross validation could take place. This is due to no explicit ratings being available for the ingredients. It is therefore necessary to compute the folds of 20% test and 80% training data before disassembling recipes to ingredients. There will however be some overlap in ingredients due to several ingredients being part of multiple recipes, see figure 4.1. This should be avoided, as recipes from the training set can influence the ingredient ratings in the test set and thus skew the results of the test. To overcome this obstacle we decided to remove all explicit ingredient ratings from the training set for overlapping ingredients. The predicted recipe ratings is then achieved by only summing up the predicted ingredient ratings and taking the mean.

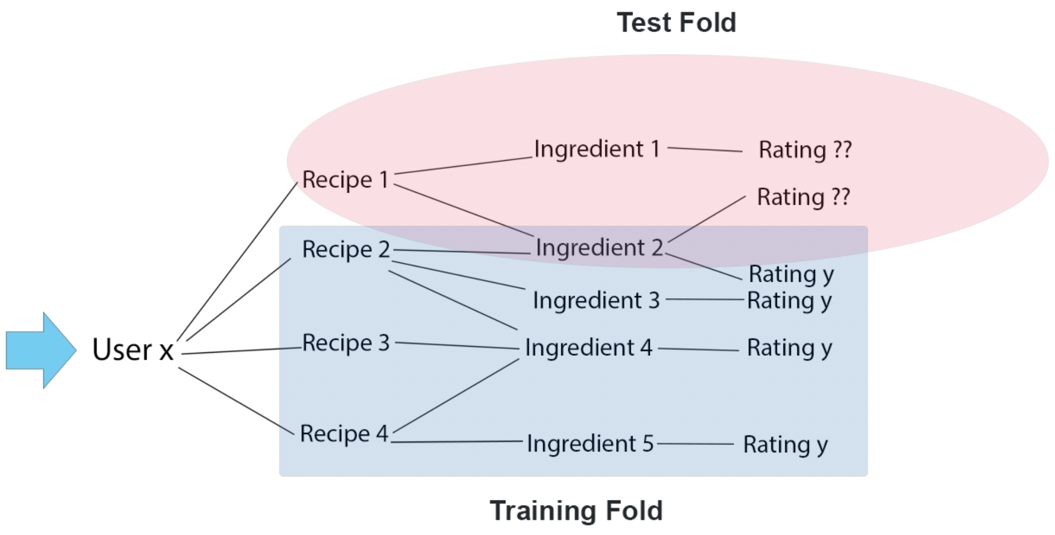


Figure 4.1: Recipe to ingredient fold model

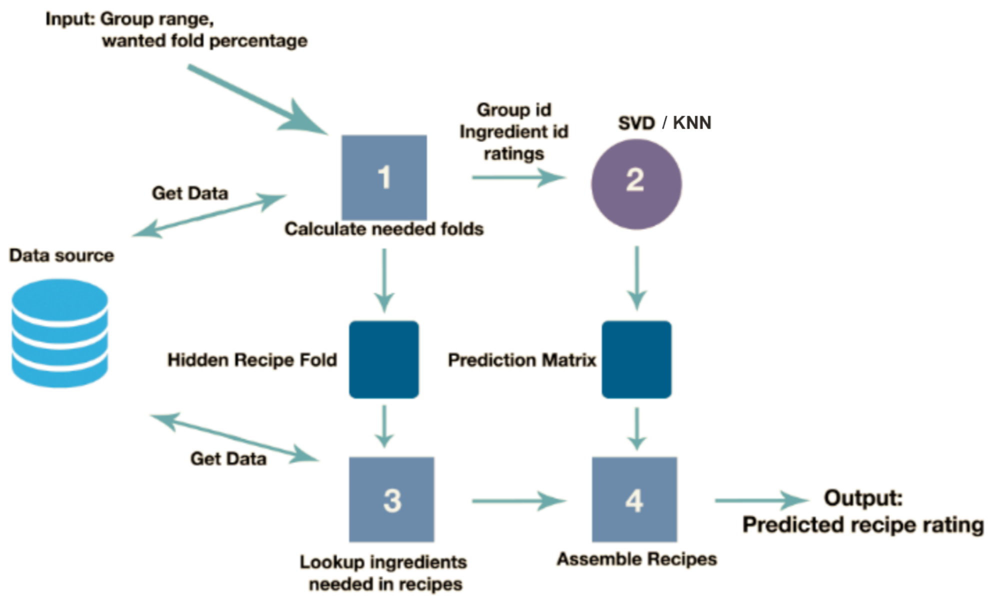


Figure 4.2: SVD offline testing model [20]

The approach to the offline testing can be summarized in four main parts, as illustrated in figure 4.2.

1. First the test fold and training fold is calculated for all users within the dataset to ensure that each user gets a 20% test fold and 80% train fold. Each users data is fetched individually and then divided into folds. Once every users rated recipes are divided, the disassembling into ingredients begin.
2. Ingredients are fetched for the recipes within the training set and given as input to either the SVD or KNN algorithm. The input format is (User ID, Ingredient ID, Rating Value). Based on this, a matrix is constructed and ingredient ratings are predicted.
3. The test fold consisting of rated recipes, is now scanned to identify the ingredients needed within each recipe.
4. Once a list of ingredients for each recipe within the test fold is created, a look up is made in the trained model of user ingredients, which is outputted from the SVD or KNN algorithm in step 2. The recipes are then assembled and the predicted rating for each recipe is calculated.

Once the four steps are completed each predicted recipe rating from the test set is compared to it's original rating given by users.

SVD Offline testing

The parameters of the SVD used within the application and for offline testing are as follows. The epochs, in which SGD is iterated, is set to 20 and 30 to test both values. The learning rate is set to 0.005, whereas the regularization parameter is set to 0.02 and the number of features within the SVD is kept at it's default which is 100.
The evaluation of the SVD algorithm is as mentioned done using 5-fold cross validation. Each fold is then iterated 10 times, to get an MAE for each iteration. The results for each fold and it's iterations can be seen in table 4.1 for 20 epochs. The results for 30 epochs can be seen in appendix C. Based on the two tests, the 20 epochs is found to be superior with an MAE of 0.7238 compared to 0.7524. Even though a promising result for SVD had been found, it was deemed necessary to test KNN as well.

KNN Offline testing

Whereas the SVD algorithm is iterated 10 times for each fold, the evaluation of KNN is only run once for each fold, this is due to no learning function being present in KNN. The results are therefore static. It is however possible to change

SVD - epochs = 20										
20 epochs	test 1	test 2	test 3	test 4	test 5	test 6	test 7	test 8	test 9	test 10
Fold 1	0.6249	0.6289	0.6227	0.6238	0.6242	0.6239	0.6247	0.624	0.6235	0.6236
Fold 2	0.6539	0.6554	0.6532	0.6532	0.653	0.656	0.6543	0.6517	0.6552	0.6535
Fold 3	0.9917	0.9897	0.9883	0.99	0.9935	0.9919	0.9895	0.9881	0.9889	0.9892
Fold 4	0.7208	0.7235	0.7205	0.7191	0.721	0.7203	0.7198	0.7205	0.7213	0.7208
Fold 5	0.6313	0.6301	0.6281	0.6291	0.6305	0.6318	0.6286	0.628	0.6277	0.6296
MAE	0.72452	0.72552	0.72256	0.72304	0.72444	0.72478	0.72338	0.72246	0.72332	0.72334
Average MAE: 0.7238										

Table 4.1: SVD results epochs = 20

KNN(k-max=10, k-min=1) Results				
Fold 1	Fold 2	Fold 3	fold 4	Fold 5
0.6196	0.6193	1.0032	0.7512	0.7263
Average MAE: 0.7438				

Table 4.2: KNN results k-max = 10

the max k and min k parameter for KNN, which defines how many neighbors the algorithm should take into evaluation. KNN will therefore be tested with different k values to see which derives the best possible result.

The first test of KNN was run with a value of k-max = 40, see appendix C, as this is the default value in the Surprise framework. The second evaluation was executed using a smaller k-max value of 10. A last KNN test was done using k-max = 23, see appendix C. The value of k-max = 23 is chosen as it is the square root of the number of users, which is a common default value of K [29, p. 91].

Based on the offline testing of SVD and KNN the MAE comparison between the two algorithms shows that SVD computes the lowest MAE score, indicating best accuracy. The best SVD score was found to be 0,7238 with epochs = 20, see table 4.1, compared to the best KNN MAE of 0.7438, see table 4.2.

4.1.2 Bundle algorithm

In order to measure the performance of the bundle algorithm, an offline experiment was conducted. Since offline testing isn't conducted with real users, an existing dataset containing users, recipes and ratings were used. Different versions of the bundle algorithm are tested, where each version of the algorithm is defined by the weights of the parameters Popularity (Pop), Estimated Appreciation (Eapp), Diversity (Div) and Health. In total 11 versions of the algorithm are tested. Generally there are 3 different categories of weights as shown below.

- Maximum weight for each of the parameters.
- Parameters weighted equally.
- Two parameters given maximum weight e.g. health+diversity is weighted highly.

The performance of the algorithm is also tested based on the returned number of bundles, where the number varies between 5, 10 and 15. The performance metrics used to evaluate the algorithm are precision, recall, diversity and health. Precision measures how the algorithm performs in regards to selecting relevant items in its recommendation. Recall measures how many of the users relevant items are selected in the recommendation [13, p. 283-284]. Diversity is measured as well, since it is considered a good quality in recommendation lists [13, p. 316]. The diversity measures how dissimilar the items are in a bundle. The diversity metric is calculated using *Mean Intra-Package Diversity* (MIPD), see equation 4.1. Here k is the number of bundles recommended to the user. The health metric is used to measure which algorithms recommend the most healthy bundles, based on the recalculated food pyramid health score. The health metric is calculated using *Mean Health* (MH) which is a simple average of the health scores, see equation 4.2.

$$diversity = MIPD(P_1, ..., P_k) = \frac{\sum_{i=1}^{k} IPD(P_i)}{k} \tag{4.1}$$

$$health = MH(P_1, ..., P_k) = \frac{\sum_{i=1}^{k} health(P_i)}{k} \tag{4.2}$$

Since the metrics precision/recall require true ratings to determine relevant / not relevant items and the dataset mainly contains users with 1-3 ratings, a decision is made to only select the most active users. An active user in this case is considered a user who has rated 7 or more recipes, thus 65 users are used for the offline testing. The number of recipes used for the offline test is 404, hence many of the recipes are still unrated by users. An unrated item can possibly be, either a relevant item or irrelevant item. Since the quality of an unrated item can't be determined, it is disregarded in the calculation of precision and recall [13, p. 283]. This means that only items with a true rating (either relevant or irrelevant) is considered in the precision/recall calculations. An item is considered relevant for a user if it has a rating greater or equal to 3, otherwise it is considered irrelevant.

Bundle algorithm - Results

The testing results of all the different bundle algorithm versions are reported in the tables 4.3, 4.4 and 4.5. Each version of the algorithm can be compared to each other according to precision, diversity, recall and health. Across all three of the different *k* (5, 10, 15) the diversity is reported to be relatively high ranging between 62-74% in every version. It is obvious the the version weighting diversity highly performs best in this regard. However, it is interesting to see when weighting other parameters highly that the diversity still remains relatively high. In the case when *k* is 5, the highest drop in diversity is 11.46%, between the versions *Div* and *Health+Pop*. In regards to precision however there are considerable differences between versions. Especially when *k* is 5 several versions report a low precision with a value ranging from 52-58% whereas the best performing version, precision-wise, is *Health+Eapp* at 72.44%, which is roughly a 20% difference. When increasing *k* to 10 and 15, the precision goes up to a range between 76-86%. The best performance in terms of precision is reported when *k* is set to 15 and popularity is weighted highly, see version *Pop* and *Div+Pop*. Another finding in regards to popularity is the level of healthiness. When popularity is weighted highly in the algorithm, the findings report the lowest levels of healthiness across all three *k's*. This result can to some extent confirm the assumption that users go for more unhealthy food, since the popularity of a recipe is defined as the number of times it has been rated. As for recall, the value increases when the number of recommendations is increased. All versions of the algorithm which include the health parameter have the best performance in regards to recall when *k* is 15. The results of the tests give indications that weighting health highly is a viable option. There is expected to be some trade-offs when weighting one metric over another. When weighting health highly, small drops in precision and diversity are expected, but at the benefit of increased healthiness.

	k = 5			
	Precision(%)	Diversity(%)	Recall(%)	Health(%)
Equal	70.86%	67.99%	**27.46%**	53.60%
Div	68.82%	**74.31%**	26.37%	45.23%
Eapp	70.31%	68.03%	**27.46%**	52.34%
Pop	52.30%	69.42%	23.97%	33.30%
Health	69.87%	65.67%	15.39%	**74.79%**
Health+Div	70.39%	67.96%	15.34%	73.37%
Health+Pop	59.32%	62.85%	14.45%	70.38%
Health+Eapp	**72.44%**	62.87%	24.19%	54.95%
Eapp+Pop	61.14%	67.21%	22.87%	38.72%
Eapp+Div	57.83%	73.03%	25.29%	46.39%
Div+Pop	57.94%	70.10%	22.21%	41.08%

Table 4.3: Results when the returned number of bundles is set to 5

	k = 10			
	Precision(%)	Diversity(%)	Recall(%)	Health(%)
Equal	80.13%	67.94%	31.02%	54.30%
Div	**85.11%**	**74.32%**	30.70%	46.80%
Eapp	80.08%	67.96%	**31.03%**	53.90%
Pop	84.89%	69.42%	28.68%	33.02%
Health	76.68%	65.94%	29.12%	**75.90%**
Health+Div	78.61%	68.07%	26.73%	73.94%
Health+Pop	78.35%	68.75%	25.40%	71.58%
Health+Eapp	78.15%	66.41%	25.99%	55.60%
Eapp+Pop	80.84%	67.24%	23.06%	40.76%
Eapp+Div	83.16%	73.34%	25.32%	47.55%
Div+Pop	84.96%	70.14%	22.55%	41.55%

Table 4.4: Results when the returned number of bundles is set to 10

	k = 15			
	Precision(%)	Diversity(%)	Recall(%)	Health(%)
Equal	81.12%	67.79%	42.25%	51.50%
Div	84.92%	**74.25%**	39.29%	44.40%
Eapp	81.07%	67.81%	38.81%	50.81%
Pop	86.62%	69.40%	35.33%	33.20%
Health	78.34%	65.94%	**46.42%**	**75.86%**
Health+Div	78.70%	67.93%	44.44%	72.73%
Health+Pop	79.69%	68.51%	45.04%	51.63%
Health+Eapp	79.91%	66.26%	41.46%	55.96%
Eapp+Pop	82.23%	67.27%	36.13%	38.91%
Eapp+Div	53.97%	73.25%	38.15%	43.20%
Div+Pop	**86.84%**	70.11%	36.34%	40.92%

Table 4.5: Results when the returned number of bundles is set to 15

4.2 Health score

Paired t-tests were conducted for all health scores in order to confirm whether there was a statistical significant difference between the scores. For all paired tests there were found acceptable p-values of $P < 0.05$, see appendix C.1 for p-values.

In this project two versions of a health score have been created. The hypothesis is that the readjusted health score should better reflect what a healthy dinner meal is. Therefore, as the recipes considered in this test are all dinner meals, the readjusted score should reflect that with healthier scores.
Optimally, it would be possible to evaluate in regards of absolute error or squared error, as it would indicate the accuracy of the health scores. However, in order to evaluate accuracy, a true baseline healthiness of the recipes is required. Unfortunately this is not available, and using one of the health scores investigated(WHO/FSA) as a baseline is not possible. The reason being that the health scores are based upon different health guidelines and it cannot be said that one is more correct than the other. However, even though the health scores are based upon different guidelines, looking at the correlation between the health scores can still give an indication of agreement. The intuition is that even though the health scores are different, they should at least share some positive correlation. A strong negative correlation of -1 would indicate that the scores completely disagree on recipes where a positive correlation would suggest agreement between the health scores. The correlation matrix can be seen in table 4.6.

Pearson correlation matrix	WHO	FSA	Health score recalculated	Health score original
WHO	1			
FSA	0,297623	1		
Health score recalculated	-0,004890	0,018723	1	
Health score original	0,002883	0,012235	0,460888	1

Table 4.6: Health score correlations

The correlations show that WHO and FSA have a weak positive correlation. Indicating that there is some general agreement on the recipe health scores. For both the recalculated and the original health scores, compared to WHO and FSA, there is a correlation of approximately 0. There may be many reasons for this, one being that they are based on different health guidelines. However, a positive correlation greater than zero was expected. It was initially though that because the values of WHO and FSA are ordinal, on a scale of 0-6, and the developed health scores

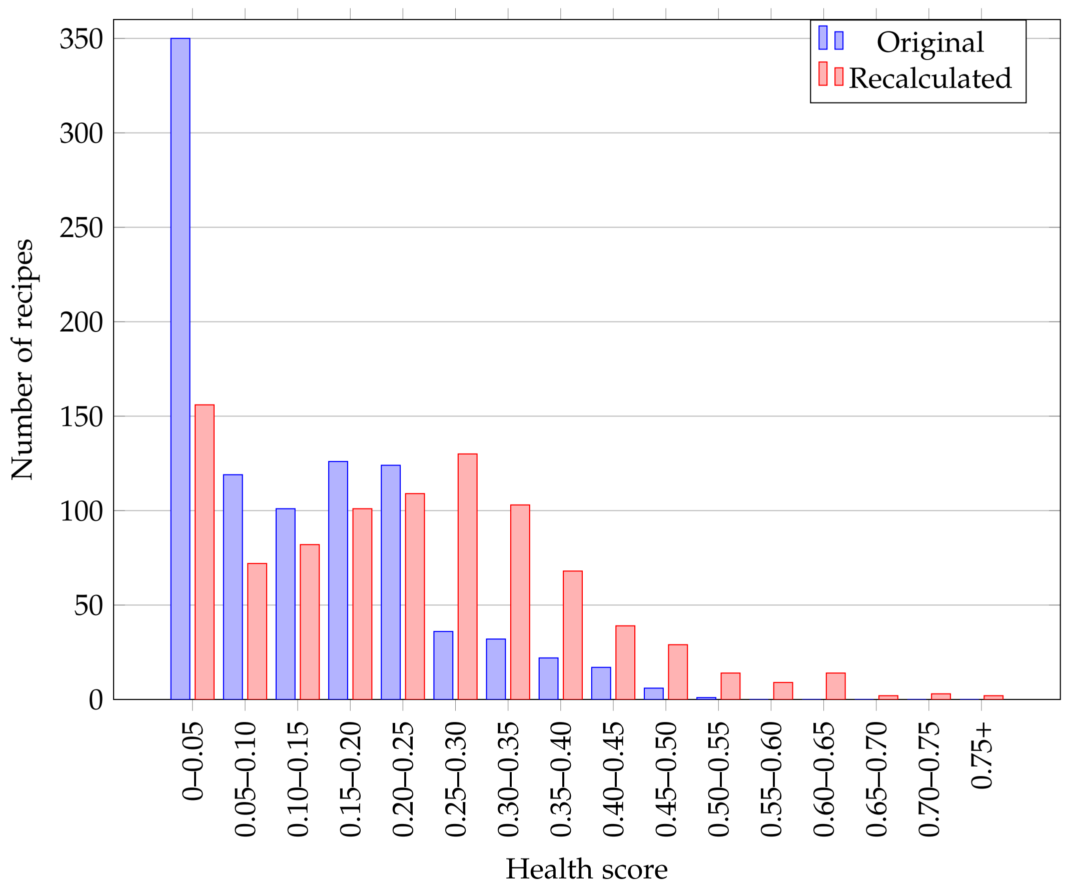


Figure 4.3: Histogram of original and recalculated health score

have continuous values, that it might influence the correlation values. However, a spearman's rank correlation was also done which show very similar correlation values. See appendix C.2 for the spearman's rank correlation matrix.
Looking at the two versions of the health score in this project(original and recalculated) it was also expected to find the recalculated health score to generally assign higher scores to dinner recipes. By looking at a histogram of the two health scores, see figure 4.3, this is also confirmed.

4.3 User study

In this project it was also of interest to see what impact different interface layouts would have on the users perceptions of a meal plan recommendation system. Therefore, the test setup is very similar to the one used in [7], by using the same three types of interfaces and measure on the same subjective measures. The differences lie in the type of recommender system. In [7] they test a movie recommender where the users were given a specific task of finding a movie of a specific theme. As such it is of interest to see if there are similarities in the findings for a movie recommender system and a meal recommender systems were recommendations are shown in bundles of 7.

4.3.1 Measures

Chen et al. [7] uses both objective measures and subjective measures. Their objective measures are clicks and time spent. In the study conducted in this project, only time is logged in order to measure the objective effort. The reason being that clicks is not really of interest in terms of first click, distinct clicks, etc. In the study by Chen et al. [7] it makes sense to see which items are clicked, as a show of interest, as their system is more exploratory in nature. However, in the case of this project, as exploration is very limited and bundles should be viewed as a whole, clicking behaviour is not of much relevance. Furthermore, due to time constraints it was not expected to obtain a lot of users for the study, so it was also expected to have scarce data for statistical evaluation. Therefore, the main focus of the study was to gain qualitative data.
The subjective measures used were also similar to the measures used by Chen et al. [7]. The measures and their respective questions are shown in table 4.7.

Measures	Questions 1-5 likert scala.
Decision confidence	I am confident in the bundle ratings i made
Perceived interface competence	The presentation made it easy to see and understand the contents of a bundle
Enjoyability	The presentation of a bundle was enjoyable

Table 4.7: Subjective measures and questions

These subjective measures are just a subset of all the measures identified in the framework explained in section 2.3.1, as in this study we are only interested in the interface. In the end of the questionnaire, users were also asked to select a favorite of the three interfaces.

4.3.2 Test setup

The users were asked to participate in the test in focus groups ranging from 2-5 persons. The reason for using focus groups was to promote discussions between users in post-study interviews.
Task were completed individually, where each user was given a paper with 3 tasks and questionnaires. They were initially introduced to the system and the tasks at hand. Each user was asked to take on the role of a fictitious character and were given 10 recipes which indicate the character's taste in food. They were then asked

to rate 5 different bundles of recipes for each interface layout (list, grid, pie) which results in 15 different bundles in total. The ordering of interfaces was shuffled between users to reduce bias in the viewing order. After rating 5 bundles, on each interface, the users were asked to complete questionnaires regarding their experience. A follow-up interview was done when every person in the focus group was done. The full questionnaire can be seen in appendix B.1 and interview summaries can be found in appendix B.2 - B.4. In total 16 users participated (6 female) with most users being friends and family. 14 were 20-30 of age, with the remaining(2) being 50-55.

4.3.3 User study results

Objective measures - Time spent

Time stamps were collected during the studies to measure the time spent on different interfaces. The averages of the three different interfaces are shown in table 4.8, measured in seconds. Even though pie has a shorter time spent there is no statistical significance (p > 0.05 in three paired t-tests).

	List	Grid	Pie
Mean	40.05	42.87	37.97

Table 4.8: Time averages on interfaces

Subjective measures

Overall from the questionnaire the overall favorite chosen was grid with 11 votes, list with 4 and pie with 1 (16 total). This corresponds well with the findings of the subjective measures. Here it is found that the interface competence and enjoyability of the grid interface is significantly higher compared to pie. Also, list is significant higher than pie in regards to enjoyability. Furthermore, this also corresponds well to the focus group interviews where most users mentioned that the pie interface seemed cluttered or confusing. P values for all t-tests can be seen in appendix C.3.

Subjective measures (1-5 likert)	Mean		
	List	Grid	Pie
Question 1 - Decision confidence	3.9375	3.9375	4.0625
Question 2 - Interface competence	4.125	**4.375**$*P$	3.75
Question 3 - Enjoyability	**4.3125**$*P$	**4.5625***P	3.3125

Table 4.9: Subjective measures - * denotes statistical significance in two tailed paired t-tests. List(L), Grid(G) and Pie(P)

Focus group interview

From the focus groups the most general comments where on images and recipe descriptions which failed to load. This can of course have an impact on the perception of the interfaces when evaluating them. Most agreed that images were the most important when rating bundles. Lastly, many felt that the pie interface was cluttered or confusing. Many additional comments where given where most were wanted features of the system.

Chapter 5

Discussion

Dataset

The dataset used within this report is as mentioned, in section 3, crawled from the site Epicurious.com. The categorization of the data was done using predefined categories from the danish food pyramid. However, due to the different steps needed to achieve the final list of crawled ingredients, namely writing the crawler, translating the ingredients from Danish to English, categorizing its position within the food pyramid and defining the weights in grams, we have to take a certain error rate into account. Some ingredients were simply not crawled, as they were not a part of the accepted ingredient list, furthermore some ingredients would get a wrong weight measure. An example of such, was the ingredient "chicken". If the user who defined the recipe did not specify the amount, for example as "1 chicken breast" our crawler would categorize it as a whole chicken weighing 2400 grams. This was due to static predefined values, which in some cases proved to be difficult to determine as certain ingredients weigh differently based on their size. We also noticed that user's had different ways of describing ingredients which resulted in some ingredients being misclassified. An example could be "onion" compared to "chili onion dip" which are classified in the same category, but are clearly different food products. To deal with the classification problems, we made the script write a file of all ingredients with weights of over 1000 grams, or those who simply weren't recognized by the crawler. The file was then used later for manual processing. Furthermore, we decided to discard all recipes associated with less than five ingredients. We do however know that certain ingredients are still missing for a small amount of recipes and we have also found several ingredients throughout the project period which have been tagged with the wrong weight amount. This is however something that we are aware of but still need to address, as the following results are affected by such errors.

SVD/KNN

The two recommender approaches used to initialize the generation of a bundle within this project is SVD and KNN. Both algorithms are based on a collaborative approach where the SVD is item-based and KNN is user-based. The main algorithm, SVD, was chosen based on previous work were it performed well. However, this thesis does not test any other algorithmic approach to how recommendations could have been made or even improved beyond the current MAE score. Additionally, only a single parameter was changed in the SVD, namely the amount of epochs. Similarly with KNN, the algorithm was only implemented to deal with new users, which were yet to be part of the SVD model. Again no real changes to the parameters or similarity function were made and the results are therefore only reflecting a small area of what might be possible to test and achieve. The limited testing of the two algorithms came as a result of time constraints within the project and it could therefore be interesting to further investigate and test how recommendations could be improved by testing several parameters within each algorithm.

Bundle recommendation

A crucial part in testing the bundle algorithm is the dataset used. The dataset used, consisting of 901 users and 971 recipes, is a somewhat sparse dataset where approximately half of the users only have rated 1 item each. Due to the sparsity it was difficult to get sufficient ground truth regarding user ratings for calculating precision and recall. Therefore, it was decided to only select the most active users in the dataset for the testing, which resulted in 65 users whom had rated at least 7 recipes. Testing the algorithm on 65 users severely effects the validity of the tests. As the 65 users still haven't rated a lot of the recipes within the dataset, there are many unrated recipes occurring in the recommended bundles. These unrated recipes had to be disregarded as they can possibly be either good or bad recipes for the user, so only recipes in which a ground truth was present were considered. Due to these issues, the precision and recall may be skewed. Ideally the testing would be done with a dense dataset such that precision and recall was always calculated using bundles with no items disregarded. The results regarding diversity could also possibly be misrepresented due to the similarity function used in the diversity calculation. The current similarity function used for the calculating diversity is jaccard similarity which does not capture the amount of the ingredients present in the recipes. Hence with jaccard similarity, a salad with a bit of chicken is considered the same as a chicken recipe with a bit of salad on the side. The calculated diversity might therefore give a misleading perspective on how diverse the recipes actually are in the bundles. In regards to which algorithm produces the healthiest bundles, this is based on which algorithm scores highest using our health score function. A high health scores indicates that the proportions of the bundles fit the food pyramid as best as possible. There is however no way to validate if these "healthy" bundles actually give a benefit to a users health. As these bundles represent weekly dinner plans for a user, it only considers a single meal during a whole day. Since eating healthy is usually associated with all meals eaten and not just a single meal, makes it hard to confirm if the benefits of the recommended dinner plans in regards to health. A dinner is however usually the largest meal eaten during the day(in a Danish context) and the healthy bundles can therefore somewhat ensure that a person gets a balanced amount of ingredient proportions during the week.

Health scores

The health scores developed in this paper are based upon the health guidelines from the Danish food pyramid and the meal guidelines from the healthy eating plate. The healthy eating plate is used in order to readjust the Danish guidelines towards a single meal.
Analyzing these health scores proved quite difficult as true evaluation is not possible. This is due to the fact that health guidelines is not an exact science. However, it can be argued that different health guidelines should be somewhat positively correlated. This was also found in the case between the FSA and WHO score, even though they are based on different guidelines. However, the health scores developed in this report did not share any correlation between these two scores. However, as it is assumed that there should be a positive correlation, there may be different factors as to why this is not found in the results. As discussed in the dataset section earlier, the crawled ingredient information introduces several possibilities for errors when calculating the developed health score. And, as the WHO and FSA are not reliant on this information, any errors in this data will make inconsistencies in the health scores. On the other hand, the nutritional information for the FSA and WHO health scores is also based on a natural language processing API which is provided by EDAMAM[1]. It is not known how the conversion from natural language to nutritional information is done in this API, so any differences from the method used in this project will also cause inconsistencies between the health scores.

User study

In this project a user study was also conducted in order to investigate the impact of interfaces on bundle recommendations. Due to a small amount of users the objective measures were not conclusive. In regards to the subjective measures it was found that the grid interface was perceived as the most competent, enjoyable and was also voted as favorite the most. This is not consistent with the results found the paper by Chen et al. [7] which suggests that users perceived the pie interface as having best competence and enjoyability. However, the system used in the study by Chen et al. allows users to browse and explore movie recommendations before picking one they like. In the system developed in this project, the purpose is to rate food bundles which does not promote much exploration, as every item is required to be reviewed. Therefore direct comparison may not provide much insights into which interface is overall superior. It should be said that there are some limitations of both studies. First of all, the amount of users in both studies is small, 16 and 24 users respectively. Secondly in both studies all the rating means are not really that different. They are all around 3-4 on a 1-5 likert scale. This may indicate that the users do not really perceive that big of a difference between all the interfaces. So there is not really a clear like and dislike towards the interfaces. One reason which could explain the general positive ratings for all interfaces are that in both studies users where chosen among friends, family and colleges. According to Recommender Systems Handbook [13, p.323], the use of these types of users is not recommended as they may want to please/not critique the system. Lastly, some error may also be directly linked to the system in the thesis, as the early version of the system did have it's flaws. Some of the reoccurring comments was on images, recipe descriptions and images which failed to load. This can of course have an impact on the subjective measures.

Chapter 6

Conclusion

This thesis explores the area of food recommendation, in which the focus is to recommend healthy meal plans to a user. The main problem statement of the thesis is:

- How can we recommend healthy meal plans to a user?

This question, leads to the interest of investigating the following sub-questions:

- How we can implement a health function to evaluate the healthiness of a meal plan?
- What considerations should be made in regards to presenting bundles of food items to a user?

Since the focus of this thesis is recommending weekly meal plans, we want to recommend a bundle containing 7 dinner recipes meant for each day of the week. We build upon a bundling algorithm used in previous work in which two main adjustments have been made. One of the adjustments was to incorporate a trained SVD model to the bundle algorithm. Another adjustment was to include a health function to the algorithm in order to evaluate the healthiness of a bundle.
Several different approaches to evaluating the healthiness of individual recipes was investigated, such as WHO and FSA used in related work. As most of these approaches used the nutritional content of a recipe to evaluate the healthiness, we instead present a new approach looking at the proportions of ingredients in a recipe. Our approach is mainly based on the Danish food pyramid to evaluate the best proportions. We introduce two versions of our approach, one used for evaluating a full week of eating and another better suited for evaluating single dinner meals. Both scores are tested for any correlation with the WHO and FSA approaches. The results however indicate no correlation between them.

Different versions of the bundle algorithm was tested in an offline setting where metrics such as precision, diversity and the healthiness was used to evaluate the algorithms. We also test if the number of returned bundles by the algorithm has any influence on its performance. The results of the testing indicate that the precision increases and the diversity slightly decreases as the number of returned bundles increases. The versions weighting health highly do, as expected, produce healthier bundles than versions which do not weight health highly. Weighting health highly show only a slight loss of precision and diversity.

Besides the whole process of generating bundles and evaluating the healthiness of a bundle, another interesting question is the aspect of presenting bundles to a user. As a presentation of a bundle is more complex than presenting a single item, we investigate different possible layouts for a bundle and how they possibly affect users. The layouts tested in this thesis is inspired from the paper by Li Chen and Ho Keung Tsoi[7], which investigates the 3 layouts list, grid, and pie in a movie recommendation context. We evaluate the layouts in a user study with 16 participants where they were presented a bundle of 7 recipes for each layout. The layouts are evaluated in regards to decision confidence, interface competence and enjoyability. The findings of the user study show that the users generally preferred the grid layout over the pie layout. The grid layout had significantly higher interface competence and enjoyability when compared to the pie interface. Our findings is therefore not consistent with the findings in [7], but our system is not used in the same domain as their, thus a direct comparison between our results and theirs would not be meaningful.

Chapter 7

Future work

In regards to developing a food recommender system this project has provided a lot of insights. From the user study a lot of insights were especially gained in regards to what features should be implemented in a future system. Among these features were the the inclusion of needed utensils, cooking time, budget and ability to customize the presented bundles, see appendix B for all mentioned features. The future work of more specific areas of the project are described below.

Recommendations

Due to time constraints the evaluation of the SVD and KNN algorithm used within this project is limited in regards to a single parameter change within each of them. To further investigate how the MAE could be further improved a more comprehensive evaluation could prove beneficial. Such tests should take a deeper look at how each of the parameters within the algorithms are influencing the overall results. It could also be interesting to complement the test by taking new algorithmic approaches into the final implementation to test whether the two chosen for this project are the most accurate given the task presented.

Health score

In regards to the problems between the correlation of the developed health scores and the WHO and FSA, these errors could be avoided if one should develop a complete system from the ground up. The solution would simply be to restrict natural language when users specify ingredients in a recipe. Such that users should always select ingredients and their amounts from an accepted ingredients list.

Bundle algorithm

In the current iteration of the bundle algorithm the only constraint is the size of a bundle. One of the comments in the conducted user study was that it would nice to have constraints such as cooking time for the recommendations. The cooking time would be appreciated as there is often limits on how much time one would spend on cooking. The comments regarding the cooking time gave suggestions of allowing the possibility to constrain the bundles to only include recipes with a cooking time of 20 minutes. As for future validation of the bundle algorithms performance, a more dense dataset would be preferred. This is due to how a dense dataset would provide a better ground truth to recipes we currently have available but are still unrated.

User study

The user study in the project had a very small amount of participants with a very narrow demographic. So in future work it is recommended that a study on a larger scale is conducted. This study could also be expanded to include the full framework presented by Pu et al. [28] in order to investigate the overall system more in depth.

Bibliography

[1] Edamam - eat better! https://www.edamam.com/. (Accessed on 06/12/2018).

[2] Madpyramiden historie | madpyramiden.dk. http://www.madpyramiden.dk/historie. (Accessed on 04/13/2018).

[3] Nutrition basics | washington state university. https://mynutrition.wsu.edu/nutrition-basics/. (Accessed on 04/23/2018).

[4] Who | dietary recommendations / nutritional requirements. http://www.who.int/nutrition/topics/nutrecomm/en/. (Accessed on 04/24/2018).

[5] Forstå madpyramiden | madpyramiden. http://www.madpyramiden.dk/forstaa-madpyramiden. (Accessed on 06/13/2018).

[6] Xavier Amatriain and Justin Basilico. Netflix recommendations: Beyond the 5 stars (part 1). https://medium.com/netflix-techblog/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429, April 2012. (Accessed on 05/31/2018).

[7] Li Chen and Ho Keung Tsoi. Users' decision behavior in recommender interfaces: Impact of layout design. In *RecSys' 11 Workshop on Human Decision Making in Recommender Systems*, 2011.

[8] Lene Møller Christensen, Karsten Kørup, Ellen Trolle, and Sisse Fagt. Måltidsvaner for voksne med kort uddannelse 2005-2008. https://maaltidspartnerskabet.dk/wp-content/uploads/Rapport-M%C3%A5ltidsvaner-kortuddannede-voksne.pdf, 2012. (Accessed on 05/18/2018).

[9] David Elsweiler Christoph Trattner. Investigating the healthiness of internet-sourced recipes, 2017.

[10] COOP. Danskernes madvaner 2016 coop analyse. https://om.coop.dk/Upload/om.coop.dk/Publikationer/analyser/Danskernes%20Madvaner%202016_Coop%20Analyse.pdf, 2016. (Accessed on 05/18/2018).

[11] Larissa S Drescher, Silke Thiele, and Gert BM Mensink. A new index to measure healthy food diversity better reflects a healthy diet than traditional measures. *The Journal of nutrition*, 137(3):647–651, 2007.

[12] David Elsweiler and Morgan Harvey. Towards automatic meal plan recommendations for balanced nutrition. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 313–316. ACM, 2015.

[13] Bracha Shapira Francesco Ricci, Lior Rokach. *Recommender Systems Handbook*. Springer, Boston, MA, 2nd edition, 2015.

[14] Edel Garcia. Singular value decomposition (svd) a fast track tutorial. `https://fenix.tecnico.ulisboa.pt/downloadFile/3779576344458/singular-value-decomposition-fast-track-tutorial.pdf`. (Accessed on 14/06/2018).

[15] Google. Angular. `https://angular.io`. (Accessed on 05/16/2018).

[16] Nicolas Hug. Surprise. `http://surpriselib.com`.

[17] Nicolas Hug. k-nn inspired algorithms. `http://surprise.readthedocs.io/en/stable/knn_inspired.html#surprise.prediction_algorithms.knns.KNNBasic`, 2015. (Accessed on 04/16/2018).

[18] Ida Husby. Madpyramiden: Revidering og revitalisering af madpyramiden. sundhed, smag og klima. 2011.

[19] Bart P Knijnenburg, Martijn C Willemsen, Zeno Gantner, Hakan Soncu, and Chris Newell. Explaining the user experience of recommender systems. *User Modeling and User-Adapted Interaction*, 22(4-5):441–504, 2012.

[20] Bogi H. Eliasen Mark V. Nielsen, Kristian B. Dyhrberg. Exploring bundle recommendations for groups in a food context. 2017.

[21] Sean M McNee, John Riedl, and Joseph A Konstan. Making recommendations better: an analytic model for human-recommender interaction. In *CHI'06 extended abstracts on Human factors in computing systems*, pages 1103–1108. ACM, 2006.

[22] Microsoft. Using the rest services with .net. `https://msdn.microsoft.com/en-us/library/jj819168.aspx`. (Accessed on 04/16/2018).

[23] Sarah Ball Alison L. Miller Julie Lumeng Karen E. Peterson Nicole Kasper, Cami Mandell. The healthy meal index: A tool for measuring the healthfulness of meals served to children. 2015.

[24] Harvard T.H. Chan School of Public Health. Healthy eating plate & healthy eating pyramid | the nutrition source. `https://www.hsph.harvard.edu/nutritionsource/healthy-eating-plate/`. (Accessed on 05/07/2018).

[25] MIT OpenCourseWare. Singular value decomposition. `https://ocw.mit.edu/courses/mathematics/18-06sc-linear-algebra-fall-2011/positive-definite-matrices-and-applications/singular-value-decomposition/MIT18_06SCF11_Ses3.5sum.pdf`, 2011. (Accessed on 04/13/2018).

[26] Mark Otto. Bootstrap. `https://getbootstrap.com`. (Accessed on 05/16/2018).

[27] AN Pedersen, T Christensen, J Matthiessen, VK Knudsen, MR Sørensen, A Biltoft-Jensen, and S Fagt. Danskernes kostvaner 2011–2013: Hovedresultater [dietary habits in denmark 2011–2013: main results]. *Søborg (Denmark): National Food Institute, Technical University of Denmark*, 2015.

[28] Pearl Pu, Li Chen, and Rong Hu. A user-centric evaluation framework for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 157–164. ACM, 2011.

[29] Sebastian Raschka. *Python Machine Learning*. Packt Publishing, 1st edition, 2015.

[30] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[31] Martin White Simon Howard, Jean Adams. Nutritional content of supermarket ready meals and recipes by television chefs in the united kingdom, 2012.

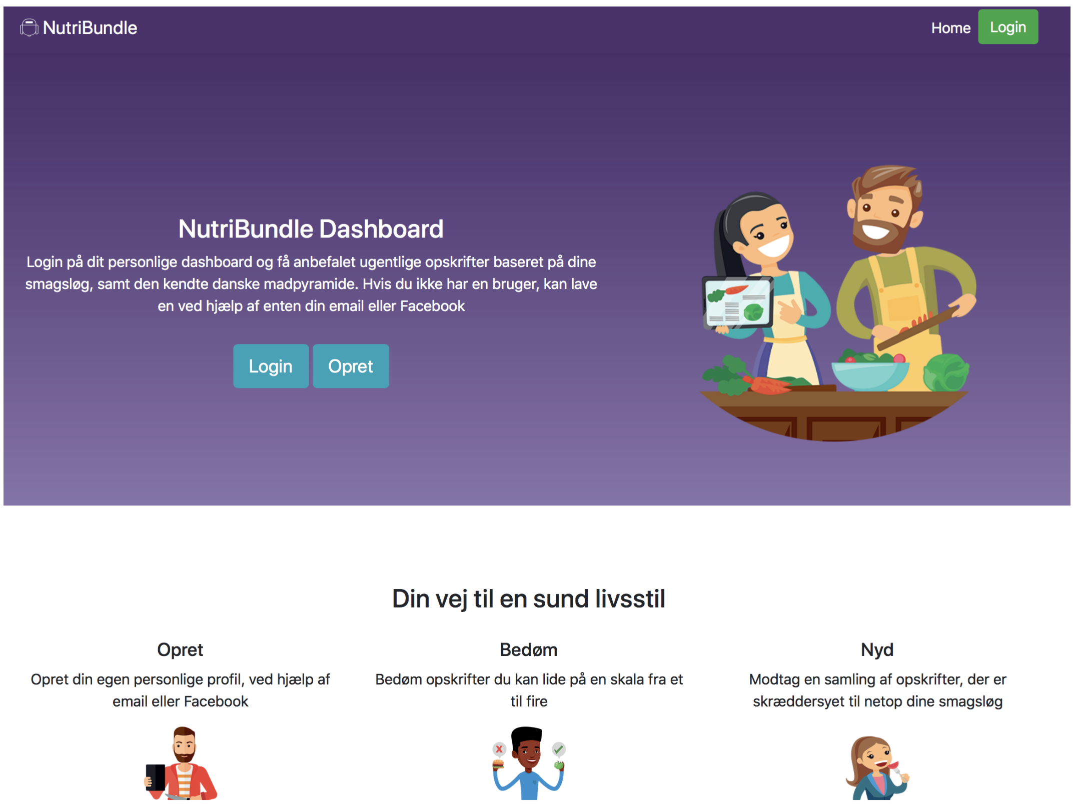
[32] Danmarks Tekniske universitet. Mål, vægt og portionsstørrelser på fødevarer. `http://www.fooddata.dk/keyhole/userfiles/mvfodevarer.pdf`, January 2013. (Accessed on 04/16/2018).

[33] Chris Volinsky Yehuda Koren, Robert Bell. Factorization techniques for recommender systems, 2009.

Appendix A

Web application Layout

A.1 Front page



NutriBundle

Home Login

NutriBundle Dashboard

Login på dit personlige dashboard og få anbefalet ugentlige opskrifter baseret på dine smagsløg, samt den kendte danske madpyramide. Hvis du ikke har en bruger, kan lave en ved hjælp af enten din email eller Facebook

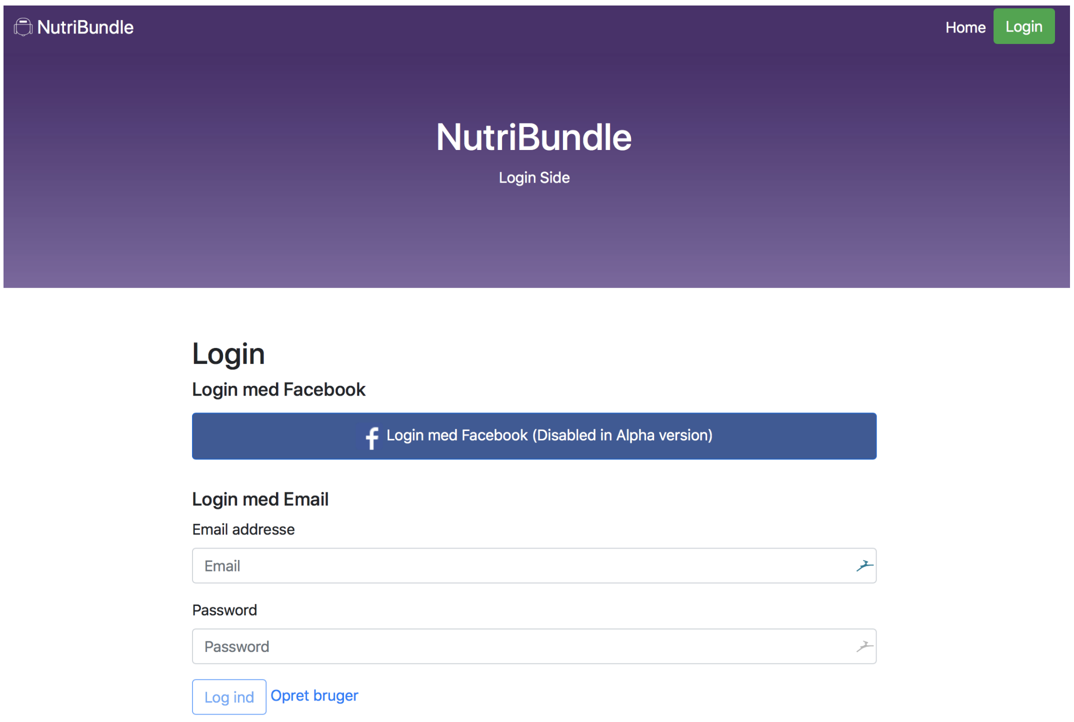
Login Opret

Din vej til en sund livsstil

Opret	Bedøm	Nyd
Opret din egen personlige profil, ved hjælp af email eller Facebook	Bedøm opskrifter du kan lide på en skala fra et til fire	Modtag en samling af opskrifter, der er skræddersyet til netop dine smagsløg

Figure A.1: Web application front page

A.2 Login page


NutriBundle

Home Login

NutriBundle

Login Side

Login

Login med Facebook

Login med Facebook (Disabled in Alpha version)

Login med Email

Email addresse

Email

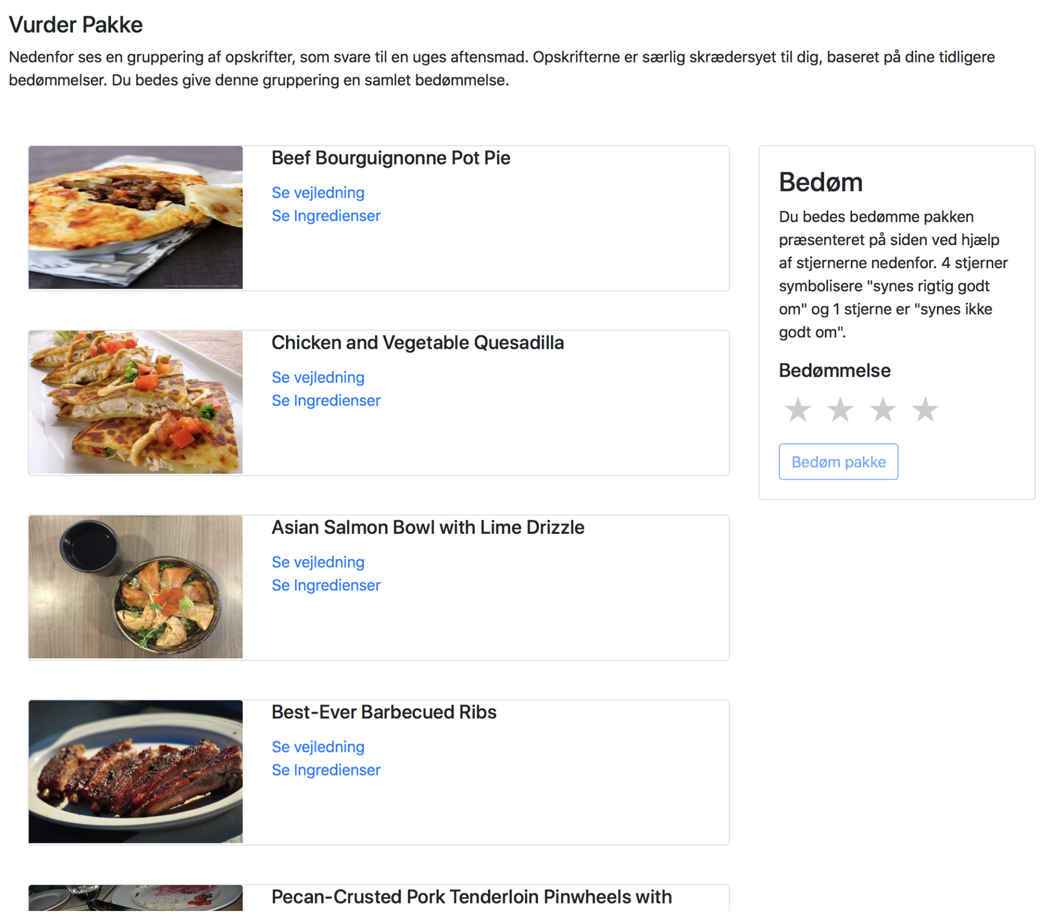
Password

Password

Log ind Opret bruger

Figure A.2: Web application login page

A.3 List interface



Vurder Pakke

Nedenfor ses en gruppering af opskrifter, som svare til en uges aftensmad. Opskrifterne er særlig skrædersyet til dig, baseret på dine tidligere bedømmelser. Du bedes give denne gruppering en samlet bedømmelse.

Beef Bourguignonne Pot Pie
Se vejledning
Se Ingredienser

Chicken and Vegetable Quesadilla
Se vejledning
Se Ingredienser

Asian Salmon Bowl with Lime Drizzle
Se vejledning
Se Ingredienser

Best-Ever Barbecued Ribs
Se vejledning
Se Ingredienser

Pecan-Crusted Pork Tenderloin Pinwheels with

Bedøm

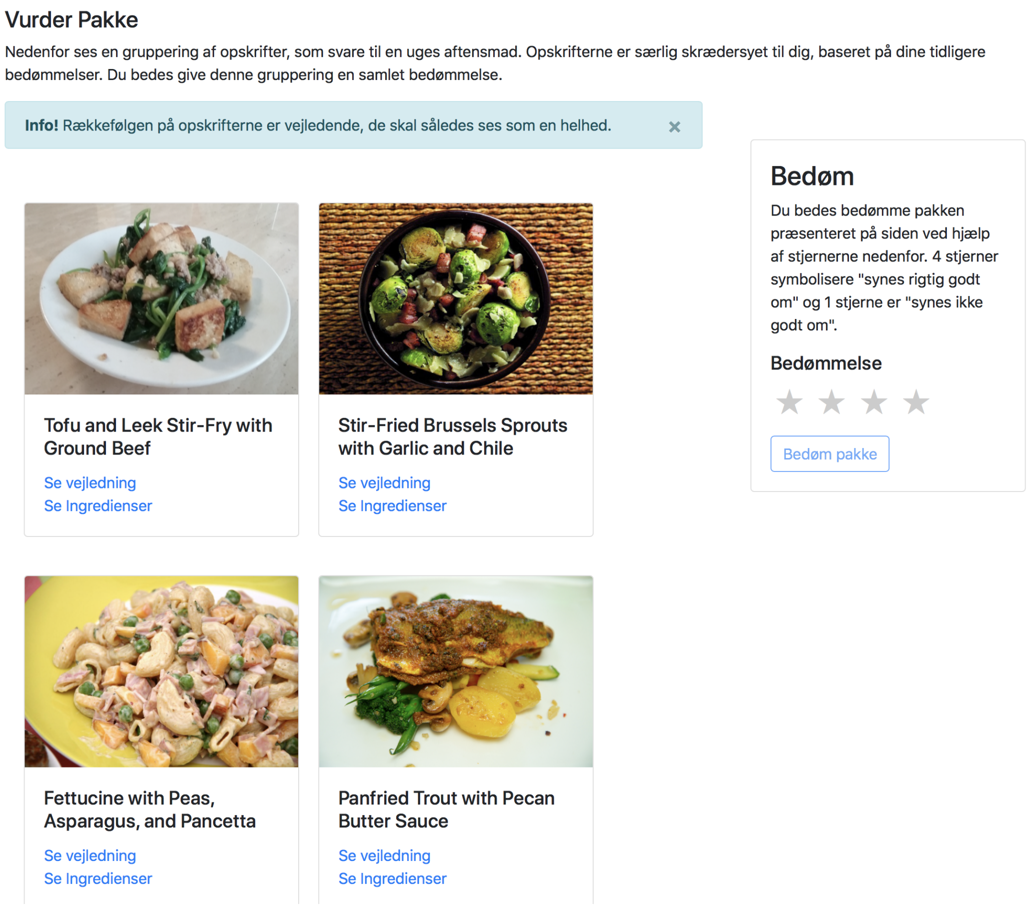
Du bedes bedømme pakken præsenteret på siden ved hjælp af stjernerne nedenfor. 4 stjerner symbolisere "synes rigtig godt om" og 1 stjerne er "synes ikke godt om".

Bedømmelse

★ ★ ★ ★

Figure A.3: Web application - Bundle list view

A.4 Grid interface



Vurder Pakke

Nedenfor ses en gruppering af opskrifter, som svare til en uges aftensmad. Opskrifterne er særlig skrædersyet til dig, baseret på dine tidligere bedømmelser. Du bedes give denne gruppering en samlet bedømmelse.

Info! Rækkefølgen på opskrifterne er vejledende, de skal således ses som en helhed.

Tofu and Leek Stir-Fry with Ground Beef

Se vejledning
Se Ingredienser

Stir-Fried Brussels Sprouts with Garlic and Chile

Se vejledning
Se Ingredienser

Fettucine with Peas, Asparagus, and Pancetta

Se vejledning
Se Ingredienser

Panfried Trout with Pecan Butter Sauce

Se vejledning
Se Ingredienser

Bedøm

Du bedes bedømme pakken præsenteret på siden ved hjælp af stjernerne nedenfor. 4 stjerner symbolisere "synes rigtig godt om" og 1 stjerne er "synes ikke godt om".

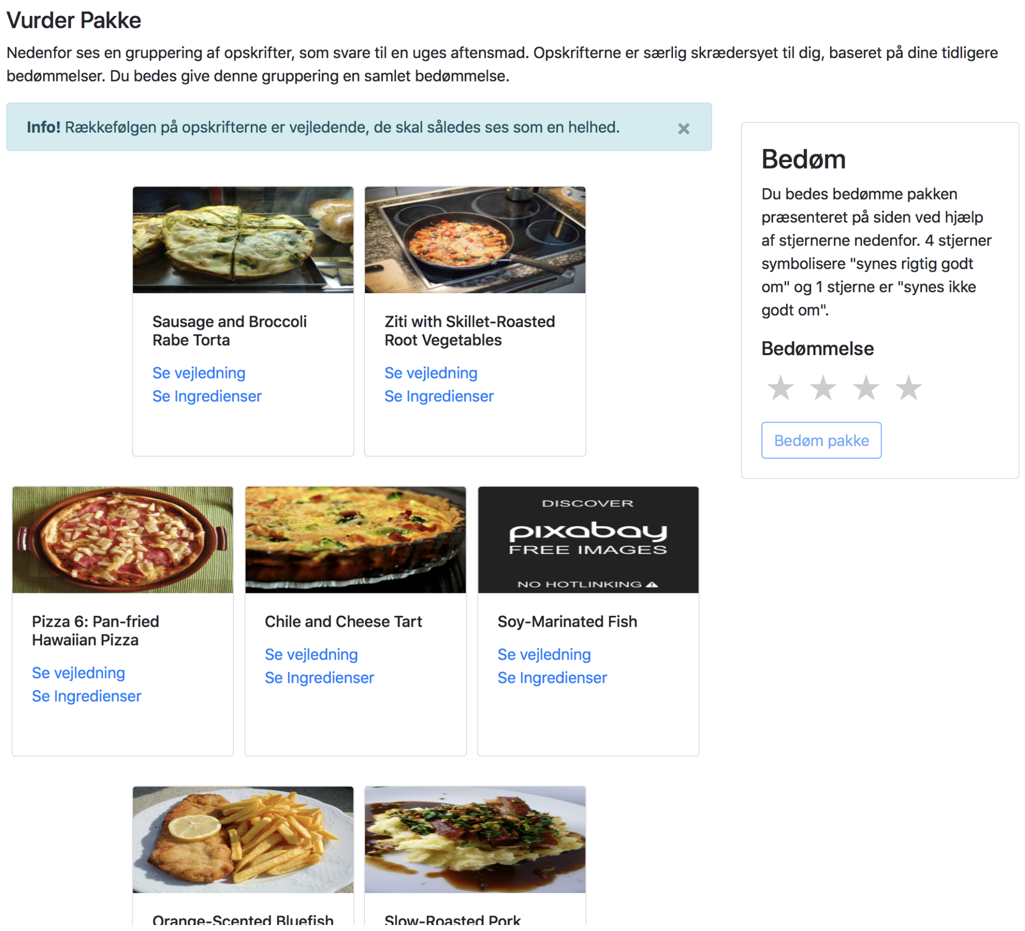
Bedømmelse

★ ★ ★ ★

Bedøm pakke

Figure A.4: Web application - Bundle Grid view

A.5 Pie interface


Vurder Pakke

Nedenfor ses en gruppering af opskrifter, som svare til en uges aftensmad. Opskrifterne er særlig skræddersyet til dig, baseret på dine tidligere bedømmelser. Du bedes give denne gruppering en samlet bedømmelse.

Info! Rækkefølgen på opskrifterne er vejledende, de skal således ses som en helhed. ×

Sausage and Broccoli Rabe Torta
Se vejledning
Se Ingredienser

Ziti with Skillet-Roasted Root Vegetables
Se vejledning
Se Ingredienser

Pizza 6: Pan-fried Hawaiian Pizza
Se vejledning
Se Ingredienser

Chile and Cheese Tart
Se vejledning
Se Ingredienser

Soy-Marinated Fish
Se vejledning
Se Ingredienser

Orange-Scented Bluefish

Slow-Roasted Pork

Bedøm

Du bedes bedømme pakken præsenteret på siden ved hjælp af stjernerne nedenfor. 4 stjerner symbolisere "synes rigtig godt om" og 1 stjerne er "synes ikke godt om".

Bedømmelse

★ ★ ★ ★

Figure A.5: Web application - Bundle Pie view

Appendix B

Focus group interviews

B.1 Full questionnaire

See attached zip-file: Liste - Questionnaire.pdf

B.2 Unique comments

User 1	Would be nice to have some information about which cooking utensils are needed. Fx. do i need a grill, a large oven tray, etc.
User 8	Easier if it was local recipes (Both in regards to the general danish food culture and in regards to ingredient availability)
User 9	Would like to see number of ingredients needed. Saw many spices as unnecessary. Wish that you could filter on different things
User 11	Would like to see budget/prices of dishes and wish there was some considerations regarding utilizing the ingredients fully. To not have leftover ingredients.
User 12	Would like to be able to have more flexibility in bundles. That is, to be able to change/insert individual items in bundles (fx, i would like to have pizza(something unhealthy) on Friday, so make the rest of the week healthy).
User 13	It was fine to rate the bundles from the shown information. Would mainly use for inspiration based on own taste preferences.
User 20	Mainly evaluated from pictures - the first impression. List and grid very close List and grid were more manageable as the user was accustomed to these formats. Pie was too much information at once. No idea where to start. Many of the dishes seemed to pretty complex for every day dinner. Recipes looked more like weekend dinners.

Table B.1: Individual comments - only unique comments shown

B.3 General comments

- Missing some pictures was a concern.
- Recipe instructions failing to load.
- Ingredients list was somewhat confusing.
- Generally used images for rating.
- Used ingredients list and instructions if the image and name was not sufficient.
- Pie seemed confusing or cluttered.

B.4 Additional features wanted

- Cooking utensils needed
- Cooking time
- Budget
- Key ingredients displayed
- Categories on ingredients
- Maximizing ingredient usage
- Ability to specify/lock items in bundle
- Ability to swap items in bundle
- Specify for how many people.

Appendix C

Tables appendix

C.1 Paired t-test p-values

Two tailed paired t-test	WHO	FSA	Rating recalculated
WHO			
FSA	7,59E-53		
Rating recalculated	1,05E-20	3,14E-13	
Rating original	1,98E-123	2,73E-03	9,91E-86

Table C.1: Two tailed paired t-test p-values

C.2 Spearman's rank corrleation matrix

Spearman's rank correlation matrix	WHO	FSA	Health score recalculated	Health score original
WHO	1			
FSA	0,297623175	1		
Health score recalculated	-0,007276096	0,014534677	1	
Health score original	0,049395605	-0,017561676	0,486454253	1

Table C.2: Spearman's rank correlation

2 tailed paired t-test p values	List - Grid	Grid - Pie	List - Pie
Decision confidence	1	0,431957081	0,4973105458
Interface competence	0,2997219144	0,02759109391	0,2702110644
Enjoyability	0,1638756137	0,005132618168	0,006442512189

Table C.3: Subjective measures two tailed paired t-tests p values

C.3 SVD - K = 30

SVD - k = 30										
30 epoch	test 1	test 2	test 3	test 4	test 5	test 6	test 7	test 8	test 9	test 10
Fold 1	0.62197	0.62061	0.62	0.62003	0.61902	0.62069	0.62126	0.618	0.62037	0.6218
Fold 2	0.64749	0.644	0.64498	0.64647	0.64494	0.64673	0.64767	0.64559	0.64499	0.6474
Fold 3	1.00709	1.00678	1.00666	1.00762	1.01017	1.00685	1.00909	1.00772	1.0095	1.0072
Fold 4	0.7161	0.71172	0.71049	0.71075	0.71081	0.71259	0.71116	0.71142	0.71321	0.70933
Fold 5	0.7744	0.77297	0.77566	0.77729	0.77577	0.77701	0.77387	0.77547	0.77452	0.78452
MAE	0.75341	0.751216	0.751558	0.752432	0.752142	0.752774	0.75261	0.75164	0.752518	0.75405
Total MAE: 0.7524										

Table C.4: SVD results k = 30

C.4 KNN - k = 23

KNN(k-max=23, k-min=1) Results				
Fold 1	Fold 2	Fold 3	fold 4	Fold 5
0.6305	0.6320	0.9856	0.7397	0.7521
Total MAE: 0.7479				

C.5 KNN - k = 40

KNN(k-max=40, k-min=1) Results				
Fold 1	Fold 2	Fold 3	fold 4	Fold 5
0.7533	0.7547	0.9663	0.6494	0.6423
Total MAE: 0.7532				

Appendix D

Health score functions

D.1 Recalculated health functions

	<optimal	>optimal
Top 1	25.97402597*x-1.58336885E-16	-25.97402597*x+2
Top 2	4.72813239*x-4.41226916E-16	-4.72813239*x+2
Middle	4*x+0	-4*x+2
Bottom	2*x+0	-2*x+2

Table D.1: Recalculated health functions

D.2 Original health functions

	<optimal	>optimal
Top 1	50*x-7.35680456E-17	-50*x+2
Top 2	9.09090909*x+1.65666092E-16	-9.09090909*x+2
Middle	2.27272727*x+1.65666092E-16	-2.27272727*x+2
Bottom	2.3255814*x-2.6490016E-16	-2.3255814*x+2

Table D.2: Original health functions

Appendix E

Code

E.1 SVD/KNN implementation

See attached zip-file: SVD-KNN-api.zip

E.2 Bundle algorithm - adjusted

See attached zip-file: BundleWithHealth.rar

E.3 SVD/KNN testing implementation

See attached zip-file: Python Implementation.zip

E.4 Nutrition crawler

See attached zip-file: nutricrawler4.0.zip