Mining Recipe Data

Benjamin van der Burgh

August 2, 2016

1 Introduction

In many ways, the kitchen can be seen as the world's oldest laboratory. People have experimented with a large number of ingredients, their complex interplay and preparation methods ever since we started using tools. With the advent of Web 2.0 and its focus on user-generated content and social networking, a lot of data about cooking is now available. On websites such as Allrecipes [3] and Food Network [6], users can submit their own recipes and rate recipes of other users. Instead of browsing through cookbooks, food lovers can now discover new recipes by limiting themselves to a specific category, such as 'Christmas' or 'Quick & Easy', or by following other users on the platform.

While these are interesting developments for cooks, it also allows us to study the recipes in much greater detail. Traditionally, clustering and classification methods have been used to identify consumer patterns [19], while more recently, matrix factorization methods have been applied [4]. It is often assumed that if a recipe can be found with a certain pairing of ingredients, this pairing must be generally favorable [11]. Using the user ratings available, however, it becomes possible to find more specific patterns for certain users groups. For example, the Food Pairing® theory states that ingredients with overlapping flavor compounds are often used in Western cuisine [7]. On the other hand, it is often thought that this theory does not hold for Eastern cuisine, which might indicate a strong cultural bias towards certain combinations of ingredients in relation to the combination of flavor compounds [2, 14]. Now that a large volume of recipes and user reviews is available, we can start to find out which ingredients make a good combination in much finer detail.

Since users rate only a very small fraction of all recipes in the database, it is non-trivial to determine whether a user likes a recipe that was not rated by this user [17]. Several well-known *Collaborative Filtering* (CF) methods are used to obtain an estimate of these unrated recipes. Instead of a user $\times$ recipe matrix, a recipe $\times$ ingredient variant can be used as well, which can be used to suggest ingredients to complete a recipe.

In this research, several aspects of recipes and ingredient pairing are studied using a dataset that was derived from the Allrecipes platform. This dataset was enriched with data from FooDB [8], a dataset that includes information on the flavor components of ingredients. The resulting dataset is explored from various perspectives, involving ingredient lists and user ratings, in order to both validate the data and get a better understanding. After that, CF techniques are investigated that are used to get a broader knowledge on user preferences in relation to ingredient combinations.

2 Related Work

Utilizing data science to study the fundamental theories that underly our perception of food is a relatively new effort. With the recent introduction of social networks focussing on food and the availability of chemical information about food products [8], it is now possible to analyze much larger quantities of data. While chemical engineering has since long influenced the search for flavor enhancing additives, little is known about what makes a good recipe. One well-known, but still controversial, theory is that of Food Pairing®, which states that ingredients with overlapping flavors make a good combination [7]. Although some research has shown the theory to hold for Western cuisine, it does not for the Eastern and southern European cuisines [2, 13]. Others have stated that the hypothesis lacks a solid basis and even more, simpler food pairing alternatives exist [14]. What is still unargued is that an interesting combination of ingredients is one of the most important components of a successful recipe, making the ingredient lists themselves a good starting point for study. One interesting direction in this way is that of *recipe completion*, an information retrieval task in which the goal is to complete a recipe, given its ingredient list [4]. The use of *Non-negative Matrix Factorization* (NMF) is a promising approach that is both able to retrieve the missing ingredients and creates an understandable model of a recipe database [21, 4]. In this study we try two other methods for recipe completion using association rules and item similarities.

3 Datasets

For the purpose of this research, several new datasets were constructed. While there exist a few recipe datasets [20, 5], none of them include both the ingredients and user ratings. The data was scraped from the food-focussed social networking service Allrecipes [3]. The resulting dataset was enriched with data from FooDB [8], a comprehensive, publicly available dataset on food constituents, chemistry and biology. This section describes how the dataset was obtained, which choices were made, and gives a high-level overview of its contents.

3.1 FooDB

FooDB is a freely available resource on food constituent, chemistry and biology. It contains a large amount of information on the chemicals that give foods their flavor, color, taste, texture and aroma. Another convenient feature is that foods in FooDB are named in quite general terms. For example, instead of having specific descriptions of various types of milk, such as *low far*, *skimmed milk* and *whole milk*, these are referred to simply as milk. These different types of milk taste differently because the ratio of chemical components is different, but the components themselves are mostly the same. These names can therefore be used as a equivalence group to which to which we will map more specific ingredient names (see Section 3.2). It contains information about 893 foodstuffs, such as a general and scientific name and a food group (e.g. 'Herbs and Spices').

3.2 Allrecipes

Visitors of Allrecipes can browse through the collection of recipes that were submitted by its members. Recipes are categorized by the type of course, by season, special occasions (e.g. 'Christmas') among others. Anyone can sign up for a membership, which gives access to some additional functionality and enables members to submit their own recipes and rate other recipes. Two datasets were generated from the information available at Allrecipes: a dataset consisting of ingredients used in recipes and a dataset of user ratings of recipes. This section describes how these two datasets were obtained and provides a preliminary analysis of their contents.

3.2.1 Ingredient sets

Resources on the website, such as recipes, reviews and users, are given a unique identifier in a subsequent sequence, which makes it trivial to download the HTML page for each resource. Furthermore, the pages follow some of schemas for structured data markup of Schema.org. The recipe pages, for example, follow the `Recipe` schema including properties such as `totalTime` (total cooking time), `aggregateRating` (average rating of the recipe) and `author` (submitter of the recipe). The preparation procedure itself was omitted, since it is not of interest for this study. The only attribute that required some additional processing were the ingredients themselves. There seems to be no standardized way in which these ingredients are provided, although they have some structure. Consider for example: "2 cups Cascadian Farm® organic frozen sweet corn, thawed". It includes a quantity in cups, a brandname, a production paradigm, taste depiction and state. For the purpose of this research, we are interested only in 'corn', which we will call a *standardized ingredient*. These are obtained by removing the quantity and brandname, using a manually created grammar, and matching the resulting string to a list of standardized ingredients in the FooDB dataset (described below).

Table 3.1: An overview of recipe attributes.

Attribute	Description
calories	nutritional energy in kilocalories
cooking_time	cooking time in seconds
id	Allrecipe recipe identifier
ingredients	a list of standardized ingredients
name	name of the recipe
nutrients	nutritional information, e.g. salt and sugar quantities
preparation_time	preparation time (for cutting, washing, etc.)
total_time	sum of preparation and cooking time
yields	number of portions for the given ingredient quantities

Table 3.1 provides an overview of all the recipe attributes that were scraped, most of which are not the focus of this research. Table 3.2 provides a summary of the datasets. We will be mostly concerned with the ingredient data, which can be represented as a sparse binary matrix Y $(n \times m)$:

$$Y_{ri} = \begin{cases} 1, & \text{if recipe } r \text{ contains ingredient } i \\ 0, & \text{otherwise} \end{cases} \qquad \text{(Ingredient matrix)}$$

Table 3.3 provides an overview of the top 10 most frequent ingredients,

Table 3.2: Summary of the datasets

Description	#
Recipes	91 910
Recipes with at least one rating	66 846
Standardized ingredients also in FooDB	406
Users	745 228
Ratings	3 253 234
Density of ingredient matrix Y	1.95×10^{-2}
Density of rating matrix R	6.53×10^{-5}

confirming our intuition. Fig. 3.1 plots the frequencies of all ingredients on a logarithmic scale, providing a global perspective. It shows that the ingredient frequencies follow a log-linear model, which can lead to problems when mining for patterns, since the available data for many ingredients is very limited (see Section 5.1).

Table 3.3: Top 10 of most frequent ingredients

Ingredient	Frequency	Relative
salt	47,292	$6.55 \cdot 10^{-2}$
sugar	33,161	$4.59 \cdot 10^{-2}$
pepper	32,797	$4.54 \cdot 10^{-2}$
onion	29,488	$4.08 \cdot 10^{-2}$
eggs	23,830	$3.3 \cdot 10^{-2}$
water	23,449	$3.25 \cdot 10^{-2}$
garlic	22,668	$3.14 \cdot 10^{-2}$
cheese	22,662	$3.14 \cdot 10^{-2}$
butter	22,410	$3.1 \cdot 10^{-2}$
flour	21,865	$3.03 \cdot 10^{-2}$

3.2.2 User ratings

The recipe pages provide some, but not all, of the reviews for that specific recipe, so these were downloaded and parsed separately. This dataset consists of $U \times M \times S$ tuples, where U is the set of users of size p, M the set of meals (or recipes) of size q, and scores $S = \{1, 2, \ldots, 5\}$. A rating is an assignment of a score $s \in S$ by a user $u \in U$ to a meal $m \in M$, i.e., $r(u_i, m_j) \mapsto S$. These tuples can be conveniently represented as a sparse matrix R $(p \times q)$ of ratings:

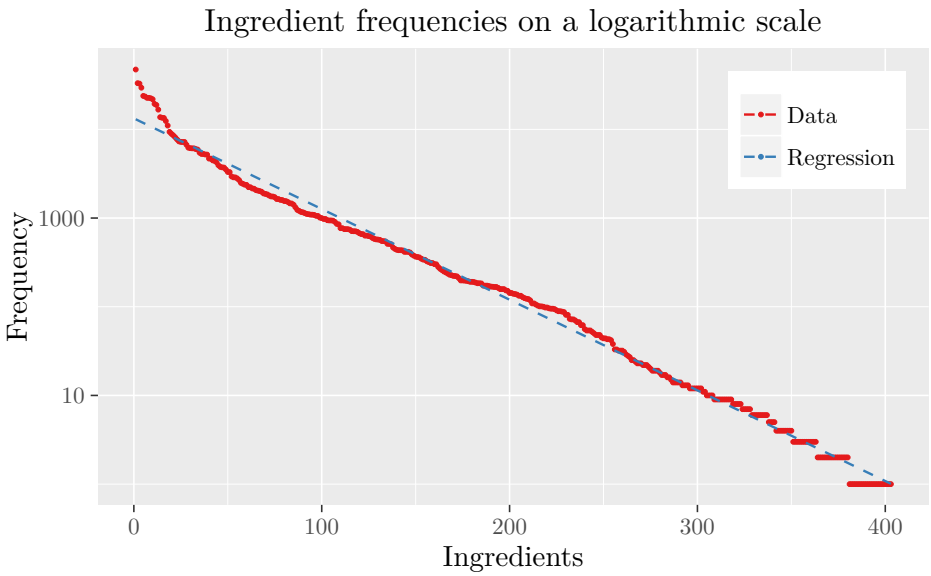

Figure 3.1: Ingredient frequencies (red) follow the log-linear model (blue), showing that a small proportion of ingredients are used in most recipes, while many are scarcely used.

$$R_{ij} = \begin{cases} r(u_i, m_j), & \text{if user } i \text{ rated meal } j \\ 0, & \text{otherwise} \end{cases} \qquad \text{(Rating matrix)}$$

Meals with no ratings are left out of the rating matrix so that no columns have only zeros. Table 3.2, that was previously referred to, also shows some statistics on the rating data. About two thirds of the recipes have at least one rating. Another observation is that the density of the rating matrix several orders of magnitudes lower, which is caused by the fact that there are many more users than ingredients and most users rate only one recipe. This is also shown in Fig. 3.3. The majority of users only reviewed a single recipe and 90 % of the users has reviewed ≤ 7 recipes. Knowing that many users submitted very few ratings might make it hard to make predictions about the ratings for recipes the user did not rate. On the other hand, Allrecipes has a few very active users with the top reviewer being a user that submitted 5250 ratings. Fig. 3.4 zooms in on this end of tail and shows that is very uncommon. In fact, only 113 users submitted more than 1000 reviews.

Fig. 3.2 shows the number of reviews per score. It seems that Allrecipes has a strong bias towards positive reviews, which might indicate that people tend to rate recipes that they appreciated. Another explanation could be that

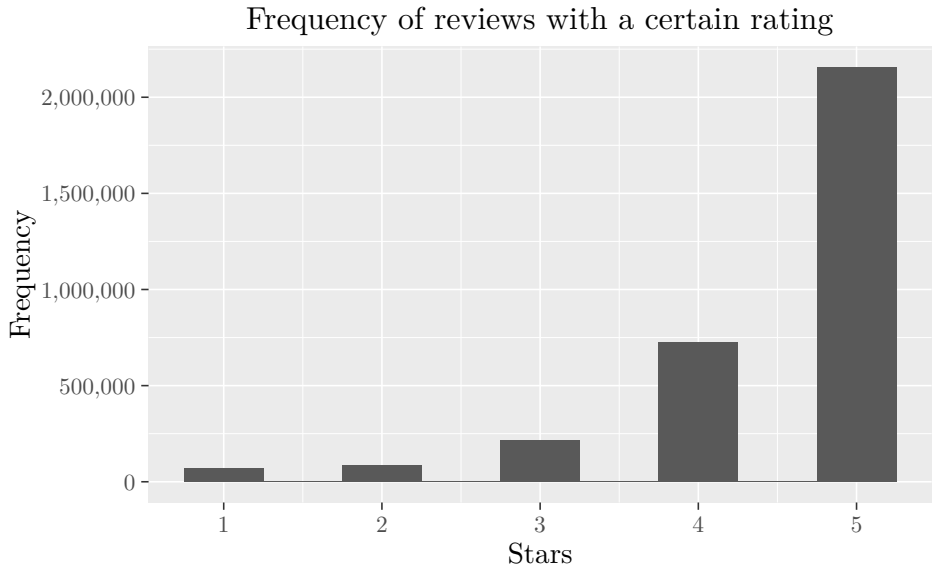

Figure 3.2: From the plot it can be seen that the vast majority of reviews give the highest rating to the recipe, which might suggest that users might be more likely to review a recipe if they are positive about it.

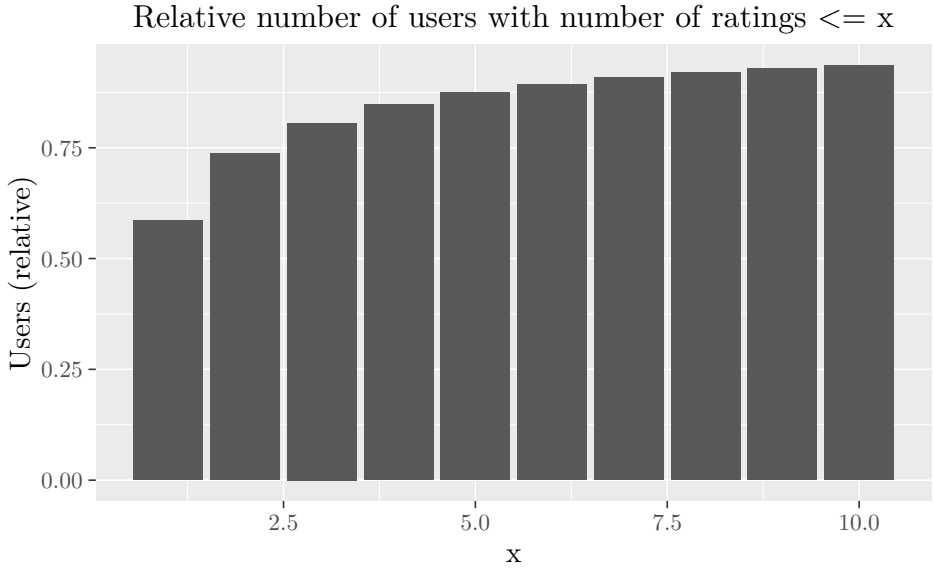

Figure 3.3: The number of people with only a few ratings is large, though this is rather common. Approximately 58 % of the users only rated one recipe, while 94 % of users rated ≤ 10 recipes.

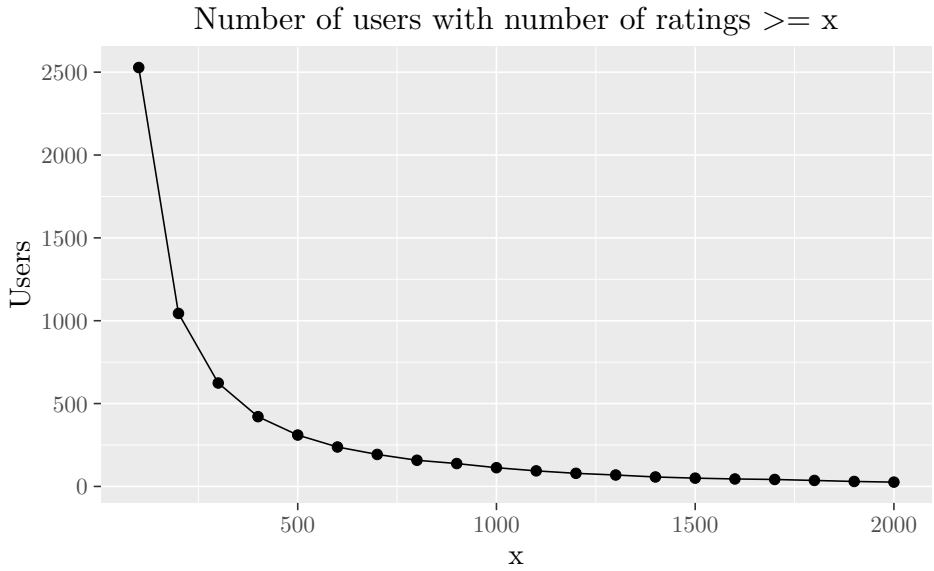

Figure 3.4: There are a few users that submitted many ratings, but only 113 users rated more than 1000 recipes. One user even submitted 5250 ratings.

people probably submit their favorite recipes, that are more likely to receive positive feedback. Whatever the reason, the plot shows that a difference between a 4 and 5 star rating is large. One way to deal with this is to convert the rating matrix into a binary matrix $\hat{R}$:

$$\hat{R}_{ij} = \begin{cases} 1, & \text{if } R_{ij} > t \\ 0, & \text{otherwise} \end{cases} \qquad \text{(Binary Rating matrix)}$$

where $t \in S$ is a threshold on the rating value. Another option is to normalize the rating matrix by subtracting the user's bias. This can be achieved, among other ways, by mean-centering the data, i.e., by subtracting the user's average rating. Fig. 3.5 shows the histogram of ratings after applying this transformation. It looks like a mixture of two gaussian distributions, one for positive and one for negative feedback, with a strong peak at the user's average rating. The deviation from the average rating is very small, with deviations > 1 star being rare. The clusters defined by two different distributions are interesting for to analyze in further detail.

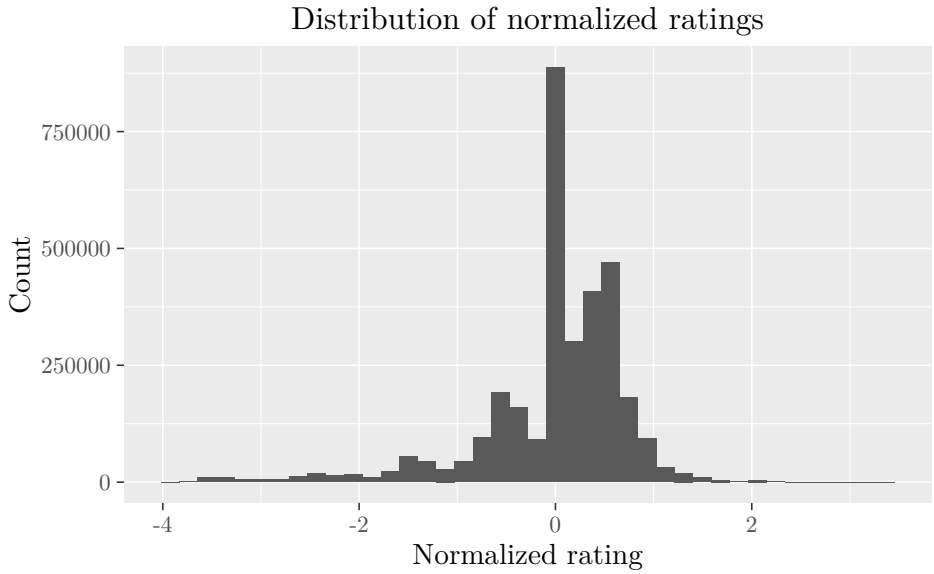

Figure 3.5: Normalizing the user ratings using their average rating reveals that users tend to give most recipes that they review their average rating and a only small proportion a slightly less positive or negative rating (with a bias towards positive).

4 Methodology

This section explains the methods and techniques used in the analysis of the datasets. The two datasets, the ingredient itemsets and user ratings, contain different types of data. Therefore, different techniques are required in order to analyze them. Looking at the ingredient itemsets, here we are particularly interested in combinations of ingredients that frequently occur in the data. The rating data can be seen as a sample of a much larger dataset that contains preferences of a set of people towards a set of recipes. Most of the ratings are however unknown and the main objective here is to approximate these ratings such that they can be studied in relation to the ingredient sets.

4.1 Frequent Itemsets and Association Rules

The ingredient itemsets can be studied using techniques from *Association Rule Mining*, which creates a depedency model from discrete data [1]. It was originally used to mine shopping baskets for frequently occurring patterns, information that can be used for product placement in stores and other

marketing purposes. The problem is formally defined as:

Definition 1 (Association Rule Mining)**.**
Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of n binary attributes called items. Each transaction in $\mathcal{D} = \{t_1, t_2, \ldots, t_m$ be a set of transaction called the *database*. Each transaction in $\mathcal{D}$ has a unique transaction ID and contains a subset of the item in I. A *rule* is defined as an implication of the form $X \to Y$ where $X, Y \subseteq I$ and $X \cap Y =$. The sets of item (for short *itemsets*) X and Y are called *antecedent* (left-hand-side or LHS) and *consequent* (right-hand-side or RHS) of the rule.

Since the number of possible combinations of ingredients, and therefore rules, is extremely large, a number of *interestingness measures* can be used to narrow the search down. One observation is that association rules that only infrequently occur are not interesting. Another is that the number of transaction for which an association rule holds, i.e., $X \to Y$, should be sufficiently large. These two observations are captured in the following measures of interestingness:

Definition 2 (Support)**.**
The *support* of a given itemset X, with respect to a database $\mathcal{D}$, is defined as the proportion of transactions in the database which contains the itemset X, i.e.:

$$\text{supp}(X) = |\{t \mid t \in \mathcal{D}, X \subseteq t\}|$$

with $|\cdot|$ denoting the cardinality of a set.

By setting a threshold on the support of an itemset, it is possible to exclude sets with a low threshold when searching the database, effectively pruning the search space.

Definition 3 (Confidence)**.**
The *confidence* value of a rule, with respect to a database $\mathcal{D}$, is the proportion of the transaction that contain both itemsets X and Y, i.e,:

$$\text{conf}(X \to Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X)}$$

$\text{conf}(X \to Y)$ is defined to be 0 for $\text{supp}(X) = 0$.

The interpretation of the confidence value is that it is an estimate of the probability $P(Y \mid X)$, the probability of finding the antecedent of the rule in transactions under the condition that these transactions also contain the precedent. The confidence value can be used to prune the search space from association rules that infrequently hold. Using the definition of support, we can formally define a frequent dataset as follows:

Definition 4 (Frequent Itemset)**.**
A *frequent itemset* is a set X for which $\text{supp}(X) > t$, where $0 \leq t \leq 1$ is some support threshold value.

If the support threshold is set to a low value, a lot of maximal frequent itemsets are often found. Because of that, computing association rules from these frequent itemsets might still prove intractable. The definition of *support*, however, defines that subsets of frequent itemsets also have to be frequent [10]. This allows us to mine only the *maximal informative itemsets*, since their union contains all frequent itemsets.

Definition 5 (Maximal Frequent Itemset)**.**
A *maximal frequent itemset* X is a frequent itemset that is not a proper subset of another frequent itemset.

There exist a number of association rule mining algorithms, but for this research, the Apriori algorithm [1] will be used. Apriori utilizes the the fact that all subsets of a frequent itemset are also frequent itemsets. The high-level pseudocode of the algorithm is given in Algorithm 1, with most of the details left out, which can be found in the work of [1]. The algorithm uses a bottom-up approach that constructs larger and larger frequent itemsets by combining frequent itemsets found in earlier iterations into a *candidate set* (line 7) and then testing for the minimal support (line 8). The stopping criterion is that the frequent itemset of the previous round should not be empty, which occurs when no itemsets were found that have a sufficiently high support. Computing frequent itemsets can be performed efficiently, because the support for subsets of new candidates have been computed in a previous iteration.

Algorithm 1 Computes maximal frequent itemsets

```
1:  function APRIORI(𝒟, s_min)
2:      C_k                                  ▷ Candidate itemset of size k
3:      L_k                                  ▷ Frequent itemsets of size k
4:      L_1 = {frequent items}
5:      k = 2
6:      while L_{k-1} ≠ ∅ do
7:          C_k := ComputeCandidateSets(L_{k-1})
8:          L_k := ComputeFrequentItemsets(C_k, 𝒟, s_min)
9:          k := k + 1
10:     return ⋃_k L_k
```

Association rules can now be computed from the set of maximal frequent itemsets F by generating from each element all possible rules and

computing their individual confidence scores. A commonly used measure for interestingness of association rules is *lift*.

$$\text{lift}(X \to Y) = \frac{\text{supp}(X \cup U)}{\text{supp}(X) \times \text{supp}(Y)} \tag{4.1}$$

It provides a measure of dependence between the precedent and antecedent. They are completely independent if $\text{lift}(X \to Y) = 1$, while they are dependent if $\text{lift}(X \to Y) > 1$.

4.2 Recommender Systems

The intended purpose of this research is to suggest recipes and ingredients to users. A lot of research has been done on *Recommender Systems* that solve this recommendation problem [17].

Definition 6 (Recommendation Problem)**.**
Given database $\mathcal{D} \in U \times M \times S$ tuples of ratings $s \in S$ that users $u \in U$ have given to items $m \in M$, and an *active user* $u_a \in U$, return a list $(m_1, m_2, \ldots, m_N)$ of items not in $\mathcal{D}$.

This problem can be approached in several ways, but two are the most common:

1. The ratings for user-item pairs not in the database can be approximated from the dataset, after which a top-N ranking for the active user is computed.
2. A top-N list of recommendations can be presented without computing all ratings first (which can be computationally expensive).

Transforming the database into a sparse matrix R with $R_{ij} = 0$ if user u_i did not rate m_j, with usually a low density (see Section 3.2.2). The first method fills in these missing values by collecting preferences of many users in collaboration. This method is also called *Collaborative Filtering* (CF), a general term for a range of series based on the idea of collaboration. Several techniques exist of which a few will be used and compared.

For the second method, association rules can be utilized, which result in a descriptive way of recommending items. The database is mined for rules in the form of "if a user likes items m_p and m_q then he/she also likes item m_r". These patterns are then searched for the user's rated item, after which a recommendation can be presented.

4.2.1 User-based Collaboration Filtering

User-based collaboration filtering computes missing user ratings by assuming that users with a similar preference, their *peers*, will give ratings similar to their peers to unseen items. The database is searched for users that are similar, called the *neighborhood*, with respect to some similarity metric and their ratings are aggregated, for example by taking the mean the neighborhood. We will use two similarity metrics in this study: the *Jaccard similarity* [12] and the *Pearson correlation coefficient* [15].

$$\text{sim}_{\text{Jaccard}}(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{4.2}$$

The Jaccard similarity operates on itemsets, so in order to apply it to input vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, these vectors are translated to their binary counterparts $\boldsymbol{x}^*$ and $\boldsymbol{y}^*$:

$$\boldsymbol{x}_i^* = \begin{cases} 1, & \text{if } x_i > 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

The Jaccard similarity can then be calculated using this binary presentation instead. For this, we first define four functions:

M_{11} total number of positions where $x_i^* = y_i^* = 1$

M_{01} total number of positions where $x_i^* = 0 \wedge y_i^* = 0$

M_{10} total number of positions where $x_i^* = 1 \wedge y_i^* = 0$

M_{00} total number of positions where $x_i^* = y_i^* = 0$

$$\text{sim}_{\text{Jaccard}} = \frac{M_{11}}{M_{01} + M_{10} + M_{11}} \tag{4.4}$$

This shows that the Jaccard similarity can be computed from the rating matrix itself.

Another similarity measure is the *Pearson correlation coefficient*.

$$\text{sim}_{\text{Pearson}} = \frac{\sum_{i \in I}(\boldsymbol{x}_i \boldsymbol{x})(\boldsymbol{y}_i \boldsymbol{y})}{(|I| - 1)\text{sd}(\boldsymbol{x})\text{sd}(\boldsymbol{y})} \tag{4.5}$$

It measures the linear correlation between two vectors of equal length, with a value in $[-1, 1]$ where -1 denotes a total negative correlation and 1 denotes total positive correlation. We use this similarity measure for real-valued rating matrices.

In order to compute the neighborhood of a given user, the entire similarity matrix $S(n \times n)$ has to be computed first, which can be a limiting factor for this approach if the number of users is very large. The user ratings are normalized by mean-centering the data (see Section 3.2.2). Once the similarity matrix has been computed, the users most similar to the active user u_a are extracted by selecting a group of fixed size k, the k-neighborhood $\mathcal{N}_k$. The rating for an item by a user is computed by averaging the rating for this item in the neighborhood:

$$r_{aj} = \frac{1}{|\mathcal{N}_k(a)|} \sum_{i \in \mathcal{N}(a)} r_{ij} \tag{4.6}$$

4.2.2 Item-based Collaboration Filtering

Item-based collaboration filtering [18] is in many ways similar to user-based collaboration filtering. This difference is that instead of computing ratings based on how similar users rated the item, the computed rating is based on ratings the user has given to similar items. As an example we might consider a group of meals that are similar with respect to their ingredients, such as 'Italian pastas with tomato sauce'. Even though the ingredients of the recipes are unknown to the recommender system, users have often given similar ratings to items within this group, revealing item similarity through the preferences of the user group.

In order to decide which items are similar to one another, the similarity matrix S $(m \times m)$ is first computed. Since the approach is essentially a k-nearest neighbor model applied to the columns, instead of the rows, of the rating matrix, only a pre-specified k number of most similar items are stored in the model, meaning that S can be a sparse matrix. This greatly reduces the size of the model, improving the space and time complexity, be it at the cost of at the potential sacrifice of the system [18].

The same similarity metrics that apply to user-based collaborative filtering can also be used in this case, e.g. the Pearson correlation coefficient and the Jaccard similarity. The latter assumes a binary rating matrix in which unrated items are assumed to be 0. Similar to user-based collaboration filtering, the set $\mathcal{S}(i)$ denotes the set with items similar to item m_i. Let

$\mathcal{R}(a) = \{l \mid r_{al} \neq ?\}$ be the set of items that the active user has rated. One way of predicting the rating of item m_i is to take the average of the items that are similar to it and were rated by the user, i.e., to take the mean of ratings by u_a to items in $\mathcal{S}(i) \cap \mathcal{R}(a)$.

$$r_{ai} = \frac{1}{\sum_{j \in \mathcal{S}(i) \cap \mathcal{R}(a)} s_{ij}} \sum_{j \in \mathcal{S}(i) \cap \mathcal{R}(a)} s_{ij} r_{aj} \tag{4.7}$$

4.2.3 Association Rule-based Recommendation for Binary Data

If the rating matrix is a binary matrix, the rows and columns can be seen as sets. The ratings can be thresholded in order to obtain a matrix in which a 1 indicates that a user is likes a certain item. Viewed in this way, association rule mining can be applied to this matrix to create a dependency model of items. These items can be recipes in the case of a binarized rating matrix *users* $\times$ *recipes*, or ingredients, in the case of a *recipe* $\times$ *ingredient* matrix. In both cases, the rating matrix is taken as input to the Apriori algorithm (Algorithm 1), treating the rows of the matrix as transactions, along with chosen support and confidence thresholds. The itemset size is also limited in order to reduce the computational complexity of the recommendation step. The result is a set of association rules $\mathcal{A}$.

To recommend an item to the active user u_a a set of association rules $A \in \mathcal{A}$ is first searched for that have a precedent in the itemset $\mathcal{T}_a$ (the transaction) of the active user and a single-item antecedent, i.e., $A = \{X \to Y \mid X \subseteq \mathcal{T}_a, |\mathcal{Y}| = 1\}$. These rules are sorted according to an interestingness measure, such as confidence, after which the top-N elements are recommended.

5 Experiments

This section describes the experiments that were performed, using the *R* statistical analysis tool [16], on the datasets to obtain a better understanding of what makes a good recipe. First, the itemsets of ingredients are mined for general patterns using the association rule mining capabilities of the *arules* R package [10]. Secondly, a recommender system was built using the *recommenderlab* [9] R package, based the user rating data.

5.1 Association Rule Mining

This section covers the analysis of the ingredient dataset using association rule mining. Using the Apriori algorithm [1], we mine the dataset for association rules with a confidence ≥ 0.5 and a coverage ≥ 0.02, resulting in a set of 1003 association rules. Table 5.1 shows the association rules that have the highest lift. They describe patterns commonly found in sweet, oven-baked dishes, such as cookies and pies. Nutmeg and cinnamon are often used in combination with clover in cookie mixes (e.g. in 'speculaas'), so it is not strange to find the pattern {nutmeg} $\rightarrow$ {cinnamon}. The list is however dominated by vanilla as the antecedent and very similar rules. This could be caused by a strong bias towards these type of dishes in the dataset, because this would result in a high coverage, which is one of the interestingness measures the rules are pruned on. Looking more closely at the recipes, this does not seem the case, however, as depicted before in Fig. 3.1. Many recipes include ingredients such as pepper, onions and cheese, which are normally not used in deserts. This result shows that the ingredients used in these meals are more predictable, in that they are often used in a similar way.

Fig. 5.1 shows a scatter plot of the found association rules in three dimensions of interestingness: support, confidence and lift. It can be seen that many of the rules have a low support value, with only few elements appearing many times. Unfortunately, the rules that have a high lift value have a relatively low confidence value. This means that some rules were found for which the consequent has a much higher support than would be expected from the antecedent alone. However, the cases in which these rules hold are limited, making them less interesting.

Table 5.1: Top 10 of association rules ordered by lift.

Rule	**Sup.**	**Conf.**	**Lift**
{nutmeg} $\rightarrow$ {cinnamon}	0.024	0.58	6.23
{chocolate, eggs, sugar} $\rightarrow$ {vanilla}	0.024	0.68	5.04
{chocolate, flour, sugar} $\rightarrow$ {vanilla}	0.020	0.68	5.00
{chocolate, flour} $\rightarrow$ {vanilla}	0.021	0.67	4.94
{chocolate, salt} $\rightarrow$ {vanilla}	0.020	0.66	4.89
{chocolate, eggs} $\rightarrow$ {vanilla}	0.025	0.64	4.76
{butter, eggs, flour, salt, sugar} $\rightarrow$ {vanilla}	0.029	0.61	4.54
{butter, eggs, flour, sugar} $\rightarrow$ {vanilla}	0.042	0.61	4.51
{butter, eggs, salt, sugar} $\rightarrow$ {vanilla}	0.031	0.61	4.49
{butter, eggs, milk, sugar} $\rightarrow$ {vanilla}	0.022	0.60	4.45

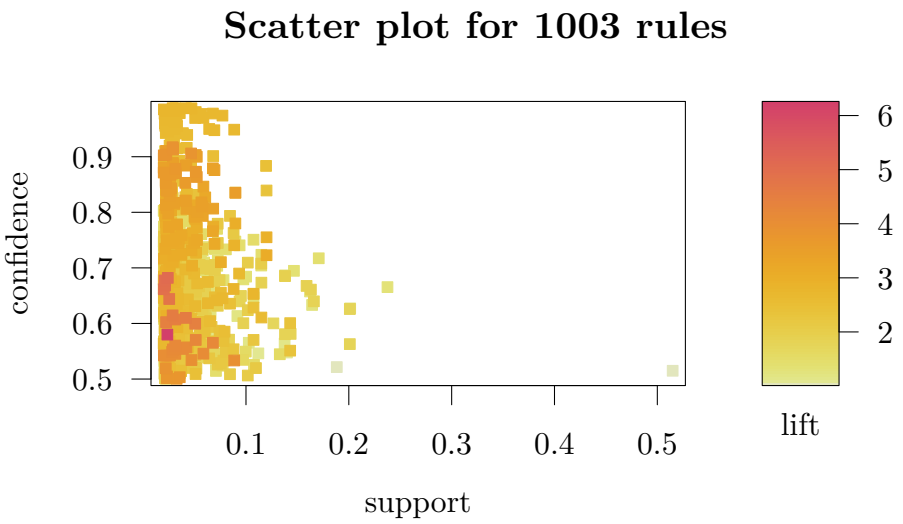

Figure 5.1: The association rules with a relatively high lift are most interesting, but they have low support in this dataset. On the other hand, rules with a high confidence are quite abundant.

5.2 Rating Prediction using UCBF

In this section we look at how the user rating data can be used to build a user-based recommender system that is able to predict user ratings for unseen recipes. As described in Section 3.2.2, the dataset derived from Allrecipes consists of over 3 million user ratings from 1 to 5 stars. The rating matrix is very sparse and the objective is to complete the matrix using the collaborative filtering techniques described in Section 5.2. First, only a part of the rating matrix was selected for modeling by setting a minimum of 10 ratings per user and 10 ratings per recipe, resulting in a $53\,569 \times 49\,463$ matrix of $1\,904\,960$ ratings. Eight different models were built on 90 % of the data, using 10 % for evaluation. Per user, the recipe ratings to be left out were selected using a Given-5 schema, meaning that 5 ratings were randomly selected and used in the training while the remaining ones (≥ 5) are to be predicted. Parameter settings for the UCBF system are the distance metric and the size of the user neighborhood k. The similarity metrics used are Jaccard similarity and the Pearson correlation coefficient (see Section 4.2.1). The neighborhood sizes were chosen to be 10, 20 and 30, resulting in 6 different models, built on the same training data.

Two additional models were added for comparison to these more sophis-

ticated models: a random recommender and global popularity model. The random recommender takes random items from the 'model', which is just the training set itself. It then averages their ratings of these items in order to predict ratings. This recommender is used because of the rating bias discussed in Section 3.2.2. It sets the upper error boundary if at least some knowledge about the rating behavior is known, as its predictions are based on the global distribution of ratings. The other baseline model bases its recommendations on the popularity of items by counting how often items are rated. Its rating predictions, however, are computed by taking the average ratings of items, disregarding items known by the active user and omitting missing values. The prediction error is defined as the difference between the predicted rating $\hat{r_{ij}}$ and actual rating r_{ij}, i.e., $e_{ij} = \hat{r_{ij}} - r_{ij}$. The models are compared using three risk functions: *mean absolute error* (MAE), *mean squared error* and *root-mean-square error*.

$$\text{MAE} = \frac{1}{n \cdot m} \sum_{i=1}^{n} \sum_{j=1}^{m} |e_{ij}| \tag{5.1}$$

$$\text{MSE} = \frac{1}{n \cdot m} \sum_{i=1}^{n} \sum_{j=1}^{m} (e_{ij})^2 \tag{5.2}$$

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{m} (e_{ij})^2}{n \cdot m}} \tag{5.3}$$

Fig. 5.2 shows the prediction errors of the produced models. Apart from the random item model, the other models have similar performances. No model scores best on all three error functions, so the best model is decided on the preferred error function, although the difference is not significant. The models perform reasonably well, with a mean absolute error of about 0.6, compared to 0.7 for the random item model.

5.3 Recipe Completion

We now look how recommender systems can suggest ingredients to complete a recipe. Here we compare item-based collaboration filtering (see Section 4.2.1) and association rule-based recommendation (see Section 4.2.3), along with two baseline models. For the association rule-based models, the confidence is used as a measure of interestingness and the support threshold is varied. A lower support threshold results in less association rules being pruned, resulting in a larger amount of association rules and increasing the space

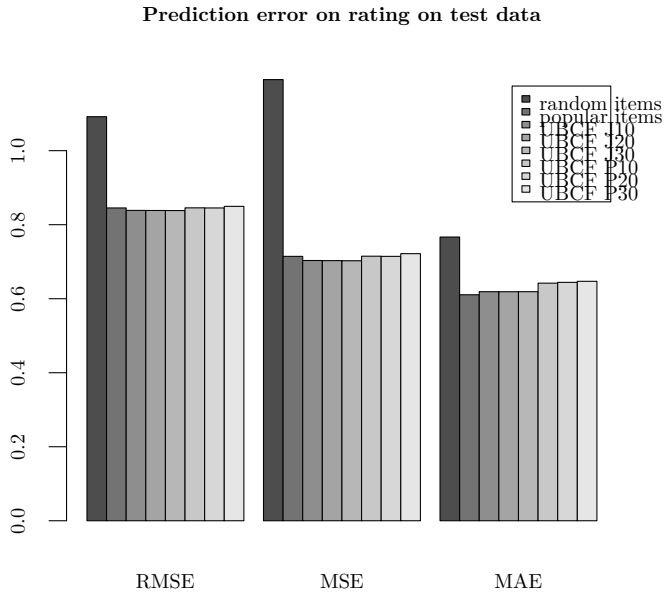

Figure 5.2: The produced models have almost almost the same prediction errors with no significant difference. The model names are follows by an abbreviation of the similarity metric used (J for Jaccard and P for Pearson correlation coefficient) and the neighborhood size k.

complexity of the model and the time complexity of making predictions. By varying the threshold, we can study the impact of this trade-off. A Given-2 evaluation scheme is used, meaning that for each recipe 2 ingredients are given and the other recipes are to be retrieved. Only recipes with at least five ingredients are selected, resulting in a $71\,184 \times 403$ matrix with 636 654 ones. Again, 90 % of the data is used for training and 10 % is held back for testing. Predictions consist of a four top-N rankings, where $N = \{1, 3, 5, 10\}$. The ingredients to be predicted are the ones not given during training, which is a set of variable size, i.e., these ingredients have no specific ordering.

True Positive Item appears in the ranking and is indeed part of the recipe.

False Positive Item appears in the ranking, but is not a part of the recipe.

True Negative Item does not appear in the ranking and is indeed not part of the recipe.

False Negative Item does not appear in the ranking, but it is actually part of the recipe.

Definition 7 (Precision)**.**
Precision, E_p, is the fraction of pairs that are correctly classified as true matches, i.e.,

$$E_p = \frac{|\mathrm{TP}|}{|\mathrm{TP}| + |\mathrm{FP}|}$$

Definition 8 (Recall)**.**
Recall, E_r, is the fraction of true matches that are detected by the system, i.e.,

$$E_r = \frac{|\mathrm{TP}|}{|\mathrm{TP}| + |\mathrm{FN}|}$$

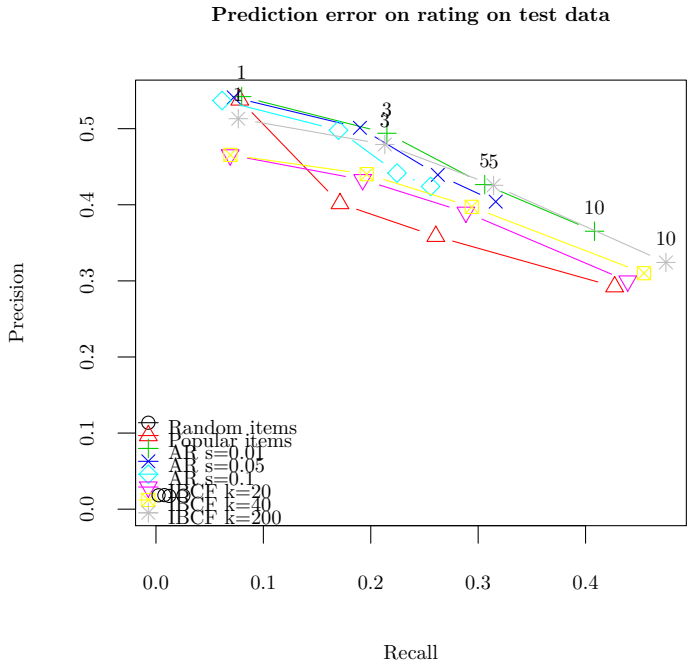

Figure 5.3: The two more sophisticated models, based on association rules (AR) and item-based collaboration filtering (IBCF), outperform the baseline random and popularity based models. A lower support threshold results in a larger model and a better performance of the AR model. The IBCF model performs better with a large neighborhood; the setting $k = 200$ uses a similar item group as big as half the number of known ingredients.

The method are evaluated by means of a *Precision-Recall plot*, which helps in quantifying a model's ability in retrieving correct elements (recall) and

the ratio of correct to incorrect elements (precision). From the plot shown in Fig. 5.3 it can be seen that the item-based collaborative filtering techniques with large item similarity group sizes ($k \geq 40$) perform the best. The item-based method performs surprisingly well considering that for $k = 200$, a group half the size of the ingredient database is used. The association rule-based method performs well for low support thresholds. This is due to the fact that results in a larger model, i.e., more association rules. It seems that the created models are not suitable for true recipe completion, because the precision and recall are not high enough for that. However, the results are good enough that a top-10 ranking could serve as inspiration for the cook. On the other hand, the simplistic popularity-based model is probably good enough to do that as well and its time complexity for both modelling and prediction are much smaller.

6 Conclusions

For this research, a large dataset of tens of thousands of recipes and millions of ratings was extracted from a social networking website. The ingredients were then standardized and joined with another dataset, resulting in one of the largest publicly available[1] datasets on the web. Enriching the data with chemical informations opens up the possibility of studying recipes from another perspective, though this has not been pursued in this study and was left for future work. Two different experiments were conducted on the data. First, a recommender system was built on the rating data using several different approaches. Even though these models were able to predict user ratings within a reasonable error range, the more complex models – based on user-based collaboration filtering – did not perform significantly better than the naive baseline model. Secondly, association rule mining and item-based collaboration filtering were tried as methods to complete recipes when two ingredients were given. In this case, the more complex models did perform better than the naive baseline models, but the system needs to be improved a lot before it can be used in a real scenario. Even though these models were able to retrieve some of the relevant ingredients, the quality of the results indicate that we were not able to find major patterns in the composition of recipes.

[1] The recipe ingredient and rating datasets are freely available, along with the used Python scripts and LaTeX source of this paper, at `https://github.com/benjaminvdb/recipes`.

References

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. "Mining Association Rules Between Sets of Items in Large Databases." In: *SIGMOD Rec.* 22.2 (1993), pp. 207–216.

[2] Yong-Yeol Ahn et al. "Flavor network and the principles of food pairing." In: *Scientific Reports* 1 (2011), p. 196.

[3] *Allrecipes – Food, friends, and recipe inspiration.* URL: `http://www.foodpairing.com` (visited on 08/02/2016).

[4] Marlies De Clercq et al. "Data-driven recipe completion using machine learning methods." In: *Trends in Food Science & Technology* 49 (2016), pp. 1–13.

[5] DataHub. *DataHub – Recipe Dataset.* URL: `https://datahub.io/dataset/recipe-dataset` (visited on 08/02/2016).

[6] *Food Network – Easy Recipes, Healthy Eating Ideas and Chef Recipe Videos.* URL: `http://www.foodnetwork.com` (visited on 08/02/2016).

[7] *Food Pairing – start discover exciting pairings.* URL: `http://www.foodpairing.com` (visited on 08/02/2016).

[8] *FooDB.* URL: `http://www.foodb.ca` (visited on 08/02/2016).

[9] Michael Hahsler. *recommenderlab: A Framework for Developing and Testing Recommendation Algorithms.* 2011.

[10] Michael Hahsler, Bettina Grün, and Kurt Hornik. *Introduction to arules–mining association rules and frequent item sets.* 4. 2007.

[11] George W. Hart. *The Incompatible Food Triad.* URL: `http://www.georgehart.com/triad.html` (visited on 08/02/2016).

[12] Paul Jaccard. "The distribution of the flora in the alpine zone." In: *New Phytologist* 11.2 (1912), pp. 37–50.

[13] A. Jain, R. N K, and G. Bagler. "Spices form the basis of food pairing in Indian cuisine." In: *ArXiv e-prints* (2015). arXiv: `1502.03815 [physics.soc-ph]`.

[14] Maurits de Klepper. "Food Pairing Theory: A European Fad." In: *Gastronomica: The Journal of Critical Food Studies* 11.4 (2011), pp. 55–58. eprint: `http://gcfs.ucpress.edu/content/11/4/55.full.pdf`.

[15] K. Pearson. "Note on Regression and Inheritance in the Case of Two Parents." In: *Proceedings of the Royal Society of London Series I* 58 (1895), pp. 240–242.

[16] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing. Vienna, Austria, 2016.

[17] Francesco Ricci et al. *Recommender Systems Handbook*. 1st. New York, NY, USA: Springer-Verlag New York, Inc., 2010.

[18] Badrul Sarwar et al. "Item-based Collaborative Filtering Recommendation Algorithms." In: *Proceedings of the 10th International Conference on World Wide Web*. WWW '01. Hong Kong, Hong Kong: ACM, 2001, pp. 285–295.

[19] F. Westad, M. Hersleth, and P. Lea. "Strategies for consumer segmentation with applications on preference data." In: *Food Quality and Preference* 15.7–8 (2004). Fifth Rose Marie Pangborn Sensory Science Symposium, pp. 681–687.

[20] Yummly. *Kaggle – What's cooking?* URL: `https://www.kaggle.com/c/whats-cooking` (visited on 08/02/2016).

[21] Mélanie Zetlaoui et al. "Extraction of Food Consumption Systems by Nonnegative Matrix Factorization (NMF) for the Assessment of Food Choices." In: *Biometrics* 67.4 (2011), pp. 1647–1658.