Trends in Food Science & Technology 49 (2016) 1–13

Contents lists available at ScienceDirect

Trends in Food Science & Technology

journal homepage: http://www.journals.elsevier.com/trends-in-food-science-and-technology

Review

Data-driven recipe completion using machine learning methods

Marlies De Clercq*, Michiel Stock, Bernard De Baets, Willem Waegeman

KERMIT, Department of Mathematical Modelling, Statistics and Bioinformatics, Ghent University, Coupure Links 653, 9000, Ghent, Belgium

ARTICLE INFO

Article history:
Received 23 January 2015
Received in revised form
8 September 2015
Accepted 30 November 2015
Available online 15 December 2015

Keywords:
Ingredient combinations
Recipe completion
Non-negative matrix factorization
Two-step regularized least squares
Recommender systems

ABSTRACT

Background: Completing recipes is a non-trivial task, as the success of ingredient combinations depends on a multitude of factors such as taste, smell and texture.
Scope and approach: In this article, we illustrate that machine learning methods can be applied for this purpose. Non-negative matrix factorization and two-step regularized least squares are presented as two alternative methods and their ability to build models to complete recipes is evaluated. The former method exploits information captured in existing recipes to complete a recipe, while the latter one is able to also incorporate information on flavor profiles of ingredients. The performance of the resulting models is evaluated on real-life data.
Key findings and conclusions: The two machine learning methods can be used to build models to complete a recipe. Both models are able to retrieve an eliminated ingredient from a recipe and the two-step RLS model is also capable of completing an ingredient set to create a complete recipe. By applying machine learning methods on existing recipes, it is not necessary to model the complexity of good ingredient combinations to be able to complete a recipe.

1. Introduction

Due to the increasing availability of data in online databases, data mining and machine learning methods are starting to play a prominent role in food consumption analytics and food preference modelling. Traditional methods such as clustering and classification methods are commonly applied to identify consumer patterns and consumer preferences – see e.g. Murray and Delahunty (2000); Westad, Hersleth, and Lea (2004); Bahamonde, Díez, Quevedo, Luaces, and del Coz (2007). More recent studies have introduced the use of more sophisticated methods, such as matrix factorization techniques and graph-based analysis tools, in order to identify consumer patterns based on surveys and check the theory of food pairing, amongst others – see. e.g. Guimet, Boqué, and Ferré (2006); Young, Fogel, and Hawkins (2006); Ahn, Ahnert, Bagrow, and Barabasi (2011); Zetlaoui, Feinberg, Verger, and Clémençon (2011). Moreover, the introduction of specialized machine learning methods in food consumer analytics has resulted in the development of a number of recipe recommender systems.

For example, Sobecki, Babiak, and Slanina (2006) developed a hybrid web-based recommender system, which can recommend certain recipes to different types of users. The system uses fuzzy reasoning techniques and bases its predictions on information contained in user profiles. Freyne and Berkovsky (2010) designed a personalized recipe recommender system that returns healthy recipes, with the aim of educating users in adopting and maintaining a healthier lifestyle. They compose recommendations by gathering ratings on individual ingredients, and combining those ratings into recipe scores, such that unknown ingredient and recipe scores can be predicted for a particular user. Forbes and Zhu (2011) showed that matrix factorization methods can yield a satisfactory predictive power for recipe recommendation, by connecting users with similar taste. The model follows a two-step approach and needs two inputs: a matrix containing the scores given to each recipe and a binary matrix containing the recipes and their ingredients. As a last example, van Pinxteren, Geleijnse, and Kamsteeg (2011) designed a recipe recommender system with the aim to deliver healthier recipes that fit into the daily routine of users. The authors constructed a measure that determines the similarity between recipes following a user-centered approach.

In the present article, we tackle the related problem of designing a recommender system for preparing a meal using some leftover ingredients in a refrigerator. One often has the feeling that by adding only one or two additional ingredients, a splendid dish

* Corresponding author.
E-mail addresses: mm.declercq@ugent.be (M. De Clercq), michiel.stock@ugent.be (M. Stock), bernard.debaets@ugent.be (B. De Baets), willem.waegeman@ugent.be (W. Waegeman).

http://dx.doi.org/10.1016/j.tifs.2015.11.010
0924-2244/

could be created. The problem is deciding which ingredients to add to the grocery list to complete this dish. Solving this problem is a non-trivial task, since there is a multitude of factors that influence the flavor perception of an ingredient combination. The two most obvious factors are smell (Small and Prescott (2005); Smith and Margolskee (2006)) and taste (Prescott (2004); Stevenson and Boakes (2004); Moyer (2013)). Another factor is texture. The sweetness of honey, for instance, is determined by its viscosity (Cook, Hollowood, Linforth, and Taylor (2003); Auvray and Spence (2008)). A well-known example of how temperature can influence the flavor perception is the bitter taste of warm beer (Talavera, Ninomiya, Winkel, Voets, and Nilius (2007); Bakalar (2012); Bajec, Pickering, and DeCourville (2012)). A fifth influencing factor is color. Studies have shown that a more intense color leads to a higher flavor intensity (Morrot, Brochet, and Dubourdieu (2001); Zampini, Sanabria, Phillips, and Spence (2007)). Also the sound of foodstuffs can have an influence. Two examples are the crispness of potato chips and the snap of chocolate (Auvray and Spence (2008); Stevenson (2009)). These factors do not only influence the flavor perception, but also influence each other.

In spite of the difficulty of the task, already several solutions to the problem are available. A first solution is to look in a cook book or database for recipes that contain the remaining foodstuffs. As this activity can be quite time consuming, online search engines exist for retrieving such information automatically. These engines are connected to online databases that contain a large number of recipes. When given some ingredient names, they explore their database for recipes containing one or several of the given ingredients mentioned by the user. Some examples of search engines that work in that way are *supercook.com*, *myfridgefood.com* and *recipematcher.com*.

A second solution to find the desired additional ingredients is to use models for ingredient combinations. One example is the online model of Foodpairing® that suggests, for one ingredient, those ingredients that create tasteful combinations with it. The model is based on the theory of food pairing, which states that a good ingredient combination can be achieved when the combined ingredients have the same major flavor components. The model is created based on scientific flavor analysis of foodstuffs. Once all the flavor components of a foodstuff are analyzed, the major flavor components are identified. These are the components that will determine the smell of a foodstuff. These components are taken into account in the Foodpairing® model (Sense for Taste (2014)). The output of the model is visualized as a tree, with the ingredients of interest in the middle and the ingredients with similar flavors arranged around it. The closer an ingredient is to the center of the tree, the higher the number of shared major flavor components.

The above methodologies have a number of disadvantages. Existing methods based on searching in cook books and online databases are restricted to recommending existing recipes, whereas methods based on flavor components are likely to recommend ingredients with very similar flavors to one of the ingredients present and do not take into account other ingredients that are present in the recipe. As such, neither one of these two approaches can answer recipe completion questions such as “which type of meat can best be combined with all remaining vegetables” or “which herbs can best be used to spice up a meal”. We present two new methods to build models that can address questions of that kind. The resulting models return, for a given set of ingredients, those ingredients that can best be combined with all the given ingredients.

Our solutions are based on data mining and machine learning methods. The first method we investigate is non-negative matrix factorization (Lee and Seung (1999)). This method will be limited to finding ingredient combinations that are already present in existing recipes. More precisely, the model using this method will not be able to find recipes that are not present in the data set, but it will focus on ingredient combinations that are often occurring in existing recipes. As a solution that is able to overcome this problem, two-step regularized least squares is introduced as a second machine learning method used to build a model for recipe completion. This model will not only suggest ingredient combinations based on those found in existing recipes, but also based on the presence of similar flavor components in the different ingredients. This second model will allow to suggest *new* ingredient combinations to complete a recipe.

We present a thorough evaluation and extensive experimental results that confirm the potential of our models. To train, validate and test the models, good data is required. The problem today is not the lack of data (recipe databases, flavor databases, consumption data, food composition data, etc.), but the fact that there is no clear vision on what can be learned from all this data. Next to this problem, there is the fact that not all the available data is equally reliable and useful, while some databases are incomplete or inaccurate. Unfortunately, a data-driven approach is only as good as the data used to learn from. For example, Varshney, Varshney, Wang, and Myers (2013) showed in a study on food pairing that two data sets containing the same type of information can result in conflicting conclusions. This problem could be solved by standardizing databases and by improving the quality of (existing) databases (e.g. Roe and Finglas, (2012)). In our experimental study, we have used the well-known and extensive data set of Ahn et al. (2011), containing data files about recipes, ingredients and flavor components.

The present article is organized as follows. First the data is described and appropriately transformed. In the next section the structure of the models is presented, after which non-negative matrix factorization and two-step regularized least squares are discussed in more detail. The performances of the two models are presented in a following section. Finally some conclusions and thoughts are formulated at the end of the article.

2. Materials

As mentioned before, the data used in the present article are taken from Ahn et al. (2011). This data set consists of five data files covering recipes, ingredients and flavor components. A first data file contains 56,498 recipes originating from eleven different cuisines. For each recipe the accompanying ingredients are listed, as well as the cuisine where the dish originates from. All these recipes together contain 381 different ingredients, ranging from almond to zucchini. Recipes containing less than three ingredients are eliminated in the present article, as these recipes contain no information on ingredient combinations necessary to complete (new) recipes. This reduces the number of recipes to 55,001. A second data file contains the names of 1,530 ingredients, including the 381 ingredients found in the first data file. Each ingredient is uniquely classified into an ingredient category (e.g. fruit, meat, vegetables, etc.). The third data file gives an overview of 1,107 flavor components found in foodstuffs. Each component is provided with its own chemical abstracts service (CAS) registry number. CAS numbers can be used to gather more information about a flavor component, like the chemical structure, synonyms, etc. A fourth data file connects each ingredient with its flavor components. From this data file, it can be concluded that the 381 ingredients, found in the recipe data, contain only 1,021 flavor components. Only these flavor components are hence taken into account. The fifth and final data file contains for 221,777 ingredient pairs, formed with the 1,530 ingredients from the second data file, the number of flavor components the two ingredients have in common. This data file is not

used in the present article.

The mentioned data files are first transformed into practical data sets. The first data file, containing for each recipe the names of the ingredients listed in that recipe, is transformed into a binary matrix Y $(n \times m)$:

$$Y_{ri} = \begin{cases} 1, & \text{if recipe } r \text{ contains ingredient } i; \\ 0, & \text{otherwise.} \end{cases}$$

This results in a sparse matrix with a filling degree of 2.16%. The columns of this matrix represent the 381 ingredients (m) found in the 55,001 recipes (n), which are represented by the rows.

The matrix Y is used to create a new matrix A $(m \times m)$, containing for each couple of ingredients the number of recipes in which these two ingredients are combined. The matrix A is square and symmetric. The 381 different ingredients are represented by both the rows and the columns of matrix A. This matrix is constructed as follows:

$$A = Y^T Y.$$

Finally, the second, third and fourth data files of Ahn et al. (2011) are combined to create a final binary matrix V $(p \times m)$. This matrix represents the 381 ingredients (m) and their 1,021 flavor components (p):

$$V_{ic} = \begin{cases} 1, & \text{if ingredient } i \text{ contains flavor component } c; \\ 0, & \text{otherwise.} \end{cases}$$

The matrix V is a sparse matrix with a filling degree of 5.01%.

3. Methodology

3.1. Model structure

As mentioned above, two methods, non-negative matrix factorization and two-step regularized least squares, will be evaluated in their ability to build a model to complete a recipe. The two models that are built in this paper are linear with a structure of $\vec{y} = \vec{x}M$, where $\vec{x}$ and $\vec{y}$ are two recipe vectors and M is a coefficient matrix. They take an ingredient set as input, i.e. a list of ingredients that are already present in a recipe, for instance, the leftover ingredients in a refrigerator. The ingredient set is transformed into a binary recipe vector $\vec{x}$ $(1 \times m)$, similar to matrix Y but with only one recipe. The recipe vector $\vec{x}$ is multiplied with a *model matrix* M $(m \times m)$ resulting in a new vector $\vec{y}$ $(1 \times m)$. Matrix M will be determined using one of the two machine learning methods, as explained further on. The output vector $\vec{y}$ contains, for each of the m ingredients, a predicted value between 0 and 1. A value close to one means that this ingredient should be added to the recipe to create a tasteful dish. A low value, means that the ingredient is not suitable to complete the recipe, and will result in a poor combination. The predicted values of the ingredients present in the ingredient set are replaced by a negative value, since these ingredients can no longer be added to the recipe, as they are already present. In a next step, the predicted values are ordered from high to low. Now, ingredients with a high value are at the top of this ordered list and the ingredients that were already present are at the bottom of this list. Both models will return the top t of best fitting ingredients to add to the ingredient set to complete the recipe. The structure described above is summarized in Algorithm 1. Once the model has returned the t best fitting ingredients, the user has to decide which of the suggested ingredients he/she will add to complete the recipe.

Algorithm 1 Ingredient Suggester

```
procedure RecipeCompletion(IngredientSet S, top size t).
    x⃗ ← convert S to binary vector                    ▷ O(m)
    y⃗ ← x⃗M                                           ▷ O(m²)
    sort y⃗ (ingredient not in S)                      ▷ O(mlog(m))
    return top t ingredients of y⃗
end procedure
```

3.2. Non-negative matrix factorization

Non-negative matrix factorization (NMF) is a matrix decomposition method that assumes that all data is non-negative (Lee and Seung (1999)). By approximating a matrix by a product of two low-rank matrices, the decomposition method searches for patterns in the data. The NMF method identifies a linear non-negative parts-based representation of the data. A given data matrix Y is approximated by

$$Y \approx WH,$$

where Y is an $(n \times m)$ matrix, W is an $(n \times k)$ matrix containing k latent features of the factorization and H is an $(k \times m)$ matrix also containing k latent features (Lee and Seung (1999)). The parameter k is the rank of the factorization and determines the number of latent features in the factorization. The value of k can be much smaller than n and m, and is restricted by $(n+m)k < nm$. The optimal value of k has to be determined during validation. The methods of training, validation and testing the model are explained in more detail below. The dimensions of W and H are much lower than those of Y. Dimensionality reduction is one of the characteristics of matrix decomposition. The low-rank matrices take less space to store and are easier to handle. All entries of Y, W and H are required to be non-negative. The time complexity of NMF is polynomial, more specifically $\mathcal{O}((nm)^{r^2 2^r})$, with n and m the dimensions of the non-negative matrix (the number of recipes and the number of ingredients) and r the dimension of the low-rank matrices. More information can be found in Arora, Ge, Kannan, and Moitra (2012).

The optimal values of W and H can be found by maximizing the following objective function:

$$L(W,H) = \sum_{i=1}^{m} \sum_{j=1}^{n} \left[Y_{ij} \log(WH)_{ij} - (WH)_{ij} \right], \tag{1}$$

subject to $W_{ia} \geq 0$ and $H_{aj} \geq 0$, for any i, a and j. Equation (1) can be seen as the log-likelihood of generating the data matrix Y from W and H. Maximizing this expression will result in a minimization of the error between Y and WH. A local maximum of this objective function can be found by iterating update rules (Lee and Seung (2001); Hastie, Tibshirani, and Friedman (2008)):

$$W_{ia} \leftarrow W_{ia} \frac{\sum_{j=1}^{n} H_{aj} Y_{ij} / (WH)}{\sum_{j=1}^{n} H_{aj}},$$

$$H_{aj} \leftarrow H_{aj} \frac{\sum_{i=1}^{m} W_{ia} Y_{ij} \Big/ (WH)_{ij}}{\sum_{i=1}^{m} W_{ia}}.$$

Equation (1) is the original objective function used in NMF to determine optimal values of W and H. However, different objective functions can be considered, each resulting in another version of NMF. For instance, one such option to determine optimal values of W and H is to minimize the squared-error function (Lee and Seung (2001); Hoyer (2004)):

$$E(W,H) = \sum_{i=1}^{m} \left(Y_{ij} - (WH)_{ij}\right)^2,$$

subject to $W_{ia} \geq 0$ and $H_{aj} \geq 0$, for any i, a and j. This objective function is more widely used. A local minimum of the squared-error function can be found by iterating:

$$W_{ia} \leftarrow W_{ia} \frac{(YH^T)_{ia}}{(WHH^T)_{ia}},$$

$$H_{aj} \leftarrow H_{aj} \frac{(W^T Y)_{aj}}{(W^T WH)_{aj}}.$$

A major difference between NMF and other matrix decomposition techniques, such as singular value decomposition, independent component analysis and vector quantization, is that NMF only allows linear combinations with positive components. This underlines the part-based characteristic of the method: NMF represents the data by combining different latent features (matrix W). All latent features are used by at least one instance, but not all latent features are used by each instance. This facilitates the interpretation of the representation of the data. It also means that the matrices W and H contain a lot of values that are zero or close to zero, leaving out those parts that are not needed to represent a certain instance. As a result, the matrices are sparse. The part-based representation is the main reason, apart from the non-negative properties of the data, why NMF was selected as matrix decomposition method to build a model to complete recipes and not one of the other matrix decomposition methods. NMF is an ideal method to use on the recipe data, since the data is binary and thus non-negative and a recipe can be seen as a linear combination of ingredients. We empirically found that singular value decomposition and independent component analysis are not suitable for completing recipes.

The NMF method will identify information from the given data and summarize this information in k latent features. The value of k is critical for a good representation of the data. When the value of k is too small, the representation of the data will be insufficient; when the value of k is too large, there is a risk of overfitting the data. A general method to determine the optimal value of k is not yet available. The method that is used in the present article is explained below. A graphical representation of a matrix decomposition method like NMF, applied to recipe data, is shown in Fig. 1.

Once W and H are known for the existing recipes, NMF can be used to complete new recipes. The factorization of the existing recipes can be seen as

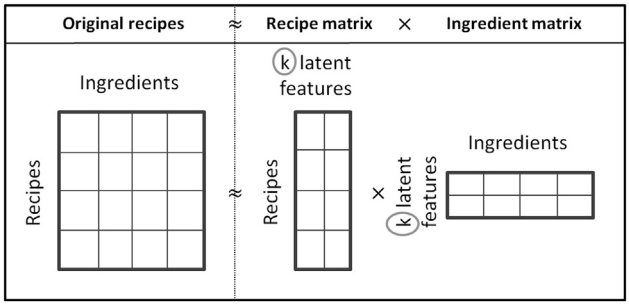


Fig. 1. Graphical representation of matrix decomposition.

$$Y_{\text{old\_recipes}} \approx W_{\text{old\_recipes}} H^T = X_{\text{old\_recipes}} \beta, \quad (2)$$

where $Y_{\text{old\_recipes}}$ and $X_{\text{old\_recipes}}$ are the binary representation of the existing recipes and β is estimated as (Hastie et al. (2008)):

$$\widehat{\beta} = \left(X^T_{\text{old\_recipes}} X_{\text{old\_recipes}}\right)^{-1} X^T_{\text{old\_recipes}} W_{\text{old\_recipes}} H^T. \quad (3)$$

$\widehat{\beta}$ is the model matrix M, needed in Algorithm 1, obtained using NMF. Once matrix M is determined, the model using non-negative matrix factorization can be built using Algorithm 1. This model allows to complete a new recipe by giving the new ingredients and a number of additional ingredients to the model.

3.3. Two-step regularized least squares

In addition to non-negative matrix factorization, we also apply a second machine learning method to build a model for recipe completion, namely two-step regularized least squares (Pahikkala et al., 2014). It is a recent method for making predictions on so-called paired-comparison data such a large sparse matrices of labels. Translated to the problem of recipe completion, this method is not only very appropriate for incorporating the information about ingredient combinations captured in the recipe matrix Y, but also for adding information on flavor components present in ingredients, gathered in matrix V. This difference with NMF makes it possible to implement the theory of food pairing, stating that ingredients containing similar flavor components form a good ingredient combination, which can be used to complete recipes. In essence, the two-step RLS biases the model such that ingredients with similar flavor components as the present ingredients will be recommended to complete the recipe.

The two-step regularized least squares (RLS) method can be seen as a variant and computationally efficient implementation of the pairwise kernel-based framework for learning relations between objects, as discussed in Pahikkala, Airola, Salakoski, De Baets, & Waegeman, (2013); Waegeman et al., (2012). Two-step RLS does not factorize the label matrix Y into two low-rank matrices, but approximates it in a different way:

$$Y \approx K_u W K_v, \quad (4)$$

where K_u and K_v are two kernel matrices, which contain information that can help representing the data in matrix Y. A kernel function in essence yields a similarity score, computed as a dot-product in a high-dimensional feature space. Such a construction projects vectorial data into a higher-dimensional feature space, hence increasing the representation power and generalization ability of linear methods. Different types of kernel functions can be applied to create K_u and K_v. Some examples of popular kernel functions are the linear kernel, the Gaussian radial basis function kernel (ter Haar Romeny (2003)) and the diffusion kernel (Kondor and Afferty (2002); Karatzoglou, Smola, Hornik, and Zeileis (2004)). We refer to Cristianini and Shawe-Taylor (2000) for more details. In the present article, the linear kernel will be applied to construct both kernel matrices in Eq. (4):

$$K_u = X_u X_u^T,$$

$$K_v = X_v X_v^T.$$

In Eq. (4), matrix W is a coefficient matrix, and its optimal values will be determined during the training phase of the model by minimizing the error between Y and $K_u W K_v$. To prevent overfitting, W is estimated as

$$W = (K_u + \lambda_u I)^{-1} Y (K_v + \lambda_v I)^{-1}, \quad (5)$$

using regularization. The optimal values of λ_u and λ_v, the hyperparameters, have to be determined during validation. The methods of training, validation and testing are explained in greater detail below. The time complexity of two-step RLS is polynomial, more specifically $\mathcal{O}(m^3 + m^2 n)$, with m the number of ingredients in the recipes and n the number of recipes. More information on two-step RLS can be found in Pahikkala et al. (2014).

Here, matrix Y in Eq. (4) is the *binary* recipe matrix. K_u and K_v are linear kernels of respectively the recipe data ($X_u = Y$) and the flavor data ($X_v = V$). This means that the recipe matrix is used on both sides of Eq. (4). K_u represents the number of shared ingredients for each pair of recipes, while K_v contains for each pair of ingredients the number of flavor components they have in common. The above is illustrated in Fig. 2(a). The fact that the predicted value of bell pepper in soup is as high as it is, although the original value equals zero, means that it is a good idea to add bell pepper to the soup.

Once W is found for the existing recipes, two-step RLS can be used to complete new recipes. For two-step RLS the model matrix M, needed to build a model using Algorithm 1, is given by

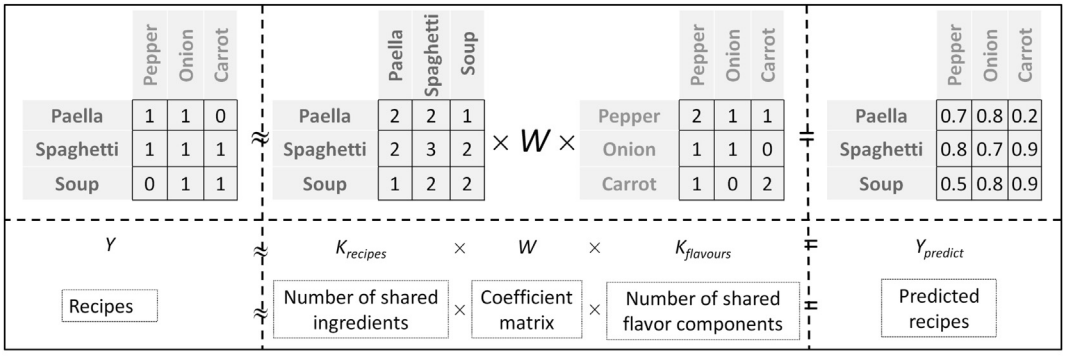
$$M = X_{u,\text{old\_recipes}}^T W K_v, \quad (6)$$

where $X_{u,\text{old\_recipes}}$ is the binary recipe matrix Y. Once matrix M is known for two-step RLS, the model structure presented in Algorithm 1 can be followed to build the two-step RLS model, which can be used to determine additional ingredients to add to a new recipe.

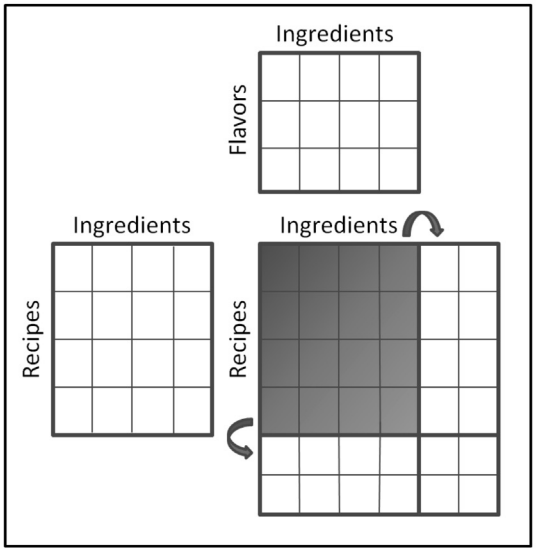
3.4. Performance evaluation

As mentioned before, the recipe data is divided into a train, a validation and a test data set for tuning the hyperparameters and for evaluating the performance of both models in a fair manner. The sizes of the train, validation and test data sets are the same for NMF and two-step RLS. The number of recipes in each set is determined based on k-fold cross-validation. Both models will only use those recipes that are needed in a certain part of the learning process, meaning that the validation and test recipes will not be used during training, nor will the test recipes be used during validation.

During training, validation and testing, the aim is to determine the ability of the models to complete a recipe. This is done by studying whether the models are capable of retrieving an



(a)



(b)

Fig. 2. Illustration of the use of two-step regularized least squares to find additional ingredients for incomplete recipes. Two-step RLS makes it possible to add new ingredients, that are never used in any recipe before, to existing recipes, by adding new ingredients to the flavor data (X_v), or to complete a new recipe using the known ingredients by adding a new recipe to the recipe data (X_u). These two actions can be combined to complete new recipes with (known and) new ingredients. (a). Illustration of the use of two-step RLS for recipe completion. (b) Graphical representation of two-step RLS.

ingredient that was eliminated from a recipe. For this purpose, the validation and test recipes are modified as follows: for each recipe one of the present ingredients is selected at random and eliminated from the recipe. This ingredient can be the same for two recipes, but can also be different. This modification can be seen as a replacement of a *one* with a *zero* in a binary matrix:

$$y_{ri} = 1 \rightarrow y_{ri} = 0,$$

with r one of the validation or test recipes and i the randomly selected ingredient for that recipe. The names, or position in matrix Y, of the eliminated ingredients are saved and will be used to evaluate the performance of the models.

3.5. Performance measures

To be able to evaluate and compare the performance of the two models, performance measures need to be selected. To tune and test the models, the model structure in Algorithm 1 is followed until the ordering of the list of predicted values for each tune or test recipe. It is this ordered list, which has the best fitting ingredients at the top and the ingredients that were already present or do not make a good combination at the bottom, that will be used to evaluate the abilities of the two models. So, during tuning and testing, both models will return for each recipe in the validation or test data set an ordered list of 381 predicted values, where each value corresponds with one of the 381 ingredients. Information on how these values are determined during tuning and testing can be found below. The rank of the eliminated ingredient in the ordered list of best fitting ingredients is determined. If the eliminated ingredient had the highest predicted value of all ingredients that were not yet present in the recipe, its rank is one. In the ideal case, the method should suggest the eliminated ingredient as best fitting ingredient. However, it is also acceptable that the eliminated ingredient is found in the top ten of best fitting ingredients, as there is always a chance that the recipe was not optimal and so the eliminated ingredient is not the best fit. The validation procedure is illustrated in Fig. 3 for the two-step RLS model.

The rank of the eliminated ingredient is used in the performance measures and will serve as a measure of how capable the models are in retrieving an eliminated ingredient and restoring the modified recipe. The performance measures are introduced below and are calculated using all the tune or test recipes.

1. Mean position of the eliminated ingredient in the ordered list of suggested ingredients:

$$\text{mean rank} = \text{mean}(\text{rank of eliminated ingredient}).$$

2. Median position of the eliminated ingredient in the ordered list of suggested ingredients:

$$\text{median rank} = \text{median}(\text{rank of eliminated ingredient}).$$

3. Percentage of test recipes for which the eliminated ingredient is located in the top 10 of suggested ingredients.
4. Mean area under the ROC curve: mean AUC. The ROC curve is found by plotting the probability that a positive prediction is a true positive observation in function of the probability that a positive prediction is a false positive observation. Here, the eliminated ingredient is seen as the only positive observation, while all the other ingredients are seen as negative observations. The AUC is defined as the area under the ROC curve. Since there is only one positive observation, the AUC is directly related to the rank of the eliminated ingredient and can be computed as

$$\text{AUC} = 1 - \left(\frac{\text{rank} - 1}{N^-}\right) = \frac{N^- + 1 - \text{rank}}{N^-},$$

with N^- the number of negative observations and rank the position of the eliminated ingredient in the ordered list of suggested ingredients. The mean AUC is computed as the mean of the AUC obtained for all recipes. The AUC ranges from 0 to 1, where 0 means that all the predicted labels are wrong, 0.5 means that labeling has occurred randomly and 1 means perfect labeling. More information on ROC curves and the AUC can be found in Huang and Ling (2005).

The first two measures should be minimized, while the third and fourth ones should be maximized. These four measures are similar to existing statistical measures like the mean reciprocal rank (Craswell (2009)) or average precision (Kishida (2005)). Either one of these measures could be used to tune the NMF and two-step RLS model. In this article we will use these measures one by one, to find the optimal value of k for the NMF model and the optimal values of λ_u and λ_v for the two-step RLS model. Each time the same measure as the one that was used to validate the models, will be used to evaluate the results of the models during testing. This will yield four values that can be used to compare the two models.

3.6. Non-negative matrix factorization

To determine the optimal value of k, the validation recipes are modified as mentioned above. Once one set is used to validate the trained model, the recipes of the used set are restored to their initial state ($y_{ri} = 0 \rightarrow y_{ri} = 1$) and the next validation set is modified ($y_{ri} = 1 \rightarrow y_{ri} = 0$).

During validation, the value of k is varied from one to 12. For each value of k, matrix Y_{train}, containing the binary representation of the training recipes, is factorized into W and H. Equation (2) can than be seen as

$$Y_{\text{train}} \approx W_{\text{train}} H^T = X_{\text{train}} \beta,$$

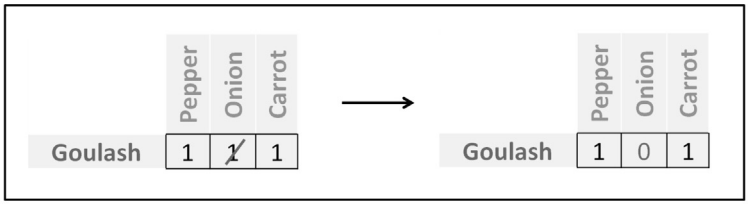
where Y_{train} and X_{train} are the binary recipes in the train set. $\widehat{\beta}$ is estimated using Eq. (3), with $X_{\text{old\_recipe}} = X_{\text{train}}$. As mentioned above, matrix $\widehat{\beta}$ is the model matrix, so predicted values for the tune recipes are obtained as

$$Y_{\text{tune,pred}} = X_{\text{tune}} \widehat{\beta},$$

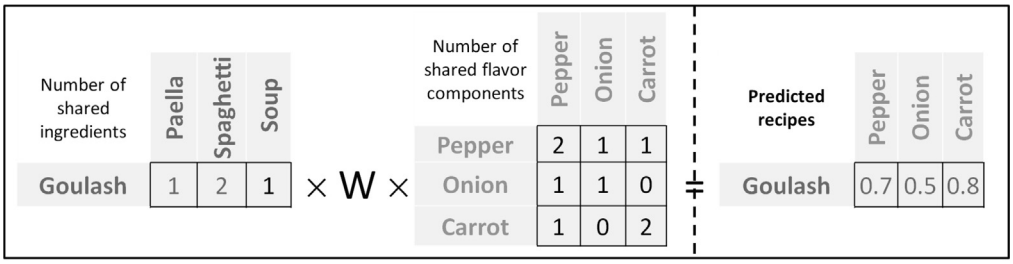
where X_{tune} is the matrix containing the binary recipes in the tune set. For each recipe in $Y_{\text{tune,pred}}$, the predicted values are ordered from high to low, after replacing the predicted value of the ingredients that remained in X_{tune} by zero.

In a first evaluation step, the rank of the eliminated ingredient in the ordered list of best fitting ingredients is determined for each validation recipe in Y_{tune}. The four performance measures are calculated based on these ranks and linked to the value of k that was used to factorize the matrix Y_{train}. After these values are calculated for each value of k in the selected range, the value of k with the optimal value is selected for each performance measure as best value of k for the validation set that was under investigation. Each validation set delivers its own optimal value of k. The median of these optimal values of k is determined and taken as the optimal value of k. The optimal value of k can be different for the different performance measures used during validation of the model.

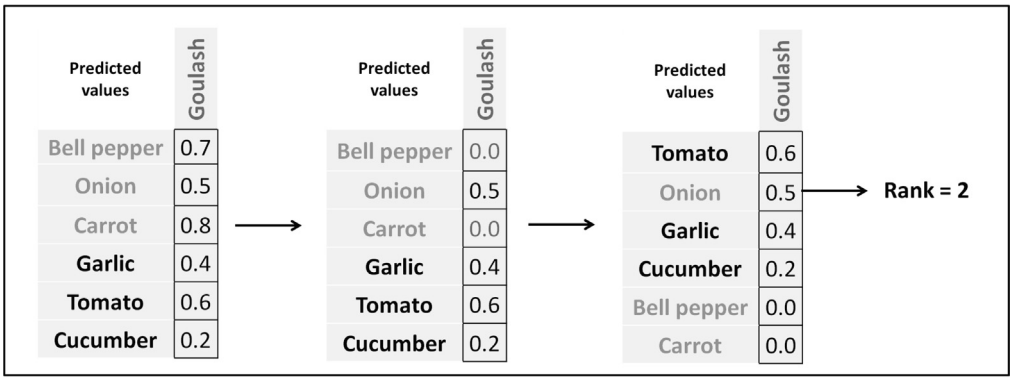
This optimal value of k is then used during testing to perform a final factorization of the matrix $Y_{\text{tune+train}}$ using NMF. Just as before, the model matrix M, needed to build the model, is determined



(a)



(b)



(c)

Fig. 3. During validation and testing one ingredient is removed randomly from the recipes, this can be seen as a replacement of a one with a zero in the binary matrix of the recipes. The aim is to retrieve the eliminated ingredient. The parameter values (λ_u and λ_v) that can best retrieve the eliminated ingredient are the optimal parameter values. (a) Modify validation recipe by removing one randomly selected ingredient for the recipe. (b) The modified recipe is given to the trained model and Ytune is predicted by the model. (c) The predicted values of the validation recipes are ordered from high to low, after replacing the predicted values of the ingredients that were already present in the recipe by zero, as those ingredients can no longer be added. The rank of the eliminated ingredient in the ordered list is determined.

using Eq. (3), with $X_{\text{old\_recipe}} = X_{\text{tune+train}}$. For each recipe in the test data set, the rank of the eliminated ingredient in the ordered list of suggested ingredients is determined. These ranks are saved and will be used to calculate the performance measures for the NMF model.

3.7. Two-step regularized least squares

The train data set is used to build the coefficient matrix W using Eq. (5) with $K_u = X_{u,\text{train}} X_{u,\text{train}}^T$. Different matrices W are built for different values of λ_u and λ_v. During validation, $Y_{\text{tune,pred}}$ is determined using Eq. (4) for each matrix W built during training. In this equation, K_u is the linear kernel of the validation and train data sets, $K_u = X_{u,\text{tune}} X_{u,\text{train}}^T$. To determine the optimal values of λ_u and λ_v, the same method is used as for NMF, i.e. the rank of the eliminated ingredient of each tune recipe is determined.

Each recipe in the validation set delivers its own rank. The four performance measures are calculated for these ranks and used to determine the optimal values of λ_u and λ_v. The values of λ_u and λ_v leading to an optimal value are selected for each performance measure as optimal values of λ_u and λ_v. This is done for each combination of train and validation data sets (cross-validation), resulting in multiple optimal values of λ_u and λ_v. The median of these values, which can be different for the different performance measures, will be used to build a matrix W during testing. This matrix W is built using Eq. (5) with $K_u = X_{u,\text{tune+train}} X_{u,\text{tune+train}}^T$. $Y_{\text{test,pred}}$ is determined using Eq. (4) with $K_u = X_{u,\text{test}} X_{u,\text{tune+train}}^T$. After replacing the predicted values of the remaining ingredients by minus infinity for each recipe in $Y_{\text{test,pred}}$, the rank of the eliminated ingredient in the ordered list of suggested ingredients is determined for each test recipe. These ranks are saved and will be used to determine the performance measures for the two-step RLS model.

4. Results and discussion

4.1. Prior analysis

Before fitting the above-mentioned models to the data, canonical correlation analysis (CCA) is applied to examine whether the flavor composition of ingredients is related to the ingredient use found in recipes as suggested by the theory of food pairing. CCA tries to find projections of both the recipe data and the flavor data to two new variables. These projections are performed such that the correlation between these two variables across all ingredients is maximized. When the coefficients of both variables are plotted against each other, a linear relation is found.

The graph is shown in Fig. 4. This linear relation means that there exists a relation between the flavor components present in ingredients and the way ingredients are combined in recipes. This shows that adding the flavor data to the two-step RLS model could lead to better recipe completions. The correlation is especially clear for ingredients with a specific flavor, like mussels, cognac, figs, star anise, etc. The correlation is less clear for ingredients that are mainly used as a main nutrient source rather than for their flavor, such as butter, onion, brown rice, egg. All these ingredients are used in a large number of combinations. Butter is not only used on bread, but also to prepare meat and vegetables or to bake cake or cookies. This indicates that the use of butter is not only related to its flavor, but rather to its other properties like structure, melting properties, etc. Another example of an ingredient that is not only used for its flavor is egg. Yolk is known for its emulsifying properties and egg white possesses the ability to create a foam structure. It can be concluded that CCA shows that in many cases the ingredient combinations in recipes depend on the flavor composition of the ingredients, but that flavor is not the only factor that determines the choice of ingredients.

4.2. Non-negative matrix factorization

The median optimal value of k equals 2 or 3, depending on the performance measure that is used during validation. This means that there are approximately two or three latent features needed to represent the recipe data.

The information collected during the test phase is used to determine the performance measures for the NMF model. These are calculated using the ranks of the 21,600 test recipes:

1. mean rank = 33.0,
2. median rank = 12,
3. recipes with rank $\leq$10 [%] = 48.2%,
4. mean AUC = 0.914.

The first two measures imply that on average an eliminated ingredient can be found on the thirty-third (8.7% of approximately 381 ingredients) place in the descending list of best fitting ingredients and fifty percent of the recipes has an eliminated ingredient with a rank smaller than or equal to 12. This is a rather good performance when keeping in mind that predicting ingredient combinations to complete a recipe is complex and depends on taste, smell, etc. The third measure indicates that the eliminated ingredient was found in the top ten of best fitting ingredients for 48.2% of the test recipes. The performance of the model can be judged even better when the percentage of test recipes having an eliminated ingredient with a certain rank is plotted against the possible ranks. This plot is shown for a rank up to fifteen in Fig. 5(a) for the model validated using mean rank as performance measure. The cumulative rank distribution is plotted as well, visualizing the percentage of recipes having a rank smaller than or equal to a selected rank. This plot is shown in Fig. 5(b) for a rank smaller than or equal to fifteen and in Fig. 5(c) for the entire range of ranks. The fact that this curve rather quickly approaches a value close to 1 and then shows a rather horizontal progress confirms the acceptable performance of the NMF model. The AUC is rather close to one, indicating, just like the previous measures, that the model is able to retrieve the eliminated ingredients. This comes as no surprise as the AUC is only depending on the rank.

Some eliminated ingredients have a very high rank, indicating that the model is not able to retrieve these eliminated ingredients. This is not a problem of the NMF model, but can be attributed to the characteristics of the eliminated ingredient and the test recipe in question. Eliminated ingredients with a high rank are often rather rare ingredients, however, this is not the only explanation, as some rare ingredients do have a low rank and some of the eliminated

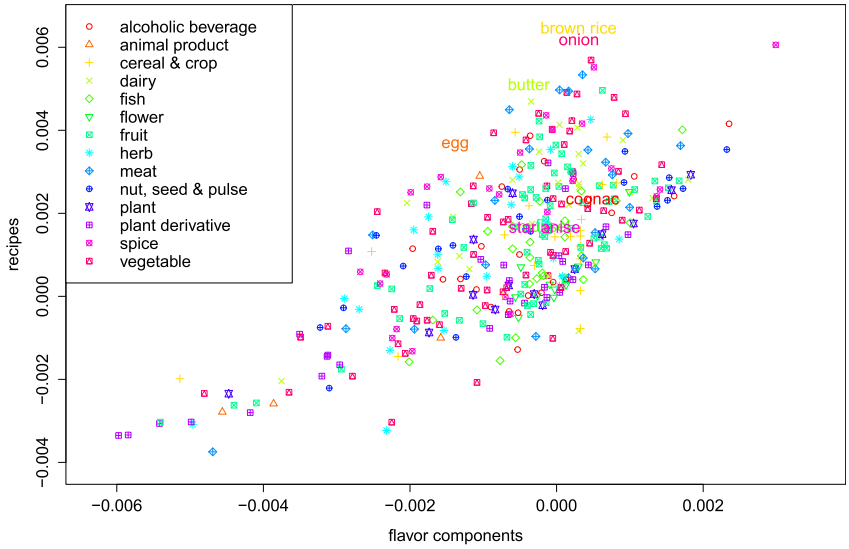
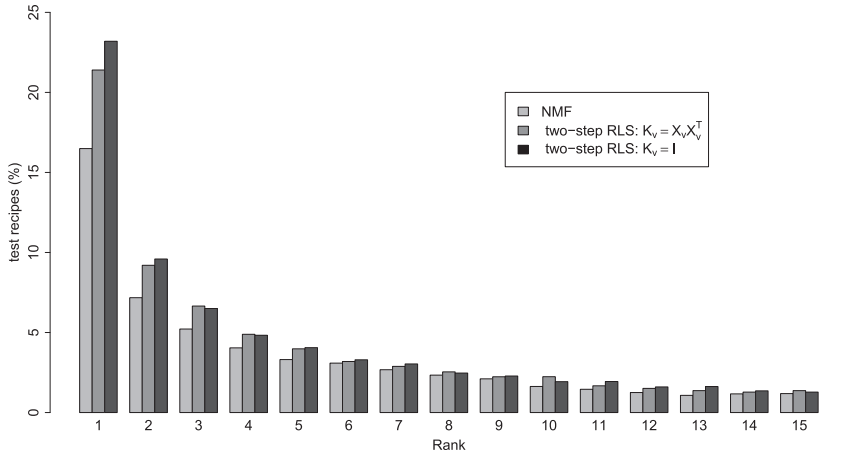
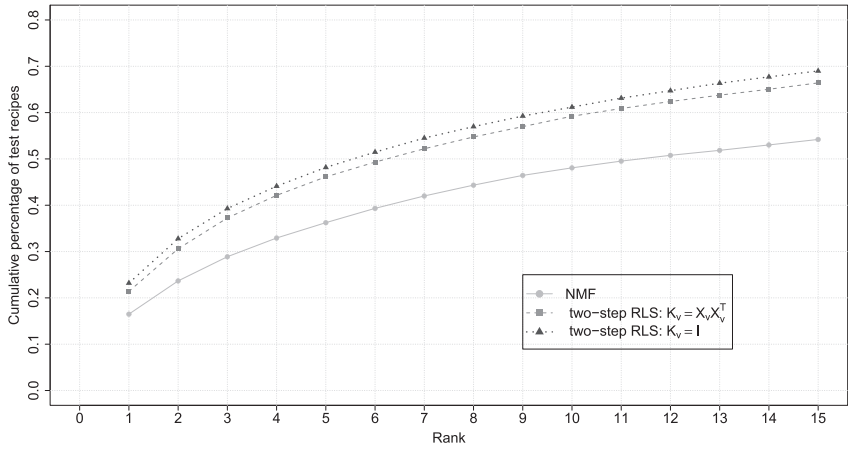


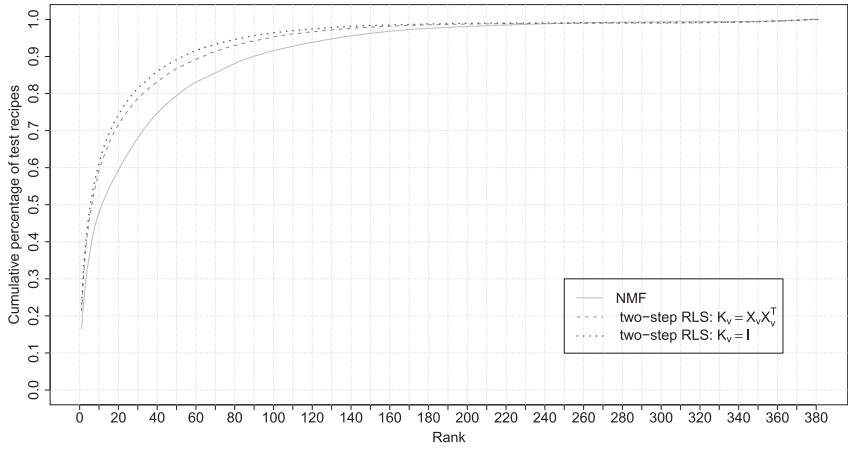
Fig. 4. Canonical correlation analysis shows that the flavor components present in the ingredients are correlated with the way these ingredients are used in recipes. Mussel, cognac, fig and star anise are examples of ingredients with a high correlation. Butter, onion, brown rice and egg on the other hand are ingredients with a low correlation between flavor components present and use in recipes. Each point on the graph represents one of the 381 ingredients.



(a)



(b)



(c)

Fig. 5. Relative and cumulative rank distributions for the NMF model and the two-step RLS models. The former represents the percentage of test recipes for which the eliminated ingredient had a certain rank in the ordered list of best fitting ingredients. All three models are validated using mean rank as performance measure. (a) Relative rank distribution (rank _ 15). (b) Absolute rank distribution (rank _ 15). (c) Absolute rank distribution (rank _ 381).

ingredients with a very high rank are just very common like butter and onion. The main cause is that the eliminated ingredient is usually not combined with the remaining ingredients of the test recipe. In some cases the eliminated ingredient is only combined with the remaining ingredients of the test recipe in that recipe. However, since this recipe is used for testing, it is not used to build the model and the model is not aware of these ingredient combinations. When an ingredient is rarely used in the recipes, this ingredient has a higher chance to end up in the above scenario.

The matrices *H* and *W* of NMF also contain useful information. The *ingredient matrix* (Fig. 1) is examined in greater detail for the model validated using the mean rank as performance measure. In

contrast with singular value decomposition, the k features of NMF are not ordered according to decreasing importance. Therefore, it is not possible to select the first two features to account for the biggest part of variation. The *ingredient matrix* gives a better view on how NMF determines which ingredients to add to complete a recipe. Since the median of the optimal values of k was approximately two, NMF is asked to compress the information present in matrix Y into two latent ingredient features. These features are sparse, which is a characteristic of NMF. The first feature primarily contains ingredients found in main dishes like onion, garlic, tomato, olive oil, black pepper, etc. The second feature contains ingredients that are mostly used for preparing desserts or pastries, like flour (wheat), egg, butter, milk, vanilla, cream, cocoa, etc. From the latent features, it can be concluded that the type of dish, e.g. main dish or dessert, is an important factor during recipe completion.

4.3. Two-step regularized least squares

The ranks of the eliminated ingredients of the test recipes are used to determine the performance measures for the two-step RLS model:

1. mean rank = 23.6,
2. median rank = 7,
3. recipes with rank $\leq$10 [%] = 59.1%,
4. mean AUC = 0.939.

Just as for the NMF model, the percentage of test recipes, having an eliminated ingredient with a certain rank, is plotted against the possible ranks. The relative and cumulative rank distributions for the two-step RLS model, validated using the mean rank as performance measure, are shown in Fig. 5.

When comparing the values of the performance measures of the two-step RLS model with those of the NMF model, it is clear that the two-step RLS model gives better results when applied to complete recipes, no matter what measure was used to validate the models. The same conclusion can be drawn when comparing Fig. 5(a), (b) and (c). The cumulative curve of the two-step RLS model starts off higher and increases faster to 1, meaning that relatively more recipes have a lower rank when using the two-step RLS model to retrieve an eliminated ingredient than when using the NMF model.

4.4. Models in practice

Testing whether a model can retrieve an eliminated ingredient is one way to test the performance of a model to complete a recipe. However, it is also important to look at other ingredients, apart from the eliminated ingredient, that are located at the top of the list with suggested ingredients, as the aim is to make recipes with ingredients in a refrigerator, for which the eliminated ingredients are unknown. This is done in the next experiment. For this experiment the two models are built following Algorithm 1 to the end, in contrast with the previous experiment where the algorithm was stopped once the list of predicted values was ordered. The models are built using the model matrices obtained when validating the previous models using mean rank as performance measure and t will be set equal to 5, meaning that each model will return the five best fitting ingredients when given an ingredient set. Three sets of three leftover ingredients are selected. A first set contains raisin, rum, oatmeal and apple. This set is based on a dessert. The model should ideally suggest ingredients that are related to desserts. A second set contains chicken, rice and cream. This set is based on a main dish. Since there are no vegetables present in this set, it would be nice if the method would suggest some vegetables that could be added. A third and final set contains tomato, beef, oregano and egg, based on Italian spaghetti. These three sets are submitted one by one to the two models.

The top five of suggested ingredients for the three sets can be found in Table 1(a) and 1(b). It is clear that the NMF model suggests ingredients that are commonly used, like wheat and butter, since there are only seven different ingredients in the fifteen ingredients that are suggested. For the two-step RLS model, the suggested ingredients are better, more diverse and recipe specific, but are still not acceptable for the second and third sets. Nobody will add brown rice, when already using rice, or will add raw beef to a spaghetti-like dish, already containing beef. After examining the model matrix, it is clear that these ingredients are suggested based on the number of flavor components they have in common with one of the given ingredients. The number of shared flavor components is too high for these undesired combinations. This could be taken into account during validation. An optimal value of λ_v could be selected not only based on the rank of the eliminated ingredient, but also based on the similarity between the given and suggested ingredients. Another solution is to leave out the flavor components by replacing K_v with the identity matrix I. This last option is selected and a new two-step RLS model is created with $K_v = I$.

4.5. Adapted two-step recursive least squares

The adapted two-step RLS model is trained, validated and tested in the same way as was done before with the first two-step RLS model. The ranks of the eliminated ingredients in 21,600 test recipes are used to calculate the performance measures:

1. mean rank [%] = 20.8,
2. median rank = 6,
3. recipes with rank $\leq$10 [%] = 61.6%,
4. mean AUC = 0.947.

These values are even better with $K_v = I$ than before with $K_v = XX^T$. This is probably because ingredients, present because of the high number of shared flavor components with one of the given ingredients, are no longer present at the top of the list with suggested ingredients.

Matrix M (Eq. (6)) of the second two-step RLS model differs from the first matrix M, not only in K_v, but also in W. The change in matrix W will partially be caused by the changed values of K_v, but also

Table 1
Top five of best fitting ingredients to add to the given set of ingredients to complete the recipe using the NMF model or one of the two-step RLS models, validated using mean rank as performance measure.

Raisin, oatmeal, apple,rum	Chicken, rice, cream	Tomato, beef, oregano,egg
(a) NMF		
wheat	onion	onion
egg	egg	wheat
butter	wheat	garlic
milk	garlic	butter
vanilla	butter	milk
(b) Two-step RLS with $K_v = XX^T$, the linear kernel of the flavor data.		
cinnamon	brown rice	raw beef
walnut	onion	onion
nutmeg	butter	basil
cane molasses	milk	garlic
egg	chicken broth	wheat
(c).Two-step RLS with $K_v = I$, the identity matrix.		
cinnamon	chicken broth	onion
walnut	onion	basil
nutmeg	butter	garlic
cane molasses	milk	wheat
egg	vegetable oil	vegetable oil

because the optimal value of λ_u will be different, resulting in a different matrix W. The same experiment as before, with three recipe sets each containing three ingredients, is performed with the new model matrix M validated using mean rank as performance measure. The top five of suggested ingredients for the three recipe sets can be found in Table 1(c). It is clear that there are no major differences between Table 1(b) and 1(c). However, the undesired ingredients from the first two-step RLS model have disappeared when using the second two-step RLS model. It can be seen that the suggested ingredients are acceptable ingredients to complete the three recipes. The ingredients, suggested to complete the dessert, are ingredients that are usually found in desserts. A possible dish that can be made with the given and suggested ingredients is apple crumble with sabayon. The second ingredient set does not contain any vegetables. The model suggests to add onion to complete the recipe. Besides vegetables, the model also suggests chicken broth. This could be added to the cream to create a sauce to which the onion could be added. The final top five, for the spaghetti-like dish, contains wheat (flour) amongst others. This ingredient could be combined with the given egg to create pasta. The model also recommends onion and garlic. These vegetables could be used in combination with the given tomato to create the sauce to which the oregano and basil can be added to spice up the dish. It can be concluded that the two-step RLS model is not only capable of retrieving an eliminated ingredient, but also of completing a recipe.

The output of the model can be adapted to the preferences of the user. One possibility is to only return ingredients from a certain category, for instance, meat or fruit. This is shown in Table 2 where the model attempts to complete a recipe containing cocoa, vanilla and coconut. These ingredients could lead to a new type of chocolate truffle or could be used to create a cocktail. In the first situation the model is asked to return the three types of nuts that can best be combined with the ingredients already present. The model suggests to add walnut to the filling of the new chocolate truffle. An important component in a cocktail is an alcoholic beverage. The model returns rum as best fit. It is also possible to ask the model to add a spice to the recipe. Coriander appears to be the most ideal spice to complete this recipe.

Restricting the model to only suggest ingredients from a certain category could also solve the problem that occurs when adding the flavor components to the two-step RLS model. For instance, when looking at the second set of ingredients in Table 1, the user could ask the model to only suggest vegetables. In that case, the undesired brown rice would no longer be part of the output of the model, as it is not a vegetable.

Matrix M of the two-step RLS model, validated using the mean rank as performance measure, is studied in greater detail, as this matrix contains information on the reasoning of the model on deciding which ingredients to add. Matrix M is a square matrix with the 381 ingredients presented in both the columns and the rows. The matrix gives for each couple of ingredients a value that reflects how well these ingredients can be combined. Only the second matrix M with $K_v = I$ is studied, as this model gives the best results.

When looking at the values in matrix M, it is clear that the presence of some ingredients will not influence the list of suggested ingredients, as these ingredient instances have a value close to zero for each feature in the matrix. This means that the presence of these ingredients has almost no influence during multiplication of the binary recipe with matrix M and that they will not be suggested to add to a recipe. Some examples of such ingredients are angelica, beech, geranium, holy basil, etc. These ingredients are rare and only used in a very small number of recipes. The presence of the flavor data could increase the importance of these ingredients in the model, as their flavor components would have influence on the suggested list of ingredients. However, as we have seen before, adding the flavor components to the model causes new problems. It can be concluded that the more common an ingredient is, like for instance butter, the higher the values are in matrix M for this ingredient, meaning that more common ingredients have more influence than rare ingredients.

Another observation that can be made while examining matrix M is that some of the values are positive while others are negative. When a value is negative, it means that the presence of one ingredient prevents the presence of the other ingredient in the list of suggested ingredients. This means that these two ingredients do not form a good combination and should not be used together. Some examples are macaroni with potato, macaroni with rice and cocoa with pumpkin. When a value is positive, the two ingredients do make a good combination and the presence of one of these ingredients will increase the chance of the other ingredient to be found in the top ten of suggested ingredients. The higher the value, the higher the chances of the ingredient to be suggested. Two examples of ingredient combinations with a high positive value are apple and cinnamon or chicken and chicken broth.

A final observation is that ingredients commonly found in a certain cuisine have high values for other ingredients that are commonly found in that cuisine. For instance, soy sauce has high positive values for sesame oil, ginger, scallion, garlic, etc. The same reasoning applies to type of recipe: ingredients especially found in desserts have high positive values for other dessert-like ingredients. Vanilla, for example, has high positive values for cocoa, egg, butter, cream, cane molasses, etc.

Table 2
Adding ingredients from a certain category to complete a recipe containing cocoa, coconut and vanilla by means of the two-step RLS model, validated using mean rank as performance measure.

Nuts, seeds and pulses	Alcoholic beverages	Spices
walnut	rum	coriander
pecan	wine	turmeric
almond	brandy	cumin

5. Conclusions

In this article, we have shown that machine learning methods can be used to build models to complete a recipe. By applying machine learning methods on existing recipes, it is not necessary to model the complexity of good ingredient combinations to be able to complete a recipe.

We examined and compared two possible methods that could be used for this purpose. The first method, non-negative matrix factorization, was chosen because of the characteristics of the method and the data. The data is non-negative, as it is binary, and part-based, as a recipe is a linear combination of certain ingredient groups. Non-negative matrix factorization yields a linear non-negative part-based representation of the data, making the combination of this method and this type of data ideal. Since the features generated by NMF are sparse, the interpretation of these results is not that hard. A feature contains ingredients that can be combined in a recipe. The features revealed that ingredients that are suggested to complete a recipe are selected based on the type of recipe (dessert or main dish). NMF is capable of retrieving an eliminated ingredient in a recipe, which can be seen as recipe completion. However, NMF has two disadvantages. A first disadvantage is that the method is not able to take into account information from a second data set. In view of the theory of food pairing, it could be interesting to add information on flavor components, present in ingredients, to the model. However, this is not an option when using NMF. A second disadvantage is the complexity to make

predictions for a new recipe. A linear regression method has to be applied to make predictions for recipes that are not present in the train data (cross-validation). However, this makes the predictions actually an approximation of the true predictions. A solution could be to add the new recipe to the recipe data and factorize the whole matrix, after which the two low-rank matrices can be combined to get the predictions for the new recipe. However, this would be a time-consuming and computationally intensive process.

We compared NMF with a second possible method that presents a solution for both disadvantages of NMF. This second method is two-step regularized least squares, which performs two regressions and allows to take into consideration two data sets. The two-step RLS algorithm is applied to linear kernel matrices of the recipe data and the flavor data. Both matrices contain information on desirable ingredient combinations, which could be useful while completing recipes. However, it is important to control the information gain due to the second data set. In this article, adding flavor data to the model resulted in suggesting ingredients that were too similar to those already present in the recipe, for instance beef and raw beef. These are combinations that will not be found in practice, therefore these combinations are undesired suggestions. Validation of the model will have to eliminate this unwanted information. However, this second data set allows to add new ingredients to recipes, without ever having combined this ingredient with other ingredients in a recipe. By just comparing the flavor components of the new ingredient with those of other ingredients, already used in recipes, the new ingredient can be added to a new or existing recipe.

Besides allowing the use of a second data set, two-step RLS also makes it much easier to make predictions on new recipes. The binary representation of the new recipe is just multiplied with a *model matrix*. There is no need to perform a regression. Just as NMF, two-step RLS is capable of retrieving an ingredient that was removed from a recipe. The results of the two-step RLS model are even better than those of the NMF model. A second experiment was performed to determine whether the methods could be applied to complete recipes. The models were given three recipe sets, each containing three ingredients. For each recipe set, the models were asked to give a top five of ingredients that could best be added to complete the recipe set. The results of this experiment are very promising in terms of usability of machine learning methods to complete recipes.

Next to suggesting additional ingredients to add to new recipes, two-step RLS is also capable of suggesting new ingredients, that where not yet present in any recipe to existing recipes. This is done by adding new ingredients to the flavor data. By comparing the flavor components of the new ingredients with the ones of ingredients that where already used in recipes, these new ingredients can be suggested to existing recipes. This can be seen in Fig. 2(b). A more difficult, but also possible action of the two-step RLS model is to complete new recipes using ingredients that were not yet present in existing recipes, as can be seen in Fig. 2(b) as well.

With better data, the performance of the model could improve, leading to better results, and we could get a better understanding of recipe completion. With better data, the application possibilities would also expand. Machine learning methods could not only be applied to complete a recipe, but also in attempts to personalize recipes. For instance, they could be used for putting together healthier recipes, adding protein rich ingredients to a recipe for sportsmen, eliminating dairy products for lactose intolerant people, etc.

Acknowledgments

The authors acknowledge the support of Ghent University. This work was carried out using the Stevin Supercomputer Infrastructure at Ghent University, funded by Ghent University, the Hercules Foundation and the Flemish Government - department EWI.

References

Ahn, Y.-Y., Ahnert, S. E., Bagrow, J. P., & Barabasi, A. L. (2011). *Flavor network and the principles of food pairing*. http://dx.doi.org/10.1038/srep00196. Scientific Reports, 1.

Arora, S., Ge, R., Kannan, R., & Moitra, A. (2012). Computing a nonnegative matrix factorization—provably. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing* (pp. 145–162). ACM.

Auvray, M., & Spence, C. (2008). The multisensory perception of flavor. *Consciousness and Cognition, 18*, 1016–1031.

Bahamonde, A., Díez, J., Quevedo, J. R., Luaces, O., & del Coz, J. J. (2007). How to learn consumer preferences from the analysis of sensory data by means of support vector machines (svm). *Trends in Food Science & Technology, 18*, 20–28. http://dx.doi.org/10.1016/j.tifs.2006.07.014.

Bajec, M., Pickering, G., & DeCourville, N. (2012). Influence of stimulus temperature on orosensory perception and variation with taste phenotype. *Chemosensory Perception, 5*, 243–265.

Bakalar, N. (2012). Partners in flavour. *Nature, 486*, 4–5.

Cook, D. J., Hollowood, T. A., Linforth, R. S., & Taylor, A. J. (2003). Oral shear stress predicts flavour perception in viscous solutions. *Chemical Senses, 28*, 11–23.

Craswell, N. (2009). Mean reciprocal rank. In L. Liu, & M. Zsu (Eds.), *Encyclopedia of database systems*. US: Springer. http://dx.doi.org/10.1007/978-0-387-39940-9_488, 1703–1703.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press.

Forbes, P., & Zhu, M. (2011). Content-boosted matrix factorization for recommender systems: experiments with recipe recommendation. In *Proceedings of the Fifth ACM Conference on Recommender Systems* (pp. 261–264). ACM.

Freyne, J., & Berkovsky, S. (2010). Intelligent food planning: personalized recipe recommendation. In *Proceedings of the 15th International Conference on Intelligent User Interfaces* (pp. 321–324). ACM.

Guimet, F., Boqué, R., & Ferré, J. (2006). Application of non-negative matrix factorization combined with Fisher's linear discriminant analysis for classification of olive oil excitation-emission fluorescence spectra. *Chemometrics and Intelligent Laboratory Systems, 81*, 94–106. http://dx.doi.org/10.1016/j.chemolab.2005.10.003.

Hastie, T., Tibshirani, R., & Friedman, J. (2008). *The elements of statistical learning: Data mining, inference and prediction*. Springer.

Hoyer, P. O. (2004). Non-negative matrix factorization with sparseness constraints. *The Journal of Machine Learning Research, 5*, 1457–1469.

Huang, J., & Ling, C. X. (2005). Using auc and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering, 17*, 299–310.

Karatzoglou, A., Smola, A., Hornik, K., & Zeileis, A. (2004). kernlab - an S4 package for kernel methods in R. *Journal of Statistical Software, 11*, 1–20.

Kishida, K. (2005). *Property of average precision and its generalization: An examination of evaluation indicator for information retrieval experiments*. Technical Report NII-2005-014E. National Institute of Informatics.

Kondor, R. I., & Afferty, J. (2002). Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the Nineteenth International Conference on Machine Learning* (pp. 315–322).

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature, 401*, 788–791. http://dx.doi.org/10.1038/44565.

Lee, D. D., & Seung, H. S. (2001). Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems* (pp. 556–562).

Morrot, G., Brochet, F., & Dubourdieu, D. (2001). The color of odors. *Brain and Language, 79*, 309–320.

Moyer, M. (2013). The food issue. *Scientific American, 309*.

Murray, J., & Delahunty, C. (2000). Mapping consumer preference for the sensory and packaging attributes of cheddar cheese. *Food Quality and Preference, 11*, 419–435.

Pahikkala, T., Airola, A., Salakoski, T., De Baets, B., & Waegeman, W. (2013). Efficient regularized least-squares algorithms for conditional ranking on relational data. *Machine Learning, 93*, 321–356.

Pahikkala, T., Stock, M., Airola, A., Aittokallio, T., De Baets, B., & Waegeman, W. (2014). A two-step learning approach for solving full and almost full cold start problems in dyadic prediction. In *Machine learning and knowledge discovery in databases lecture notes in computer science* (pp. 517–532). Springer Berlin Heidelberg.

Prescott, J. (2004). Psychological processes in flavour perception. In A. Taylor, & D. Roberts (Eds.), *Flavor perception* (pp. 256–277). Blackwell Publishing Ltd (chapter 9).

Roe, M. A., & Finglas, P. M. (2012). Assessing and improving the quality of vitamin data in food composition databases. *Food & Nutrition Research, 56*. http://dx.doi.org/10.3402/fnr.v56i0.5654.

Sense for Taste. (2014). *The science behind foodpairing*. https://www.foodpairing.com/nl/what-is-foodpairing/the-science-behind/consulted, 12th of February 2014.

Small, D., & Prescott, J. (2005). Odor/taste integration and the perception of flavor. *Experimental Brain Research, 166*, 345–357.

Smith, D., & Margolskee, R. (2006). Making sense of taste. *Scientific American, 284*, 85–92.

Sobecki, J., Babiak, E., & Slanina, M. (2006). Application of hybrid recommendation in web-based cooking assistant. In *Proceedings of the 10th International Conference on Knowledge-based Intelligent Information and Engineering Systems* (Vol. Part III, pp. 797–804). Springer-Verlag.

Stevenson, R. (2009). *The psychology of flavour*. Oxford University Press.

Stevenson, R., & Boakes, R. (2004). Sweet and sour smells: learned synesthesia between the senses of taste and smell. In G. Calvert, C. Spence, & B. Stein (Eds.), *The handbook of multisensory processes* (pp. 69–83). Cambridge, MA: MIT Press.

Talavera, K., Ninomiya, Y., Winkel, C., Voets, T., & Nilius, B. (2007). Influence of temperature on taste perception. *Cellular and Molecular Life Sciences, 64*, 377–381.

ter Haar Romeny, B. M. (2003). The gaussian kernel. In *Front-end vision and multi-scale image analysis* (pp. 37–51). Springer. http://dx.doi.org/10.1007/978-1-4020-8840-7_3. volume 27 of Computational Imaging and Vision.

van Pinxteren, Y., Geleijnse, G., & Kamsteeg, P. (2011). Deriving a recipe similarity measure for recommending healthful meals. In *Proceedings of the 16th International Conference on Intelligent User Interfaces* (pp. 105–114). ACM.

Varshney, K. R., Varshney, L. R., Wang, J., & Myers, D. (2013). Flavor pairing in medieval European cuisine: a study in cooking with dirty data. In *Proceedings of International Joint Conference on Artificial Intelligence Workshops* (pp. 3–12).

Waegeman, W., Pahikkala, T., Airola, A., Salakoski, T., Stock, M., & De Baets, B. (2012). A kernel-based framework for learning graded relations from data. *IEEE Transactions on Fuzzy Systems, 20*, 1090–1101.

Westad, F., Hersleth, M., & Lea, P. (2004). Strategies for consumer segmentation with applications on preference data. *Food Quality and Preference, 15*, 681–687.

Young, S. S., Fogel, P., & Hawkins, D. (2006). Clustering scotch whiskies using nonnegative matrix factorization. *Joint Newsletter for the Section on Physical and Engineering Sciences and the Quality and Productivity Section of the American Statistical Association, 14*, 11–13.

Zampini, M., Sanabria, D., Phillips, N., & Spence, C. (2007). The multisensory perception of flavor: assessing the influence of color cues on flavor discrimination responses. *Food Quality and Preference, 18*, 975–984.

Zetlaoui, M., Feinberg, M., Verger, P., & Clémençon, S. (2011). Extraction of food consumption systems by nonnegative matrix factorization (nmf) for the assessment of food choices. *Biometrics, 67*, 1647–1658. http://dx.doi.org/10.1111/j.1541-0420.2011.01588.x.