Faculteit Bio-ingenieurswetenschappen

Academiejaar 2013-2014

Prediction of Ingredient Combinations using Machine Learning Techniques

Marlies De Clercq
Supervisors: Prof. dr. Bernard De Baets
Prof. dr. Willem Waegeman
Tutor: ir. Michiel Stock

Masterproef voorgedragen tot het behalen van de graad van
Master in de bio-ingenieurswetenschappen: levensmiddelenwetenschappen en voeding

Ghent, June 2014

The supervisor, The co-supervisor,

Prof. dr. Bernard De Baets Prof. dr. Willem Waegeman

The tutor, The author,

ir. Michiel Stock Marlies De Clercq

Acknowledgments

Deze thesis vormt het einde van vijf jaar leven op het boerekot. Vijf jaar waarin plantjes verzameld werden, dieren gedissecteerd, chemische proeven uitgevoerd, wiskundige problemen opgelost, maar in latere jaren ook levensmiddelen bedacht, geoptimaliseerd en geproduceerd op laboschaal. Een periode waarin proffen geprobeerd hebben ons wijzer te maken en ons intellectueel uit te dagen. Deze thesis vormde de grootste uitdaging van al. Ik had deze nooit tot een goed einde kunnen brengen zonder de hulp en steun van heel wat mensen.
Eerst en vooral wil ik Michiel bedanken voor de vele tijd die hij vrij gemaakt heeft om samen te brainstormen. Ik wil hem bedanken voor de vele tips en goede raad. Vervolgens wil ik ook professor De Baets en professor Waegeman bedanken voor hun advies en ideeën. Ik wil Jan Roels bedanken voor zijn hulp met de cluster en het creëren van de website. Verder wil ik ook graag de universiteit bedanken voor het ter beschikking stellen van de Stevin Supercomputer Infrastructuur.
Ik wil ook mijn ouders bedanken voor het mogelijk te maken om bio-ingenieurswetenschappen te studeren en om mij te steunen doorheen mijn volledige schoolcarrière. Zoals mijn mama altijd zegt: “Doorzetten is verder doen ook als je geen zin meer hebt”. Tenslotte wil ik ook mijn vrienden bedanken voor de pauzes zoveel aangenamer te maken en in het bijzonder mijn vriend voor in mij te geloven.

Contents

Summary

Predicting ingredient combinations for *new* recipes is not trivial, since the success of the combination depends on a lot of factors such as taste, smell, texture, temperature, etc. of the ingredients. All these factors together create the flavour perception. However, the factors can influence each other, what makes it very difficult to predict the flavour perception of a certain ingredient combination. For this reason, data mining and machine learning techniques are used to predict ingredient combinations.

The aim of our work is to build a model that suggests one or more ingredients to a given number of ingredients. The idea is based on leftover ingredients in a fridge. A person could list the available ingredients in his or her fridge and the model would predict which ingredients can be combined with the remaining ingredients.

We built our models using data provided in Ahn et al. (2011). In a first step, the data was examined through the use of machine learning techniques. The supervised learning techniques (naive Bayes and random forest) showed that the origin of a recipe can be predicted based on its ingredients and that the flavour components found in a certain ingredient can be used to predict the ingredient's category (e.g. fruit). When using unsupervised learning techniques (principal component analysis and spectral clustering) it is possible to cluster ingredients with the same properties: present in the same recipes or having similar flavour components.

In a next step, predictive models were built to predict ingredient recommendations for some given recipes. A first model is based on matrix decomposition techniques. This model only takes into account the information on ingredient combinations captured in the existing recipes. In a next step, a two-step recursive least squares model was used, taking into account not only the use of ingredients in recipes but also the flavour components of the ingredients. This model can be seen as a variant of the pairwise kernel-based framework for learning relations between objects, as discussed in Waegeman et al. (2012).

To test the models, one ingredient of an existing recipe is removed. The remaining ingredients are given to the model, to test the ability of the model to bring back the eliminated ingredient. The model returns a list of ingredients, where the first ingredient makes the best combination with the given ingredients and the last ingredient the worst. When using the non-negative matrix factorization model, the eliminated ingredient can be found in the top ten of best fitting ingredients for 43.6% of the test recipes. For the two-step RLS model this is true for

57.5% of the test recipes.

Dutch summary

Voorspellen van ingrediëntcombinaties voor recepten is niet eenvoudig. Dit komt doordat het succes van zo'n combinatie afhangt van heel wat factoren, waaronder de smaak, de geur, de kleur, de textuur, de temperatuur van de ingrediënten en nog veel meer. Al deze factoren leiden samen tot de $flavour$perceptie. De factoren kunnen elkaar echter ook onderling beïnvloeden, wat het voorspellen van de $flavour$perceptie heel moeilijk maakt. Om deze reden zijn data mining en machinaal leren nuttig om ingrediëntcombinaties te voorspellen.
Het doel van dit werk is om een model te bouwen dat één of meerdere ingrediënten toevoegt aan een gegeven aantal ingrediënten om zo een recept te vervolledigen. Het idee is gebaseerd op een koelkast, die nog enkele ingrediënten bevat. Een persoon zou de overgebleven ingrediënten in zijn of haar koelkast kunnen oplijsten en het model zou dan voorspellen welke ingrediënten nog aangekocht moeten worden om een smaakvolle maaltijd te creëren.
We bouwden onze modellen gebruik makend van data voorzien in Ahn et al. (2011). In een eerste stap werd de data onderzocht door gebruik te maken van machinaal leren. Het *supervised* leren (naive Bayes en random forest) toonde aan dat de oorsprong van een recept kan voorspeld worden op basis van de aanwezige ingrediënten en dat de ingrediëntcategorie (bv. fruit) voorspeld kan worden door te kijken naar de aanwezige *flavour*componenten in een ingrediënt. *Unsupervised* machinaal leren (principal component analysis en spectral clustering) toonde aan dat het mogelijk is om ingrediënten met gelijke eigenschappen te groeperen: aanwezig in hetzelfde recept of beschikken over gelijke flavourcomponenten.
In een volgende stap werden predictieve modellen gebouwd om ingrediënten te voorspellen die gecombineerd kunnen worden met een aantal gegeven ingrediënten. Een eerste model is gebaseerd op matrix decompositie. Dit model houdt enkel rekening met de informatie over ingrediëntcombinaties verborgen in bestaande recepten. Vervolgens werd een tweestaps *recursive least squares* model gebouwd, dat niet enkel informatie uit bestaande recepten haalt, maar ook informatie haalt uit de flavourcomponenten aanwezig in de verschillende ingrediënten. Dit model kan aanzien worden als een variant van het paarsgewijze kernel-gebaseerd framework om relaties tussen objecten te leren, wat besproken wordt in Waegeman et al. (2012).
De modellen werden getest door telkens een ingrediënt te verwijderen uit een recept. De overgebleven ingrediënten werden meegegeven aan de modellen, dan was het aan de modellen om het geëlimineerde ingrediënt te vinden. Elk model geeft een lijst van best passende ingrediënten, het bovenste ingrediënt in de lijst past het best bij de opgegeven ingrediënten, het onderste ingrediënt past het slechtst. Als er gebruik gemaakt wordt van het *non-negative matrix factorization* model, wordt het geëlimineerde ingrediënt teruggevonden in de top tien van best passende ingrediënten bij 43.6% van de recepten. Voor het *two-step recursive least squares* model staat het bij 57.5% van de recepten in de top tien van best passende ingrediënten.

Prediction of Ingredient Combinations using Machine Learning Techniques

M. De Clercq, M. Stock, B. De Baets, W. Waegeman
MM.DeClercq@UGent.be
KERMIT, Ghent University

Introduction

Aim: To build a model that recommends ingredient combinations
Problem: The mechanism to create new recipes is unknown:

- *Taste and smell*
- *Sound* (e.g. crispness of chips, snap of chocolate)
- *Colour* (e.g. the colour of wine)
- *Texture* (e.g. viscosity of honey determines sweetness)
- *Temperature* (e.g. bitterness of warm beer)
- *Interactions* (e.g. vanilla odour can enhance the perception of sweet taste without increasing the amount of sugar)

→ ***Flavour perception***

Two **data sets**, found in Ahn, et al. (2011).

1. 56,498 recipes containing 381 ingredients + origin of the recipes
2. flavour profile of 1,525 ingredients coming from fourteen categories (e.g. fruit, herb, meat)

A closer look at the data

Naive Bayes and random forests:

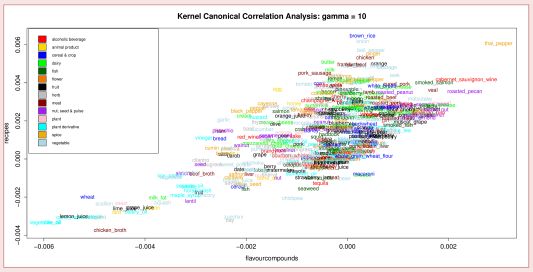
- origin of a recipe is partially determined by its ingredient composition
- category of an ingredient is partially determined by its flavour profile → ingredients within a category have a similar flavour profile

PCA and spectral clustering:

- ingredients can be clustered based on their use in recipes and their flavour profile → needed to predict ingredient combinations

Canonical correlation analysis:

- relation between the flavour components present in ingredients and the use of these ingredients in recipes
  - high correlation: mussel, cognac, fig, etc.
  - low correlation: onion, butter, egg, etc.



Model building

Model evaluation

1. Eliminate ingredient from recipe
2. Give modified recipe to the model
   a) Model returns list of best fitting ingredients
   b) Determine position of the missing ingredient in this list

Matrix decomposition

$$Y \approx X_{recipes} X_{ingredients}^{T}$$

- Y = binary recipe data
- $X_{recipes}$, $X_{ingredients}$ = two low-rank matrices
- best technique: *non-negative matrix factorization*:

→ eliminated ingredient in the top ten of best fitting ingredients for 43.6% of the recipes

Two-step recursive least squares

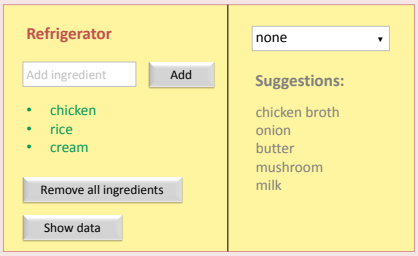
$$Y \approx K_{recipes} W K_{flavours}$$

- Y = binary recipe data
- information from both the recipe data ($K_{recipes}$) and the flavour data ($K_{flavours}$)

→ eliminated ingredient in the top ten of best fitting ingredients for 57.5% of the recipes

Model in practice

The model returns a list of ingredients that can best be combined with the given ingredients.



References

Y.Y. Ahn, S.E. Ahnert, J.P. Bagrow, and A.L. Barabási. Flavor network and the principles of food pairing. Scientific Reports, 1(196), 2011.

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

Figure 1: Summary of the work on a poster made for Benelearn 2014.

Chapter 1

Introduction

Everyone has had the following problem at least once in his or her life. You still have several ingredients left in your refrigerator, but you do not know how to combine them into a good dish. You are convinced that with the addition of only one or two ingredients, they would make a splendid dish. But how will you know what to add to your grocery list?
There already exist cookbooks in which you can look for a recipe containing the remaining ingredients, there already exist websites that search through a database of recipes for all those recipes containing all (or several) remaining ingredients in your refrigerator and there even exists a website where you can give one of the remaining ingredients and get a list of ingredients that make good combinations with this one ingredient. But what if you are just looking for the best fitting type of meat, or which herbs to use with the remaining ingredients? Or you just want to see which other ingredients you can add to the remaining ingredients in general. There is no book or website that tells you which type of meat can best be combined with a set of given ingredients. Solving this problem is our aim.
In this work, we will present a set of models that gives for a given set of ingredients, those ingredients that can best be combined with all the given ingredients. These combinations will not be restricted to combinations found in recipes. We will present a set of models that not only make combinations based on those found in recipes, but also based on the presence of similar flavour components in the different ingredients. This will allow us to create *new* ingredient combinations.
Before we could start building the models, we had to find out what causes people to say some combination of ingredients is really tasteful, but another combination just does not work. The answer is: the flavour of the ingredients is responsible for accepting an ingredient combination. However there are a lot of other factors like taste, sound, color, temperature, etc. that influence our perception of flavour. And these factors can also influence each other and in that way indirectly influence the flavour perception. More information about these interactions can be found in Chapter 2. These interactions are difficult to predict and they are not trivial. For this reason we will use data mining and machine learning techniques to

build the model that will predict which ingredients will make a good combination with a given set of ingredients and which will not. We will try to make new ingredient combinations with our model, combinations that are unexpected, but tasteful.
Literature on existing models can be found in Chapter 3. In Chapter 4 and 5 the data is examined using data mining and machine learning techniques. The results of these chapters are used to build a first model using matrix decomposition techniques that can be found in Chapter 6. The final model is a two-step recursive least square model, everything about how it is trained, tuned and tested and the results can be found in Chapter 7.

Chapter 2

The science behind recipes

As the aim of this work is to predict ingredient combinations, it is important to look at the science behind recipes. In this chapter the interactions between taste, smell, temperature, etc., leading to flavour perception of ingredients are summarized. Besides that, other reasons to add certain ingredients are studied.

2.1 Flavour perception

2.1.1 Smell: olfactory system

Smell is an important part of flavour (Smith and Margolskee (2006)). Smell has two main functions: first of all it allows people to sniff the air in the environment, and secondly it allows people to enjoy their food (Auvray and Spence (2008)). This is a result of the two different routes to reach the olfactory receptors: either through the nose by sniffing (*orthonasal* olfactory) or through the mouth into the nose while breathing out (*retronasal* olfactory). These routes can be seen in Figure 2.1. The last one occurs while eating and drinking: volatile chemicals will come free in the mouth and reach the receptors in the nasopharynx, so they play a big role in the perceived flavour of the food.

2.1.2 Taste and the influence of smell

The taste of a foodstuff is perceived by the tongue, on which papillae containing chemical receptors are found (Moyer (2013)). These receptors can receive the five basic tastes: sweet, salt, sour, bitter and umami (Smith and Margolskee (2006)). Sweet taste identifies the energy-rich nutrients, umami is an indicator of amino acids, salt taste is important for a proper dietary electrolyte balance, sour and bitter taste, however, are mostly related to potentially noxious and/or poisonous chemicals (Chandrashekar et al. (2006)). This explains why children have a natural instinct to like sweet and salty food and dislike sour and bitter food.
There exists confusion between senses of taste and senses of smell. The most famous example is

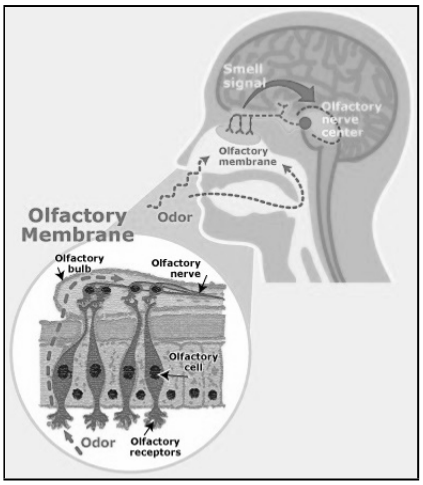


Figure 2.1: Orthonasal and retronasal olfactory (source: Exploratorium).

that people say they cannot taste their food when their nose is blocked. However, as mentioned above, the taste receptors are located in the mouth and not in the nose. This indicates that smell and taste are coupled. Another example is that people will describe the odour of vanilla as sweet (Stevenson and Boakes (2004), Prescott (2004)), even though sweetness is associated with taste and not with smell. Besides the confusion, odours can also influence the intensity of a perceived taste. For instance, a sweet taste can be experienced as more intense (sweeter) when caramel odour is added to a sucrose solution, without changing the sucrose concentration (Auvray and Spence (2008)). This phenomenon is described as sweet enhancement; the reverse phenomenon does also exist and is called sweetness suppression. This also occurs with other tastes like sour and bitter. An odour mostly found in sweet dishes will induce a sweetness enhancement effect. This relation between the odour and sweetness might naturally be formed during eating (Auvray and Spence (2008), Prescott (2004)). This shows that the change in taste perception due to the presence of odours is not due to a specific property of the odour, but a result of linking odours with taste. An odour can have a different effect on taste for people from different parts of the world: not all ingredients (and thus odours) are in all parts of the world used in a same type of dish. It is possible that an odour is related to sweet dishes in Western countries, but not in other parts of the world. This would result in a sweetness enhancement effect of the odour for western people, but not for others (Auvray and Spence (2008), Small and Prescott (2005)).

2.1.3 Influence of texture, temperature, vision and audition

Texture has an influence on the perception of taste and flavour. For instance, when the viscosity of a solution increases, a suppression of both flavour and taste perception can occur (Auvray and Spence (2008)). In 1987 Baines and Morris investigated the perception of sweetness and strawberry aroma in solutions with different concentrations of guar gum (thickener).

They found that the flavour and taste perception were progressively suppressed when the concentration of guar gum became larger than the coil-overlap concentration. At this concentration the hydrocolloid chains start to overlap each other in the solution, resulting in a decrease of movement and a clear increase of the viscosity of the solution. Below this concentration the perception of taste and flavour was practically unchanged (Cook et al. (2003)). On the other hand perceived texture can also be influenced by the intensity of the taste. For instance increasing the amount of sucrose in a solution can lead to a lower perceived viscosity (Auvray and Spence (2008)).

The most famous examples of how temperature can influence the taste perception are the intense sweetness of melted ice cream or the bitterness of a warm beer (Talavera et al. (2007), Bakalar (2012)). Studies have shown that temperature can influence the perceived maximum intensity from bitter and sour stimuli, but not from sweet stimuli (Bajec et al. (2012)). The effect of temperature on the perceived sweet taste is different for different types of sugars: the perceived sweetness increases with increasing temperature in case of glucose, fructose and sucrose. However it decreases with increasing temperature when looking at aspartame and temperature does not seem to have an effect on the perceived sweetness of saccharin, an artificial sweetener (Bajec et al. (2012)).

A study of the interaction between the vision of colours and odour determination by Morrot et al. (2001), has shown that the odour of a white wine, coloured red with odourless dye, is described with odour terms related to red wine. This experiment confirmed the existence of a perceptual illusion between odour and colour. But this is not the only effect of colour, it also has an influence on the taste and flavour perception. Studies have shown that the perceived intensity of flavours and tastes increases when the colour level of a solution increases (Auvray and Spence (2008)). Zampini et al. (2007) showed that people will link certain flavours with the colour of the solution. First, they let their participants link each flavour with a solution of a specific colour (visually). In a second step, they had to taste the solutions and identify their flavour. The combination of colour and flavour could be the same as in the previous step or different or the solutions could be colourless. The participants knew that the colour of the solutions was not an indicator for its flavour. The results of these experiments show that the accuracy of flavour recognition was significantly lower when the colours and flavours were mixed with respect to the first step, than when the colours and flavours were not mixed (or the solutions were colourless).

Sound of food will mostly influence the perception of its texture properties. Example of this phenomenon are the crispness of potato chips (Auvray and Spence (2008)) and the snap of a chocolate bar. All the interactions mentioned above are also described in detail in Stevenson (2009). Figure 2.2 gives a schematic representation of most of the interactions mentioned above, except for odour.

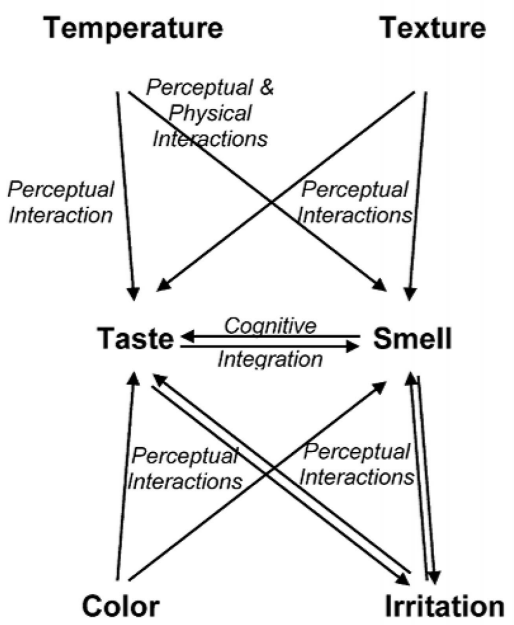


Figure 2.2: Summary of interactions that occur during digestion (source: Delwiche (2004)). Cognitive interactions are based on knowledge. Sweet enhancement is a nice example of such an interaction. For physical interactions there was to be a physical process that froms the basis of the interaction. For instance, when heating a foodstuff, volatile components will be released, which will lead to an increase in smell. Perceptual interactions occur when the interaction has an effect on the detection of components. For instance, increasing the viscosity of a product (texture) will lead to a slower diffusion of components through and out of the product, leading to a lower receipt of components at the taste and olfactory receptors. So an increase in viscosity will result in a decrease of taste and smell intensity.

2.2 Why add certain ingredients?

Some ingredients are added to a recipe, not for their flavour, but for other specific characteristics. They are added because of their ability to thicken a solution (sauces) or their gel forming capacity (pudding) or their emulsifying properties to form a stable emulsion (chocolate), etc. A well-known example is egg. The egg yolk contains lipoprotein, which give the yolk emulsifying properties, as they can bind both hydrophilic proteins and hydrophobic lipids. For instance, mayonnaise contains egg yolk, not for its flavour, but for its emulsifying properties. The egg white contains ovalbumine and ovoglobuline among other proteins. Ovalbumine will create a gelling structure after boiling and ovoglobuline is the reason that egg white can be transformed into a stable foam. This last property gives for instance cake and mousses their

airy structure (De Meulenaer (2011)).
Beside egg, there are other ingredients with unique properties, such as butter. Cocoa butter is chosen to make chocolate because of its melting properties. When it is crystallized into the desired crystal form, it will make the chocolate melt in the mouth. An other example is potato: it is added to soup before boiling and will lose its structure during boiling, this will release the starch inside the potato, which will create a thickening effect of the soup.

2.3 Conclusion

It can be concluded that the interaction between smell, taste, touch, vision and audition are quite substantial. This brings us to the conclusion that flavour can not easily be defined or modulated, since it is actually a mixture of all the senses mentioned above, because all these senses will interact with each other and will create the final perception of flavour of the food a person gets when eating it. Therefore data mining and machine learning techniques are suitable for predicting ingredient combinations.
Another conclusion is that an ingredient is not always added for its flavour but sometimes it is added as an aid for structure or stability.

Chapter 3

Modeling of recipes

There already exist several models that relate to recipes. Some models predict which recipes people will like based on scores they have given to some other recipes or to certain ingredients. Other models are more like search engines: when given several ingredients, they search a database of recipes and return those recipes that contain all or some of the given ingredients. Other models predict ingredient combinations as is the aim of this work. Some of the existing models will be reviewed below.

3.1 Some existing models using recipe data

Sobecki et al. (2006) developed a web-based information system (a hybrid recipe recommender), which can recommend certain recipes to different types of users based on given information in the user profile. This system uses fuzzy reasoning for demographic recommendations of recipes. The demographic attributes that are used in the model are age, gender, number of inhabitants in the place of living, cooking experience and whether or not the user wants vegetarian recipes or not. Hybrid means that the model is a combination of demographic, content-based (user likes new item, if he/she liked a similar item) and a collaborative approach (user likes an item, if a similar user likes the item or, when the user unregistered: recommend item that is liked by whole population) of recommendation.
Freyne and Berkovsky (2010) did research on designing a recipe recommender that recommend healthy recipes. Their aim was to educate the users and help them get a healthier lifestyle with personalized recommendations and to keep this healthier lifestyle. They gathered ratings on individual ingredients, but also on recipes. Based on the scores given, the model will recommend a certain recipe. For ingredients that are not yet rated by the user, the ratings are predicted. They will also relate the recipes and the food items to optimize the accuracy of the recommendation, by breaking down the recipe into its ingredients.
Forbes and Zhu (2011) improved the accuracy of recipe recommendations by using the matrix factorization approach for collaborative filtering. The collaborative filtering approach is

able to recommend an item without understanding the item itself. This approach links users together based on similar taste. It suggests that when one of the users like a certain recipe, other users (with similar taste) will like it as well, although it does not know anything about that recipe. Before the model can be built, the users need to give scores to recipes. For each recipe a list of the ingredients is available for the model. The model is 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 (1 if an ingredient is present in the recipe, 0 otherwise). A single-step model would only take into account the information captured in the score matrix.
In 2011, van Pinxteren et al. designed a recipe recommender with the aim to deliver healthier variants of routine recipes. They constructed a measure that determines the similarity of recipes, which is created with a user-centered approach. The similarity was not determined based on the ingredients, but by means of a card-sorting experiment. People could find essential information of a recipe on the card of that recipe, like the title, ingredients, preparation time, a picture, etc. There were two sets of cards: in the first one, each card contained a very different kind of recipe, in the second set every card contained a different Italian pasta with meat. As the recipes in this second set were all very similar, this set was made to see which factor (beside ingredients) had most influence. With the first set the participants had to divide the recipes into groups and characterize the different groups. This showed which characteristics are mainly used to determine similarity between recipes. With the second set, one recipe was selected and the participants had to rank the remaining recipes on a five point scale going from very similar to very dissimilar. The results of these tests were used to determine those ingredients which are most important to people to decide whether or not two recipes are similar. Only these ingredients were taken into account in the model. Also, the results on the meal type, preparation time, cooking technique, etc. were evaluated and in total 55 features were created that are important for people during recipe similarity determination. They created an algorithm that can search for these features in a text. Each feature was assigned a certain weight in the similarity measure, based on the results of the participants. This similarity measure was then used to recommend healthier alternatives for the people that fit their daily routine.

3.2 Search engines

There already exist several websites that provide search engines for recipes. Some examples are *supercook.com*, *myfridgefood.com* and *recipematcher.com*. All these sites work in a similar way. One can select some ingredients with which one would like to make a dish. After selecting the ingredients, one can press a button and the search engine embedded in the site will start to search through all the existing recipes in its database and select those which contain all or most of the selected ingredients. Those recipes are shown in a list on the site. For each recipe

the site tells which of the selected ingredients will be used and which additional ingredients one will need to buy, in order to be able to complete the dish.

3.3 Foodpairing.com

Another website built around a model about recipes, more precisely ingredient combinations, is Foodpairing.com. The model is not based on existing recipes, but on the principle that a good combination can be achieved when the combined ingredients have the same major flavour components. So the model is created based on scientific flavour analysis of foodstuffs. For each foodstuff, the flavour components are determined by using gas chromatography coupled mass spectrometry. Once all the flavour components are identified, the major flavour components are determined. These are the components that will determine the smell of the foodstuff and it are those components that are taken into account in their model. Foodpairing.com gives, for a certain ingredient, a number of ingredients that are most similar to it, based on the flavour composition. Those ingredients result in the creation of a tasteful combination when the ingredient is combined with one of the proposed ingredients.
The model returns a tree, with the ingredient of interest in the middle and all the ingredients with similar major flavour component around the center. An example is given in Figure 3.1. The ingredients are divided into categories and for each category the three ingredients which are most similar to the ingredient of interest are represented in the model. There are nine categories (e.g., meat, fruit, vegetables) which brings the number of suggested ingredient to twenty-seven. The more flavour components a suggested ingredient has in common with the ingredient of interest, the closer the ingredient is located to the center of the tree. This model does not return recipes, but it does suggest ingredient combinations that are new and of which one would not expect that the ingredients really match together.
The website of foodpairing.com has been renewed since the beginning of this year and now contains a new model: *twist it*. This model makes it possible to replace one ingredient in an existing recipe: one gives the ingredients in the recipe and tells the model which ingredient should be replaced. The model returns a list of fitting ingredients. This can for instance be used when one is allergic to one of the ingredients in the recipe.

3.4 Conclusion

Most of the models that are already built are made to recommend existing recipes to people based on information of those people that was gathered. This is not the aim of the models that are built in this work. These models will predict new ingredient combinations or, more precisely, give for a certain number of ingredients those ingredients that make good combinations with all given ingredients. The Foodpairing model is most similar to the models in this work. However, it gives ingredients that make good combinations with a single ingredient.

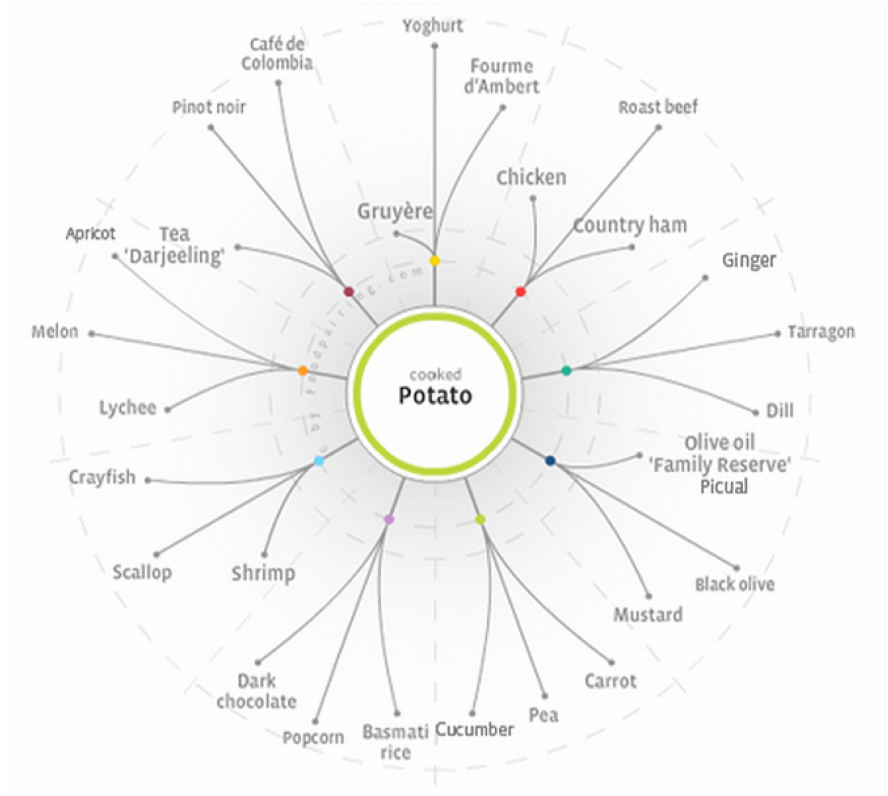


Figure 3.1: Example of a Foodpairing tree (source: Sense for Taste (2014)). Different branches leave the center of the tree. Each branches contains ingredients from a different category, for instance, dairy, meat, herbs, etc. The closer the ingredients are to the center of the tree, the more flavour components they have in common with the ingredient in the middle. This means that the ingredients closest to the center make the best combinations with the given ingredient.

Chapter 4

First step of model building: data

4.1 Description of the existing data sets

One of the main requirements to build a good model is sufficient data, both in quantity as well as in quality, to train and test the model. The data that will be used to build the models is found in "Flavor network and the principles of food pairing", by Ahn et al. (2011). This data consists of five data sets covering recipes, ingredients and flavour components. A first data set contains 56,498 recipes originating from eleven different regions. The ingredients needed to prepare each recipe are enumerated, as well as the region where the dish originates from. All these recipes together contain 381 different ingredients, from almond to zucchini. The number of ingredients used in a recipe ranges from very small to very large, the distribution can be found in Figure 4.1.
The eleven regions from which recipes are collected are Africa, East Asia, Eastern Europe, Latin America, the Middle East, North America, Northern Europe, South Asia, South East Asia, Southern Europe and Western Europe. Table 4.1 shows for each region the number of recipes present in the data and the average number of ingredients used in the recipes coming from this region.
A second data set contains 1,503 ingredients, including the 381 ingredients found in the first data set. The ingredients are given an identifier going from zero to 1,529. Besides the number and the name of the ingredient the data shows the ingredient category of each ingredient. There are fourteen different categories covering all types of ingredients. The categories are alcoholic beverage, animal product, cereal and crop, dairy, fish, flower, fruit, herb, meat, nut, seed and pulse, plant, plant derivative, spice and vegetable. The subdivision of the ingredients into the different categories allows to predict to best fitting ingredient within a specific category. For instance when you leave one ingredient out of a recipe, the missing ingredient can be predicted or the best fitting ingredient within the same category of the missing ingredient can be found. Some examples of animal products are honey and gelatin. Examples of plant derivatives are cocoa and tea. Parsley and basil are herbs, while pepper

Table 4.1: Number of recipes and number of ingredients per recipe per region in the flavour Network data.

Region	Number of recipes	Average number of ingredients	st. dev.
Africa (Afr)	352	10.45	4.22
East Asia (EaAs)	2512	8.96	3.81
East Europe (EaEu)	381	8.39	3.56
Latin America (LaAm)	2917	9.38	3.66
Middle East (MiEa)	645	8.39	3.63
North America (NoAm)	41524	7.96	3.44
North Europe (NoEu)	250	6.82	3.20
South Asia (SoAs)	621	10.29	4.54
Southeast Asia (SEAs)	457	11.32	4.60
South Europe (SoEu)	4180	8.86	3.57
West Europe (WeEu)	2659	8.03	3.71

and ginger are spices.
In the third data set an overview is given of 1,107 flavour compounds found in food. The flavour compounds are numbered from zero to 1,106 and each compound is provided with its own CAS-number. The CAS-number can be used to find more information about the flavour compounds like the chemical structure, synonyms, etc. A fourth data set consists of two columns. The left column contains different ingredients shown with their id-number. A same number can be found more than once. The right column contains id-numbers of flavour compounds. This data set links the flavour compounds with the ingredients. A fifth data set reorganizes the fourth data set and gives for two ingredients the number of flavour compounds they have in common. The information found in data sets two, three and four is used to give an overview of the different ingredient categories. For each category the number of ingredients given for this category and the average number of flavour compounds found in these ingredients is given in Table 4.2.

4.2 Bringing the data in the desired form

4.2.1 Recipes and their ingredients

The first data set containing for each recipe the names of the ingredients found in this recipe is transformed into a binary matrix Y, since a binary matrix is more typical for machine learning and data mining. The features of the data are the 381 different ingredients (i) found in this data set. The rows of the matrix contain the different recipes (r). If an ingredient is

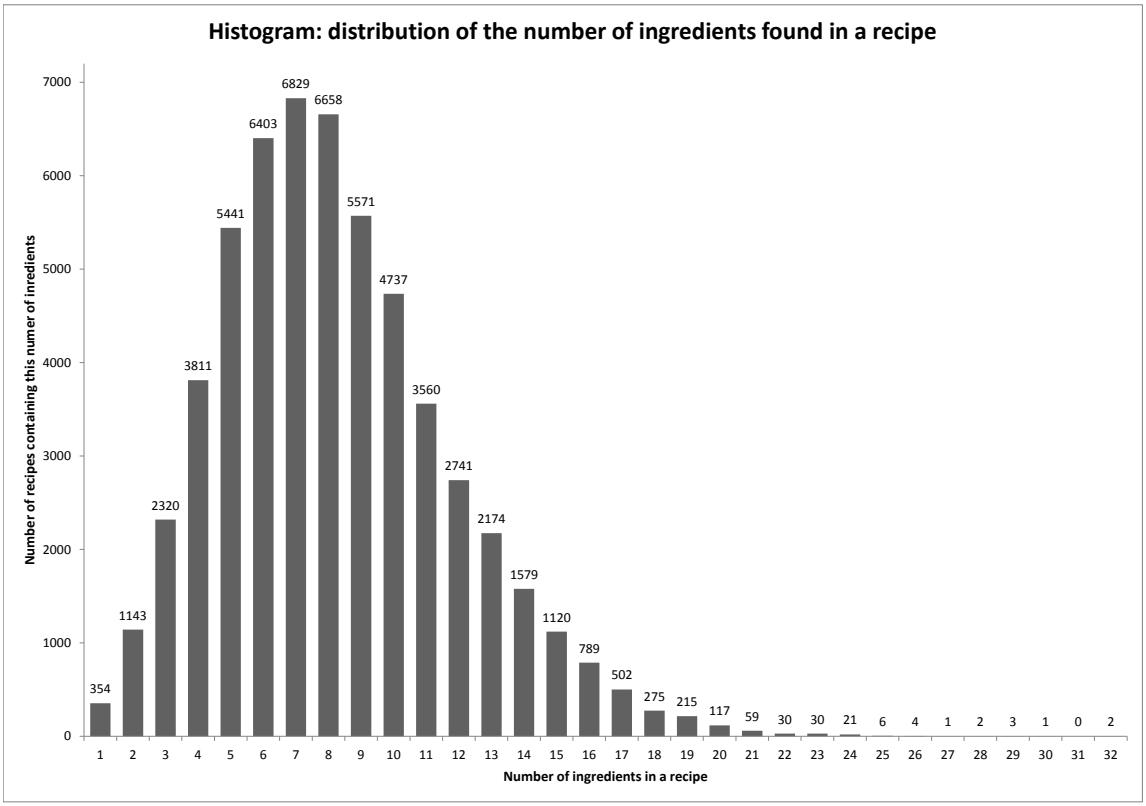


Figure 4.1: Distribution of the number of ingredients in the recipes

found in a recipe a one will be found where the row of this recipe meets the column of this specific ingredient. If the ingredient is not found in the recipe the element on this place in the matrix will be zero. This results in following matrix:

$$Y_{ri} = \begin{cases} 1, & \text{if ingredient } i \text{ is part of recipe } r; \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

A 382th column is added to the matrix. This column contains the origin of the recipes. The new data set is a sparse matrix with a filling degree of 2.16%. This matrix can be used to check whether or not the origin of the recipe can be predicted based on the ingredients found in the recipe.

4.2.2 Combinations of ingredients in recipes

The data set constructed in the section above (Y) can be used to create a matrix containing for each couple of ingredients the number of recipes where the two ingredients are used together. This matrix (A)is square and symmetric. The 381 different ingredients form both the variables as the observations. The matrix can easily be constructed by multiplying the transpose of the

Table 4.2: The different ingredient categories and the number of ingredients and average flavour compounds found in each category per ingredient.

Category	Number of ingredients	Average number of flavour components	St. dev.
alcoholic beverage	50	59.56	75.67
animal product	18	8.50	16.37
cereal/crop	39	32.69	33.82
dairy	39	91.90	60.96
fish/seafood	56	46.79	30.16
flower	66	6.35	8.03
fruit	186	42.10	44.92
herb	90	12.00	17.74
meat	57	85.23	59.72
nut/seed/pulse	33	29.48	34.48
plant	313	3.33	7.01
plant derivative	420	12.00	38.63
spice	54	20.51	20.86
vegetable	104	36.67	42.35

binary matrix (381×56498) created in the section above with the this matrix (56498×381):

$$A = Y^T Y. \tag{4.2}$$

This matrix will be used to study the combining ability of two ingredients in a recipe.

4.2.3 Ingredients and their flavour compounds

A new matrix is constructed using the data in the fourth data set to combine the ingredients with their flavour compounds. This new matrix (X) contains 1,525 rows; these are the ingredients for which flavour compounds are given. This means that there are five ingredients for which no flavour compounds are given. The data set consists of 1,107 variables or 1,107 flavour compounds. The data in this new data set will be binary. For each ingredient a one is given in the columns of the flavour compounds that are found in that ingredient and a zero if the flavour compounds are not found in the ingredient. Two extra columns could be added to the matrix; a first containing the number of flavour compounds found in each ingredient and a second giving the ingredient category in which the ingredient belongs to:

$$X_{ic} = \begin{cases} 1, & \text{if flavour component } c \text{ is present in ingredient } i; \\ 0, & \text{otherwise.} \end{cases} \tag{4.3}$$

4.3 Data mining analysis of the data

In this section the data is analyzed for a first time. To examine the data, machine learning techniques are used. There are two types of techniques: supervised and unsupervised. In this section mainly supervised learning techniques are used to analyze the data. In the next chapter, the data will be studied using the unsupervised machine learning techniques. More information on machine learning and the difference between supervised and unsupervised techniques can be found in Barber (2012a).
The data is examined using supervised learning techniques to see if there are differences in use of ingredients between the different regions and whether the composition of a recipe is enough to determine its origin. For this part the binary data set, containing the recipes as labels and the ingredients as features, is used (Section 4.2.1). Next to that, the ingredient categories are studied. The aim is to determine whether or not ingredients within one category consist of different flavour components compared to ingredients belonging to a different category. Or, in other words, whether or not the category of an ingredient can be predicted based on the flavour components found in this ingredient. For this analysis the binary data set in Section 4.2.3 is used. These results will be a first indication to determine whether or not a universal model predicting ingredient combinations can be made; or if each region would need its own model.

4.3.1 Origin of recipes

The object of this analysis is to determine whether or not the origin of a recipe can be predicted based on the ingredients that are in the dish. For these predictions the binary representation of the 56,498 recipes is used. The features of the models that are built are thus the 381 different ingredients and the response is the origin of the recipes. First a heat map is drawn of the data set together with a dendrogram of the regions based on this heat map. In a following step the data is divided into four groups for cross validation. One group serves as test data. The remaining groups will form the data for training the model. The origin of the recipes is predicted using a naive Bayes model and a random forests classifier. For each model, a confusion matrix is constructed based on the predicted and the actual regions of origin. Based on these matrices the classification accuracy for each model will be determined.

Naive Bayes

Due to the large difference in number of recipes per region; it is necessary to subsample a same number of recipes per region to train the model or to give a weight to each region. This guarantees that each region has the same importance while training the model. Otherwise the group with the largest number of data, will dominate the trained model, which results in

wrong classification of the test recipes into the largest group.
The naive Bayes-model is built by using a preprogrammed function in R ('`e1071`' package). This function allows the user to give weights to the different groups. Next to the binary data of the recipes, per origin the number of recipes in the train set is given to train the model. After the model is built and trained, the accuracy of the model is determined. The test set is given to the model and the origin of each recipe is predicted. The predictions are compared with the actual origins in a confusion matrix and through the classification accuracy.
If the recipes would be classified randomly into the eleven groups, the accuracy would be approximately 0.09. The trained model does not assign all the recipes of the test data to the right region. However, the numbers (different from zero) in the confusion matrix are quite well distributed over the matrix. This means that there is no region that is chosen more than another. However, since most recipes are coming from North America, this is the region that shows most misclassifications in both ways. Recipes that are actually coming from North America are assigned to other regions; and recipes that do not belong to North America are assigned to North America. The accuracy of the prediction achieved by the naive Bayes-model is 0.714, which is a lot higher than 0.09. This means that the origin of the recipes can be predicted quite well by the naive Bayes model.

Dendrogram based on the heat map

A dendrogram is actually an unsupervised learning technique, but it is performed here, since it will tell something about the differences between the different regions and it can be used to group the regions for further analysis. Due to the large number of recipes in the data set, it is not possible to create a heat map of the whole data set. So before analyzing the data through a heat map, the data is transformed into a (11×381) matrix. The eleven rows of this matrix represent the eleven regions and the 381 column represent the ingredients found in the recipes. For each region the posterior probability to find a certain ingredient in a recipe coming from that region is calculated. This probability is placed in the new matrix on the row associated with the region for which the probability was calculated and the column associated with that certain ingredient. Once the matrix is completely filled in with the posterior probabilities, the heat map can be built using this new data set. The heat map is created using a preprogrammed function in R (`heatmap.2`) and can be found in Figure 4.2. Once the heat map is completed, a dendrogram of the regions is constructed based on this heat map. The dendrogram can also be found in Figure 4.2.
The regions can be separated into two clear groups. One group contains the Western regions: Northern-Europe, Western-Europe, Eastern-Europe and Northern-America. The other group contains the remaining regions. This means that the composition of the recipes coming from the Western regions is much alike, while it is rather different for the recipes made in the non-Western regions. Looking at the dendrogram this last group rapidly separates into three groups: Eastern regions, Southern regions and South-Asia. This means that the

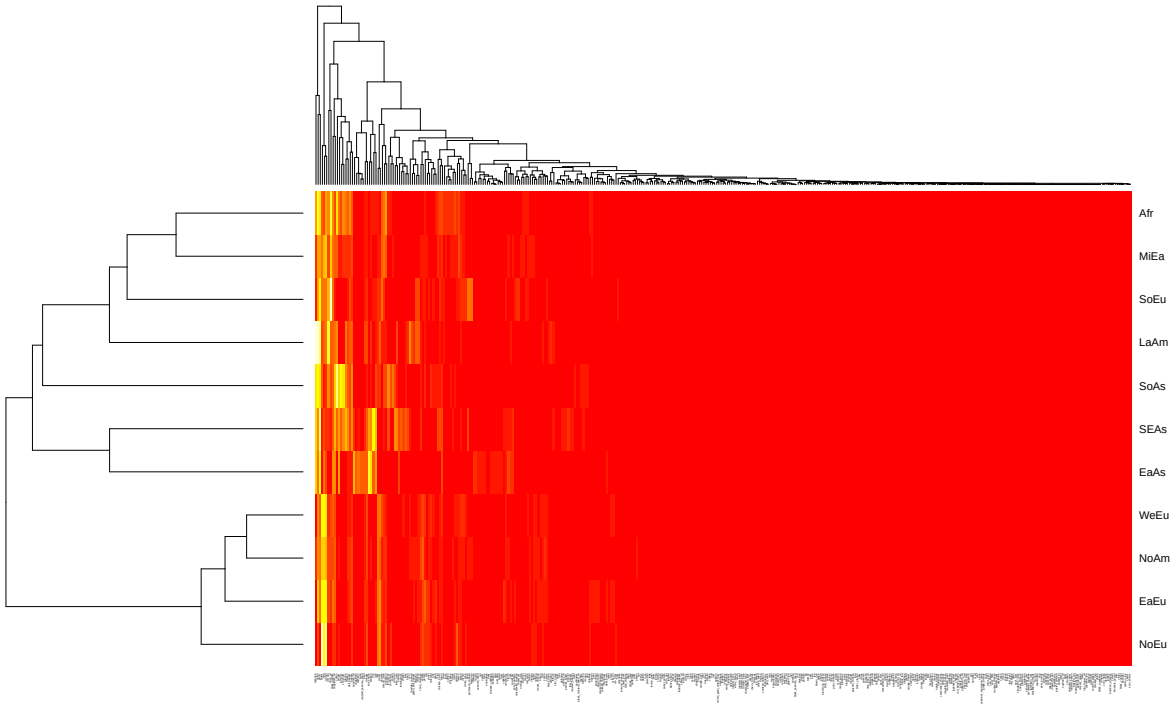


Figure 4.2: Hierarchical clustering of the origin of the recipes based on the ingredients by means of a heat map.

(combinations of) ingredients used in these recipes are also quite different for the different groups of regions. These four groups are given below and will be used in further analysis to subdivide the recipes into four groups instead of eleven:

- Cluster 1: West: Northern Europe, Eastern Europe, North America, Western Europe
- Cluster 2: East: East Asia, Southeast Asia
- Cluster 3: South Asia
- Cluster 4: South: Latin America, Southern Europe, the Middle East, Africa

When the regions are divided over three groups, the third and fourth cluster become one.

Random forests

In a next step the recipes are classified using a random forests model. R also provides a preprogrammed function to build a random forests model in the `RandomForest` package. The model is trained and tested with the same train and test data as used in the naive Bayes model. This allows for comparison of both methods. The trained random forests model is used to predict the origin of the recipes in the test data. A confusion matrix is built and the accuracy is determined. The random forests model predicts the origin with an accuracy of

0.79, a bit better than the naive Bayes model. It can be concluded that the origin of a recipe can be predicted quite well based on the ingredients that are used in the recipe.
A random forests model can not only be used to predict the origin of the recipes; it can also be used to determine which ingredients are important during the classification process of the recipes. The thirty most important ingredients to determine the origin of each recipe are given in Figure 4.3. Olive oil and cayenne are clearly the key ingredients. Important ingredients must have a high posterior probability of appearance in some of the regions and a low probability in the other; in that case the ingredient can make a preselection of the regions of origin.
The posterior probabilities of appearance in the different regions for these two recipes can be found in the naive Bayes model. Olive oil is mostly related to the Southern regions (Africa, Southern Europe, the Middle East) and has a very low probability to be found in recipes coming from the Eastern regions (East and Southeast Asia) and Northern and Eastern Europe. Cayenne on the other hand has a very high probability to be found in Eastern recipes and a very high to medium probability of appearance in recipes coming from the Southern regions. Sesame oil only has a high to medium posterior probability in East and Southeast Asia and has a very low posterior probability in all other regions. Tomato is mostly found in recipes coming from Southern regions (Africa, Latin America, Southern Europe). These differences in posterior probability directly show the importance of these ingredients. The number of ingredients (features) that should be taken into consideration while building the model could be reduced based on the importance of the ingredients determined by the random forests model. For instance, only the fifty most important ingredients could be chosen as features for the model instead of all 381 ingredients.
Since the random forests model has the largest accuracy, this model is used to test the clusters found in the dendrogram of the heat map. The 56,498 recipes are redistributed into the four clusters of the dendrogram. These clusters form the new *origin* of the recipes. To predict these origins a random forests model is built and trained with three quarters of the recipes. The trained model is used to predict the origin of the remaining recipes. Just as before, the predictions are compared to the actual clusters in a confusion matrix and the accuracy of the model is determined. The only difference is that there are only four categories instead of eleven regions. This means that the accuracy of a random model would be around 0.25. The actual classification accuracy of the model is 0.86. The value is higher than before the recipes were grouped together based on the clusters found in the dendrogram, which means that the clustering was good. The ingredients that are important for this classification are almost the same as those in the general model and can be found in Figure 4.4. However, the order has changed a little, as well as the shape of the graph. This means that the redistribution into the four groups makes little difference in predicting the origin of the recipes.

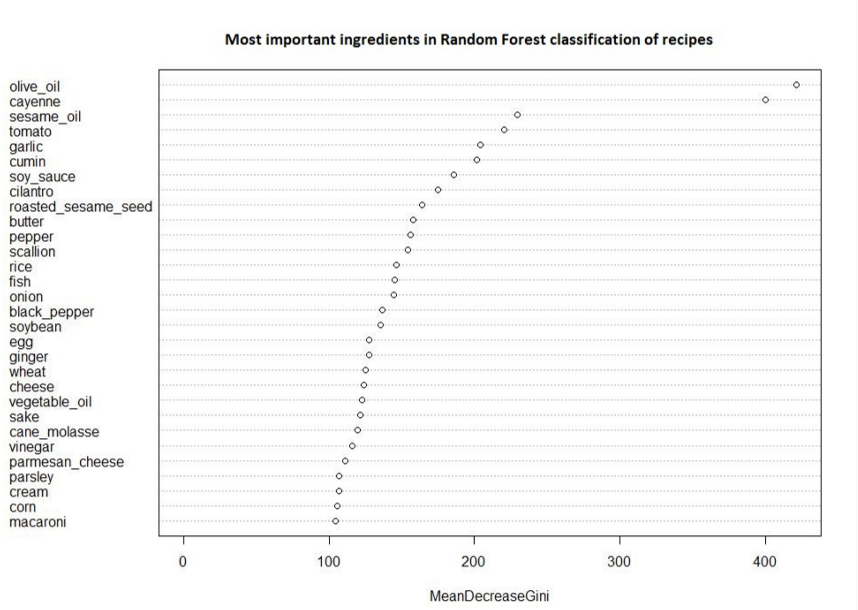


Figure 4.3: Most important ingredients to predict the origin of the recipes according to the random forests classifier.

Naive Bayes versus random forests

The naive Bayes approach differs from the random forests approach in the fact that the naive Bayes approach assumes conditional independence, meaning that the feature values, here the ingredients, are treated as independent, conditioned on the class, here the origin of the recipes (Bishop (2006)). This is not the case in the random forests approach. The fact that random forests scores better may suggest that the different ingredients are not independent, but that the presence of certain ingredients depends on the presence or absence of other ingredients.

Predicting the origin based on the total number of ingredients per recipe

To study whether the origin of the recipes can be predicted based on the number of ingredients present in each recipe, a random forests classifier is used. The classifier will have the origin as response and the number of ingredients as feature. The classifier is built and trained with three quarters of the data after which it is tested with the remaining quarter of the data. To study the accuracy of the model, a confusion matrix of the predicted and actual origin is built and the accuracy is determined.
The origin of the recipes cannot be predicted based on the number of ingredients found in the recipes. The model classifies each recipe as coming from North America. It can be concluded that there is too little difference between the different regions when looking at the number of ingredients per recipe.

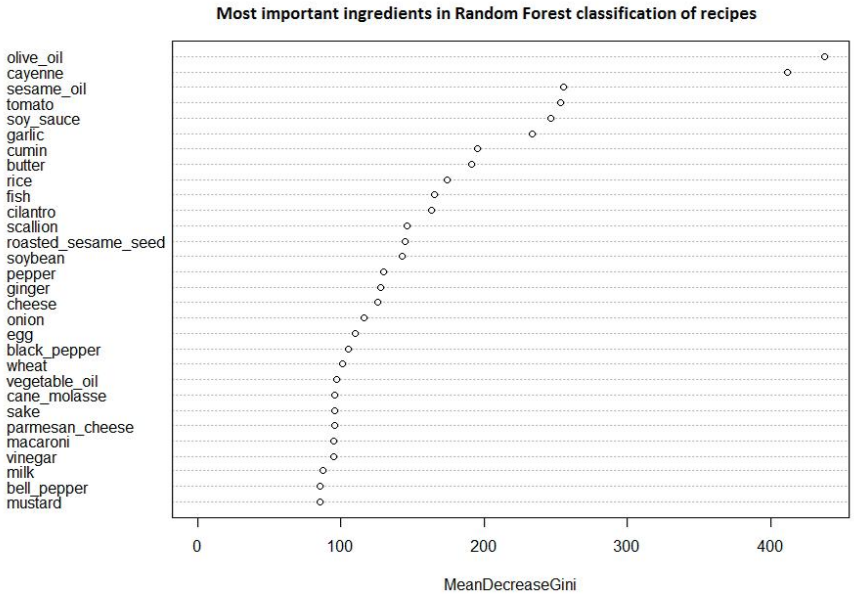


Figure 4.4: Most important ingredients to predict the cluster number (heat map in Figure 4.2) of the recipes according to the random forests classifier.

4.3.2 Flavour compounds in ingredients

The aim in this section is to determine whether or not the ingredients can be classified into their own category based on the flavour components that are present in these ingredients. The binary data set containing the different flavour components as features and the ingredients as observations, will be ideal to study this objective. The models that are built to analyse the ingredients, have the 1,107 flavour components as variables and the fourteen ingredient categories as response. Because of the large number of flavour components, it will be difficult to visualize a heat map, therefore a dendrogram will be built based on hierarchical clustering instead of a heat map. This dendrogram will tell which categories are more alike than others. In a next step a naive Bayes model and a random forests classifier are built and trained with three quarters of the ingredients of each category, after which their classification accuracy is determined by testing the models with the remaining one quarter of the ingredients of each category and comparing the predicted and actual categories.

Dendrogram based on hierarchical clustering

To perform a hierarchical cluster analysis on the categories, a preprogrammed function is used (`hclust` from the `stats` package in R). First, each column of the data set is scaled. This will result in a better clustering, because the number of times a flavour component appears is otherwise taken into consideration as well. The `hclust` function calculates the Euclidean distance between the different ingredients based on the values of the flavour components for

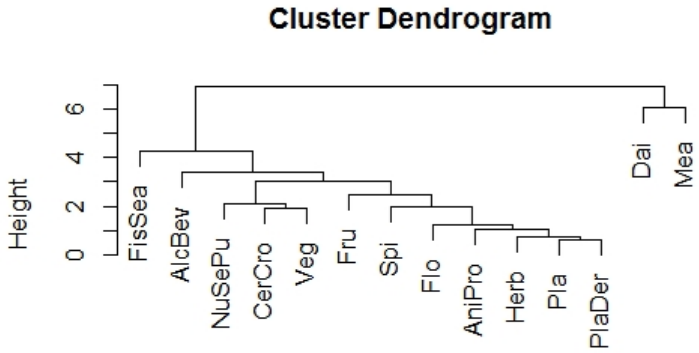


Figure 4.5: Dendrogram of the ingredient categories based on the flavour compounds of the ingredients.

each ingredient. The Euclidean distance is then used to cluster the categories. The hierarchical clustering is done based on the complete linkage method. The hierarchical dendrogram is built in such a way that at each node, the cluster that goes to the left is tighter than the cluster that goes to the right. This means that the value of the merge on the left will be lower than the value of the merge on the right. The height represents the distance of the furthest neighbor, or in other words, it is the distance between the two most dissimilar categories.
The result of the hierarchical clustering can be found in Figure 4.5. There are two clear groups in the dendrogram. The one on the right contains the meat and the dairy category; the one on the left contains the remaining categories. The order of the categories in which they are separated from the remaining categories is quite logical. First meat and dairy are separated followed by the fish and seafood. These are all ingredients of animal origin. The only category also belonging to this group is animal products which contains ingredients like honey, gelatin... But apparently the flavour components present in this category are quite different from those found in the other animal ingredients. The remaining categories are from vegetable origin. From left to right there are first the fruit and root parts of trees and plants, like nuts, seeds, cereals, fruit, etc, followed by the flower and leave parts and finally the whole plant. The two categories that are most alike based on the flavour components are the plants and the plant derivatives. Those will probably be the two categories that are most difficult to distinguish by the predictive models.

Naive Bayes

The ingredient category forms the response of the naive Bayes model and the different flavour components the variables. For this model it is not necessary to scale the data first. After building and training the model with the train data, it is tested with the test data. For each category the number of ingredients belonging to this category is given to the model. This is to

give weight to the categories, so each category has an equal part in the model. The model has a classification accuracy of 0.442. Based on the 14 categories, the accuracy of the predictions would be approximately 0.071 if the classification would be done randomly. As the accuracy of the trained model is higher than 0.071, the categories can be predicted quite well based on the components present in the ingredients of each category. The distribution of the confusion matrix is rather good. In the previous section, it was predicted that the model would have most difficulty in distinguishing plant derivatives from plants, since those two categories have the most similar flavour composition. This is confirmed here, as the largest confusion occurs between the plants and the plant derivatives.
When looking at the posterior probabilities of the categories, it is clear that ingredients of some categories have most of the time the same flavour components, while the ingredients of other categories do not. The maximum posterior probability for the meat and dairy categories is 81% and 85%, respectively. For dairy only one component has a probability of 85%. This component is valine, an essential amino acid typically found in protein-rich products like dairy and meat. But for meat 21 components are found with a posterior probability of 81%. When those components are present, the probability is rather high that the ingredient will belong to the meat category. This explains why these two categories are first separated from the remaining categories in the dendrogram. The lowest posterior probability is found for plants and plant derivatives. The categories have a posterior probability of 5.4% and 9.8%, respectively. This means that there are no flavour components that are typical for these categories, making it much more difficult to identify the ingredients belonging to these categories.

Random forests

The random forests classifier has the category as response and the flavour components as features just like the naive Bayes model. It is built and trained with the same training data as for the naive Bayes model. After which it is tested with the same test data. A confusion matrix is built to compare the predicted categories with the actual categories. The classifier has a classification accuracy of approximately 0.57. This accuracy is higher than the one of the naive Bayes model. Apparently the random forests classifier can better approximate the reality than the naive Bayes model.
The values in the confusion matrix are quite well distributed. However, the category of plant derivatives contains the highest number of ingredients that are incorrectly classified: ingredients not belonging to this category are identified as plant derivatives. Most of the times it are actually ingredients belonging to the plant category but the random forests classifier also places some herbs and fruits into the plant derivatives. In the naive Bayes model the category having the highest number of ingredients that were incorrectly classified were the plant category and not the plant derivatives.
For dairy and meat there are no incorrect classifications in both ways: all the ingredients

belonging to these categories are classified into the correct category and no other ingredients are classified into these categories as well.
It can be concluded that the ingredient category can be predicted quite well based on the different flavour components and that the random forests classifier is more suitable to perform this task than a naive Bayes model.

Predicting the category based on the total number of flavour components present in the ingredients

Since the random forests model had the best classification accuracy, this technique is used to study whether or not the category can be predicted based on the total number of flavour components present in the ingredients. The classifier has the variable category (containing the 14 categories) as response and the variable total (containing for each ingredient its total number of flavours) as feature. Just like before three quarters of the data is used for training and one quarter for testing the classifier. The predicted categories are compared with the actual categories in a confusion matrix and the classification accuracy is determined.
The model has a classification accuracy of approximately 0.34. This means the categories cannot really be predicted based on the number of flavour components present in the ingredients. So, it is better to use the different flavour components, since then the classification accuracy is larger.

4.3.3 General conclusion

The data set of the recipes contains a lot of recipes coming from North America. The main reason here for is the way of collecting the ingredients. Ahn et al. (2011) used three online websites (*allrecipes.com, epicurious.com, menupan.com*) and downloaded all the recipes. When a recipe was classified as belonging to an ethnic cuisine, it was placed in the corresponding group. But not all the recipes where labeled with an origin.
When looking at several of the North American recipes, they do not seem to have their origin in North America at all. For instance, one recipe contains peanut oil, tomato, pepper, onion, beef, cayenne, ginger, garlic, rice and turmeric. These ingredients rather suggest an Asian recipe (or maybe an African recipe). So does the recipe containing vinegar, meat, vegetable oil, soy sauce, rice, oyster and seaweed. Other recipes are more likely to come from Southern Europe. An example is a recipe containing tomato, cheese, olive oil, parsley, macaroni, basil, garlic, egg and bread, which reminds of Italy.
For this reason the reclassifications done by the random forests classifier or the naive Bayes model are probably not that bad for recipes labeled as being North American. There even is a chance that the *new* origin of these recipes is better than the original. However, because of the large number of North American recipes in the data set, and the questionable origin of some of these recipes, a lot of the recipes labeled as not coming from North America are

now classified as if they did come from North America. This is probably worse than before the reclassification.
Therefore it could be better to only reclassify the recipes labeled as coming from North America, instead of relabeling all the recipes.
This reasoning is more difficult for the ingredient categories. Since classifying ingredients into categories is based on science, it is not possible to claim that a vegetable should be classified as a fruit, a cereal or as an alcoholic beverage. However, it could be stated that the difference between a plant, herb, vegetable... is rather small, as a herb is actually also a plant and a vegetable is a part of a plant. And that the classification into the categories normally is not done based on flavour components. Therefore, it could be possible to reclassify the ingredients into different groups, not based on type of ingredient, but based on flavour components present in the different ingredients.

Chapter 5

Unsupervised learning techniques

5.1 Objective

In this chapter the objective is to group the ingredients that go well together. For the recipe data (Section 4.2.2), this means grouping together those ingredients that are frequently used together in a recipe. With the flavour data (Section 4.2.3), the ingredients containing a comparable flavour composition are grouped together. As indicated in Chapter 3, ingredients having similar flavour components often form a tasteful combination. The clustering of the ingredients into groups will be a first indication of the results that could be expected with the predictive model that will be built.

5.2 Combinations of ingredients in recipes

In this section, the data set containing information about the combinations of ingredients in the recipes is used (Section 4.2.1). This data set has the 381 different ingredients both as features and as observations and gives for each combination of two ingredients the number of times these ingredients are found together in a recipe. What makes this data set suited to study which ingredients go well together, and which do not, based on the existing recipes. These are ingredients that go well together not only in theory but also in practice. These combinations will be interesting for the final model that will be built.

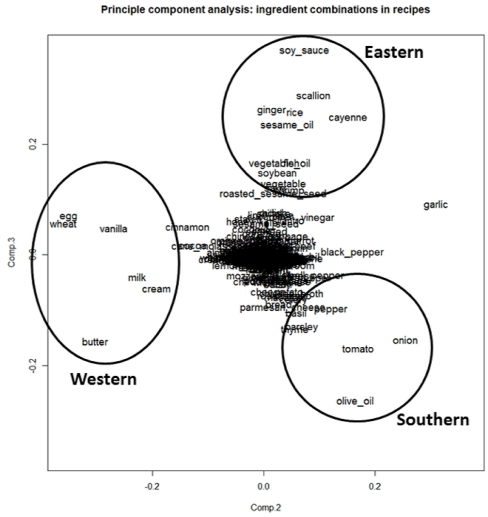
5.2.1 Principal component analysis

Information on the principles of principal component analysis (PCA) can be found in Barber (2012b). To determine the principal components of the data set, a preprogrammed function, written in R, is used (`princomp`). The function will calculate 381 principal components, the same number as ingredients.
First the columns of the data are scaled. The `princomp` function is used to calculate the principal components. To be able to analyze the results, two of these components will be

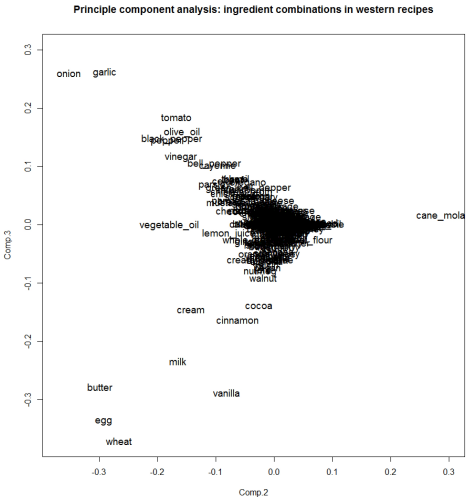
plotted against each other, to visualize the results. This plot will make it easier to divide the ingredients into groups. Since the first component points in the direction of ingredients with the largest variation instead of correlation (e.g., the ingredients that appear more or less frequently in recipes), this component is not a relevant choice when the aim is to group the ingredients based on similar characteristics. The second and third component are more appropriate choices. These components usually contain more information about characteristics that are present in some observations but not in other. This makes it easier to divide the ingredients in groups. Figure 5.1(a) shows the graph that is obtained by plotting the second and third components. In a next step the data set is divided into subsets, based on the origin of the recipes and the groups found in the dendrogram (Section 4.3.1). The dendrogram is used to divide the regions into four groups. Three of these groups contain more than one region and will be studied. Those three groups represent the Southern regions, the Western regions and the Eastern regions:

- Western: Northern Europe, Eastern Europe, North America, Western Europe;
- Eastern: East Asia, Southeast Asia;
- Southern: Latin America, Southern Europe, the Middle East, Africa.

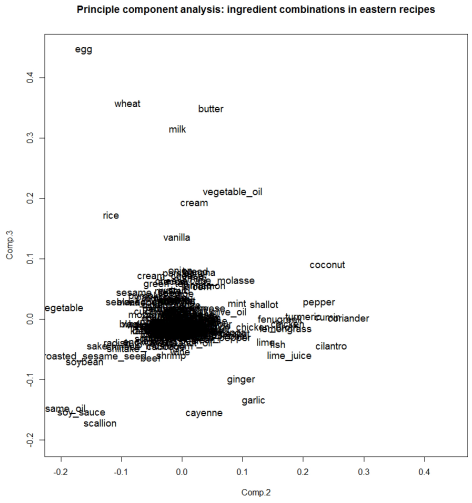
The Western subset consists of 45,196 recipes, containing 369 ingredients. The Eastern subset contains 2,969 recipes that are composed of 263 different ingredients. The third subset includes the 8,094 Southern recipes containing 318 ingredients. Apparently the number of different ingredients used in recipes is smallest for the Eastern recipes. For each of the three subsets principal components are calculated and the second and third principal component are plotted. The graphs can be found in Figure 5.1.
When looking at Figure 5.1(a) there are three groups of ingredients that are isolated from the others. The first group, located on the left, contains egg, wheat, vanilla, butter, milk and cream. These ingredients are typical found in recipes coming from Western regions. The second group, in the upper part of the graph, contains ingredients like soy sauce, scallion, ginger, rice, cayenne and sesame oil, which are mostly found in Eastern recipes. The group on the bottom of the graph contains ingredients that are typical for Southern recipes, like olive oil, tomato and onion.
The graphs for the different parts of the world prove that there is a difference in combining ingredients in the different parts. The graph of the Western recipes still shows clear groups of ingredients (bottom left: egg, wheat, butter, milk, vanilla; top left: onion, garlic, tomato, olive oil...). However, the other two graphs do not show clear groups. The graph of the Eastern recipes still shows some outliers, but the graph of the southern recipes is closely packed and most of the ingredients are located around the origin. Knowing this, it could be useful to build different predictive models for the different parts of the world (as done in Section 6.5.3), using only the recipes belonging to each part of the world.



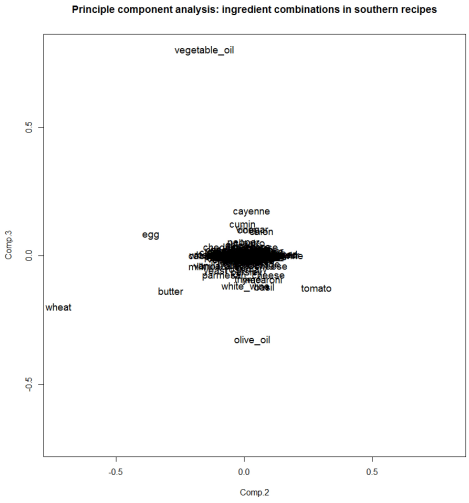
(a) World



(b) Western recipes



(c) Eastern recipes



(d) Southern recipes

Figure 5.1: Principal component analysis of the recipes. Figure (a) is constructed using all the recipes present in the recipe data set. Figures (b), (c) and (d) represent the second and third principal component when the analysis is performed with only a part of the recipes based on the origin of the recipes.

Kernel principal component analysis

Principal component analysis can also be used to analyze kernel matrices. The data set used in this section takes the form of a graph. The ingredients form the vertices of the graph. Two ingredients are linked together when they are used together in a recipe. The weights of these edges are the number of recipes in which both ingredients are present. Graphs have their own type of kernels; one example is the diffusion kernel (Kondor and Afferty (2002), Karatzoglou et al. (2004)).
The data set has almost the form of the operator of the diffusion kernel:

$$H_{ij} = \begin{cases} 1, & \text{for } i \sim j \\ -d_i, & \text{for } i = j \\ 0, & \text{otherwise,} \end{cases} \tag{5.1}$$

with d_i the number of edges connected to vertex i. The diffusion kernel is calculated as follows:

$$K = e^{\beta H}. \tag{5.2}$$

So to obtain the diffusion kernel, you have to calculate the matrix exponent of βH. Instead of using the value 1 when two ingredients are linked together ($i \sim j$), the number of common recipes can be used, to get a weighted kernel. So the only thing that has to change in the data set is the value of the diagonal items. To determine a good value for the hyperparameter β of the diffusion kernel, the principal components of the kernel matrix are calculated and plotted. The value of β that divides the ingredients best is chosen (Figure 5.2).
The ingredients are less isolated from each other, but have a better dispersion. However, the ingredients that are near to each other can likely be exchanged in recipes. For instance, vanilla can be exchanged with cocoa, mussel with oyster or clam and cured pork with pork sausage.

5.2.2 Spectral clustering analysis

The spectral clustering method groups the data points into a specific number of clusters. The difficulty of this method is determining the number of clusters that represent the data well. Especially since it is an unsupervised learning technique, which means that the accuracy of the clustering cannot be tested afterwards.
One possibility to estimate the number of clusters is by looking at the within cluster sum of squares (wss). This value tells something about how closely the clusters are packed. This value should be as small as possible. However, when taking too many clusters there is a risk of overfitting the data.
The data is clustered into different numbers of clusters and for each clustering the within cluster sum of squares is determined. These values are plotted against the number of clusters,

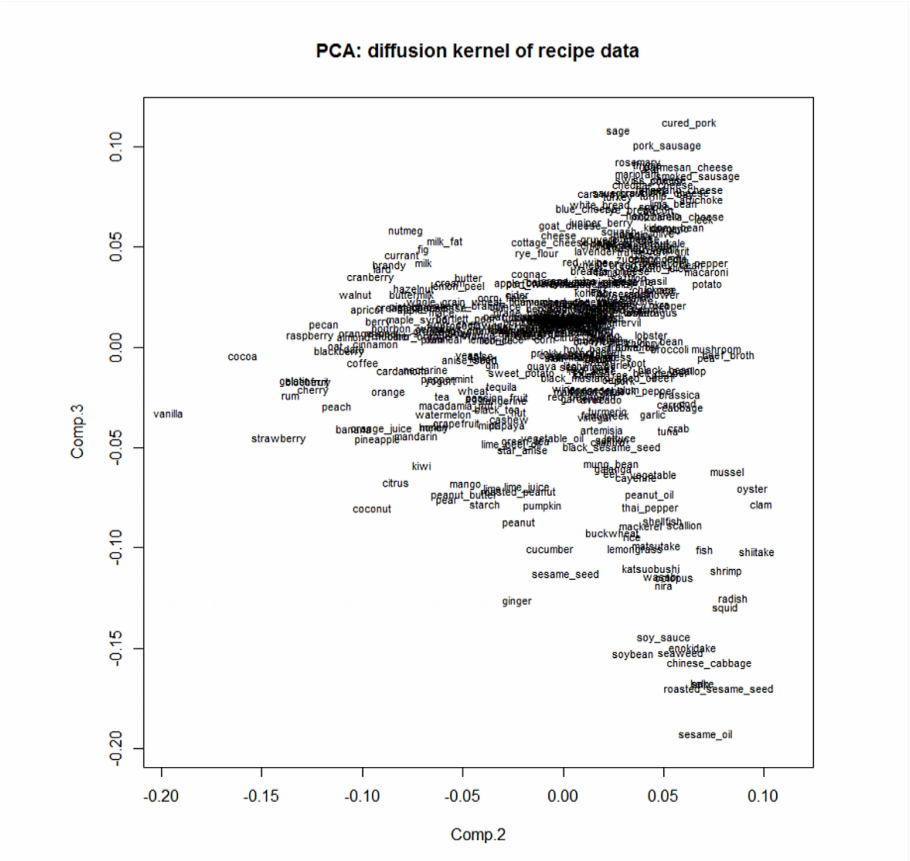


Figure 5.2: Kernel principal component analysis using a diffusion kernel of the recipe data.

resulting in a curved graph (Figure 5.3). The decision is made to divide the data into six clusters, since the within sum of squares will only decrease a little when the number of clusters is increased. The ingredients that are grouped together are ingredients that are typically used together in recipes.

When using the spectral clustering technique, the clustering is not deterministic. This means that results of the analysis can change every time the analysis is performed, or, in other words, some ingredients will change groups when the data is clustered into the same number of clusters several times in a row. Therefore it is important to determine the stability of the clustering. The stability is measured by clustering the ingredients into six clusters for thirty times. After each clustering the ingredient combinations in each group are saved in a matrix. This matrix contains 381 rows and columns, the same number of ingredients. When two ingredients are clustered into the same group the value in the box containing the first ingredient as row name and the second ingredient as column name will be increased by one. After the thirtieth clustering this matrix gives an idea of the stability of each cluster. When a cluster is stable, the ingredients in this cluster will be clustered together thirty times, when

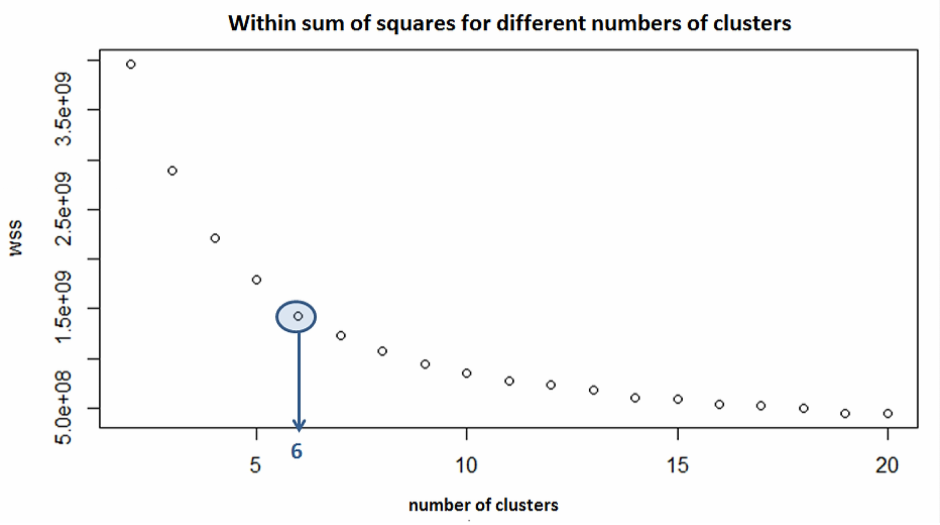


Figure 5.3: Determination of the number of clusters for spectral clustering of the ingrendient combinations in recipes.

the clustering is unstable this value will be lower, since the ingredients will not be grouped together after each clustering.
There are some stable groups, for instance the group containing butter, milk, eggs, wheat and vanilla. After each clustering these ingredients were found in the same group. Twenty-six times (of the thirty) cream was found in this group as well. Another example is the group of black pepper, pepper, cayenne, garlic, olive oil, onion, tomato and vinegar. This indicates that these are also ingredients that are used together in a lot of recipes, thus likely form a good combination. Also soy sauce, ginger, rice, scallion, beef, thyme, parsley, green bell pepper, etc. were each time found in one group. It can be concluded that some combinations of ingredients are very stable; these are ingredients that have already been used together a lot. These groups can be used to suggest a new ingredient to an existing recipe: when ingredients out of these groups are already present in the recipe, the other ingredients will probably lead to a good combination as well.

5.3 Comparison of ingredients based on their flavour profile

In this section it will be determined whether the different types of ingredients can be isolated from the others, based on their flavour components. For this study the binary data set, containing the ingredients as observations and the flavour components as variables/features, will be used. The data columns are scaled to get more correct results. The aim of the following examinations is to find the ingredients with a similar flavour composition, because those ingredients will have a similar taste and could lead to tasteful combinations. Just like before the techniques that will be used to study this case are principal component analysis

and spectral clustering analysis.

5.3.1 Principal component analysis

As mentioned before a preprogrammed function is used (`princomp` in R) to calculate the 1,107 principal components, the same number as flavour components. Just as before the second and third principal component are plotted against each other, to visualize the results. The graph will make it easier to divide the ingredients into groups based on the difference in flavour components. This graph can be found in Figure 5.4.
The graph shows that, by using the principal component analysis, at least some ingredients can be divided into different groups based on the flavour components that are present in the ingredients. The different types of tea are isolated in the upper left corner. The alcoholic beverages and the different types of cheese are separated in two groups in the lower half of the graph. The right upper arm of the graph contains the different types of meat. The berries and other types of fruit can be found in the left part of the graph. Most of the ingredients, however, can be found in the middle of the graph. These ingredients have a too similar flavour composition and cannot be separated from each other in this two-dimensional subspace. In contrast with the previous section, the ingredients are now grouped together per category.
The data set contains 23 different types of cheese: blue cheese, Camembert cheese, cheddar cheese, cheese, Comte cheese, cottage cheese, cream cheese, domiati cheese, emmental cheese, feta cheese, goat cheese, etc. All these types of cheese contain 266 flavour components, from which 127 are present in all 23 types of cheese. The other flavour components are present in 7, 6, 5, 4, 3, 2 or 1 type of cheese. Those 127 common flavour components are the reason why the types of cheese can be grouped together and can be separated from the other ingredients. However, since they do not have an identical composition, it should be possible to see the different types of cheese in the graph, since they are not plotted on the exact same spot. One possibility is to do a new principal component analysis by using only the data of the cheese. Unfortunately, this is not possible since there are 266 features (=flavour components) and only 23 observations (= types of cheese); and a linear principal component analysis can only be done when there are at least as many observations as features. A solution could be to only look at the flavour components that they do not share, however, this still results in 139 features. The only option to see the difference between the types of cheese is to isolate the part of the graph that contains the different types of cheese and magnify it. This is done in Figure 5.4.

Kernel principal component analysis

Principal component analysis can also be used to analyze kernel matrices. Firstly the data set is transformed into a kernel matrix after which it is analyzed using PCA. In R, a preprogrammed function exists for this method as well (`kpca`). The data set and the type of kernel

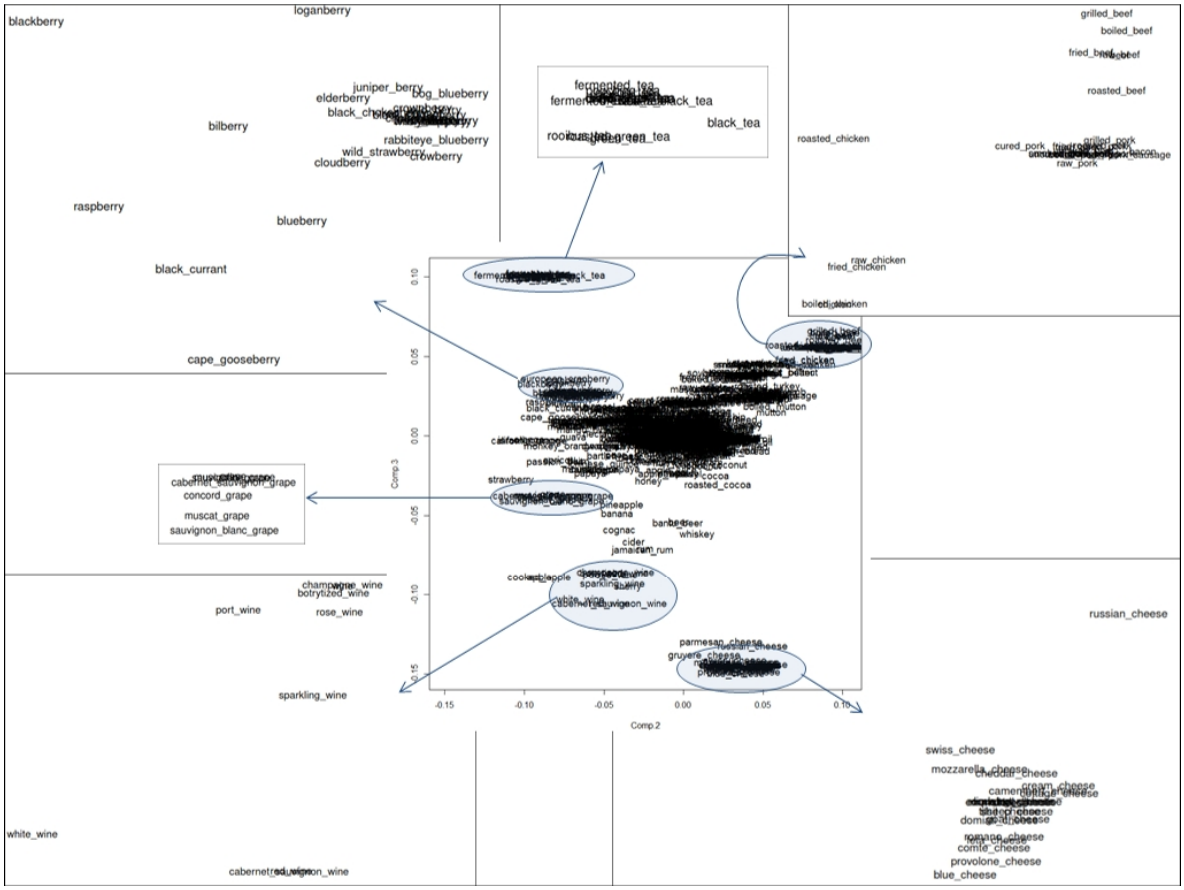


Figure 5.4: Principal component analysis of the ingredients based on their flavour components.

(linear, Gaussian, etc.) are given to the function, which returns the principal components. Just as before, the second and third principal component are plotted to examine the results. This function allows to determine which type of kernel describes the data best.

In Figure 5.5 each category has its own color. The alcoholic beverages are given in red, the animal products in yellow, the cereal and crops in dark blue and so on. This graph shows the results of kernel PCA after scaling and transforming the data into a linear kernel (`vanilladot`: $\langle x,x' \rangle$). All 1,525 ingredients are used to determine the principal components, but only the 381 ingredients present in the 56,498 recipes are shown in the graph. This way both the advantages of a large data set (more accurate model) and a small data set (readable graphs and results) are exploited.

From the graph it is clear that certain categories (or parts of these categories) can be isolated from the others. At the bottom of the graph there is a group of dairy products. Those are the different types of cheese. Above the line of ingredients there is a group of plant derivatives, namely the different types of tea. At the end of the line there are five isolated meat products:

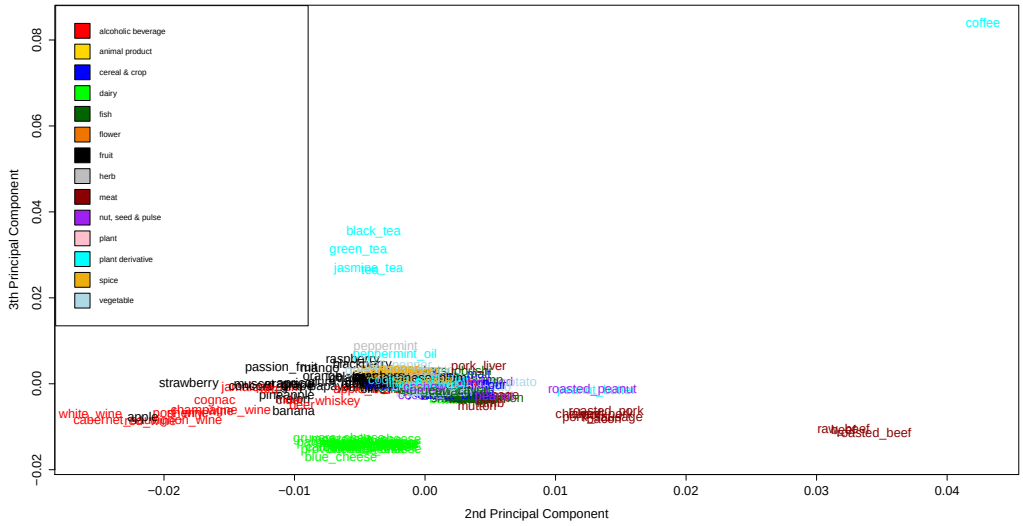


Figure 5.5: Linear kernel principal component analysis of the ingredients based on their flavour components.

all the types of beef. At the beginning of the line the alcoholic beverages are grouped together, but the ingredient that is most isolated from all the other is coffee in the right upper corner. The ranking of the categories in the line of ingredients is rather logical. From left to right: the alcoholic beverages, fruit, species and plant derivatives, herbs, flowers and plants, nut/seed/pulse, grain/crops, vegetables, dairy and animal products, fish/seafood and finally meat. There is a transition from vegetable products to animal products.

When the Gaussian kernel (`rbfdot`: $e^{-\sigma\|\times-\times'\|^2}$) is used in the kernel principal component analysis, the data is represented very poorly. Most of the ingredients are plotted at the same spot, because of some outliers even when the data is standardized before being transformed into a kernel matrix. When the outliers are removed from the data, the remaining ingredients are lying on a curve. However, the ingredients seem to be distributed over the curve randomly and no logical ranking can be identified. It can be concluded that the Gaussian kernel is not appropriate to describe the data. This is not that surprising since the Gaussian kernel represents a continuous function, while the data is discrete. The kernel `polydot` $((\text{scale}.\langle\times,\times'\rangle+\text{offset})^{\text{degree}})$ gives almost exactly the same results as the linear kernel. This type of kernel could also be used to describe the data. The Laplace kernel (`laplacedot`: $e^{-\sigma\|\times-\times'\|}$) results in a rather similar graph as the Gaussian kernel; this type of kernel is just like the Gaussian kernel, not suitable to describe the data.

5.3.2 Spectral clustering

A preprogrammed function (`specc` in R) is used to divide the ingredients into groups using a spectral clustering technique. The function not only needs the data set but also the number of

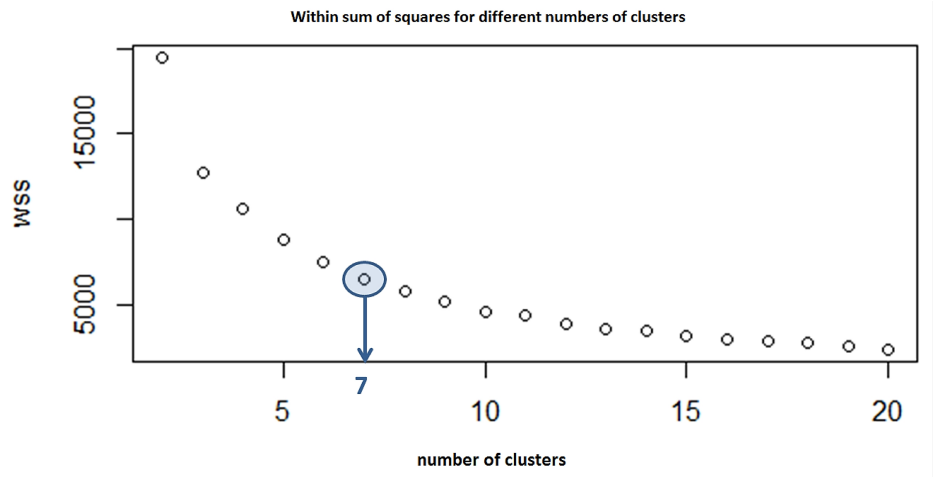


Figure 5.6: Graph to determine the number of clusters for the spectral clustering.

clusters that should be used. However, with this type of data, it is very difficult to determine the number of clusters.

Just as before the within sum of squares is determined for different numbers of clusters. The values are plotted against the number of clusters, resulting in a graph with a curved form (Figure 5.6). Both the within sum of squares and the number of clusters should be as small as possible to cluster the data well but also prevent overfitting. The data is clustered in seven clusters, since the within sum of squares of eight is not that much smaller than that of seven clusters.

Also for this data set the stability of the clustering is determined by clustering the ingredients thirty times into seven clusters and keep track of the number of times two ingredients are grouped together. The cluster stability is much lower than for the recipes. However, after each clustering round the different types of cheese are grouped together. Most of the times the different types of tea are found in one group as do the different types of meat. The types of liquor and wine are mostly found in the same group, together with the different types of grape. These ingredients are frequently found in the group containing the cheese. Thus, fermented foods are lumped together.

The stability matrix is filled with rather low values, meaning that the clustering is not that stable resulting in different groups after each clustering. It can be concluded that spectral clustering is not a good method to cluster the ingredients based on their flavour components.

5.4 Comparison of the two data sets

To study whether or not there is a relation between the flavour components present in the ingredients and the combinations of the ingredients found in recipes, a kernel canonical corre-

lation analysis is performed. The canonical correlation analysis is a data reduction technique that looks for a pair of linear subspaces that have high cross-correlation for two or more variables, in such a way that each component from one subspace is correlated with a single component in the other subspace (Bishop (2006)). In other words, the technique searches for a sequence of uncorrelated linear combinations from one dataset (the recipe data) and a corresponding sequence of uncorrelated linear combinations of a second data set (the flavour data) that leads to a maximum correlation between these two sequences (Friedman et al. (2008)).

For the data of the flavour component the same binary data set as in the section above is used. However, for the recipes the diffusion kernel (Section 5.2.1) is chosen. The kernel used for the canonical correlation analysis is the linear kernel. To perform this analysis a preprogrammed function in used (`kcca` from the kernlab package in R). The function returns for both data sets estimated coefficients for the variables in the feature space. Figure 5.7 shows a scatter plot of the first canonical variates of each data set. The values for the recipes can be found on the y-axis and those for the flavour components on the x-axis.

The distribution of the ingredients in the scatter plot shows a diagonal shape, indicating that there is a correlation between the flavour components and the use of the ingredients in recipes. The distance of each ingredient to the diagonal is calculated. Ingredients that are close to the diagonal possess a rather specific taste, resulting in the fact that they can only be combined with a limited number of ingredients. Examples are mussel, cognac, fig, star anise, etc. Ingredients with the largest distance to the diagonal are onion, butter, brown rice, bell pepper, chicken, egg, etc. All these ingredients are used in a lot of different combinations. Butter is not only used on bread but also to bake meat, vegetables or to make cake or cookies, indicating that the use of this ingredient is not related to its flavour, but rather to other properties like structure, melting point, etc. Egg is known for its emulsifying properties and its ability to create foam, and just like butter it is used in all kinds of recipes.

It can be concluded that in most cases the combinations of ingredients in a recipe are depending on the flavour composition of those ingredients. However, flavour is not the only factor determining the choice of ingredients, as discussed in Chapter 2. Some ingredients are added for other properties like emulsifying properties, structure properties, melting properties, nutritional properties, etc.

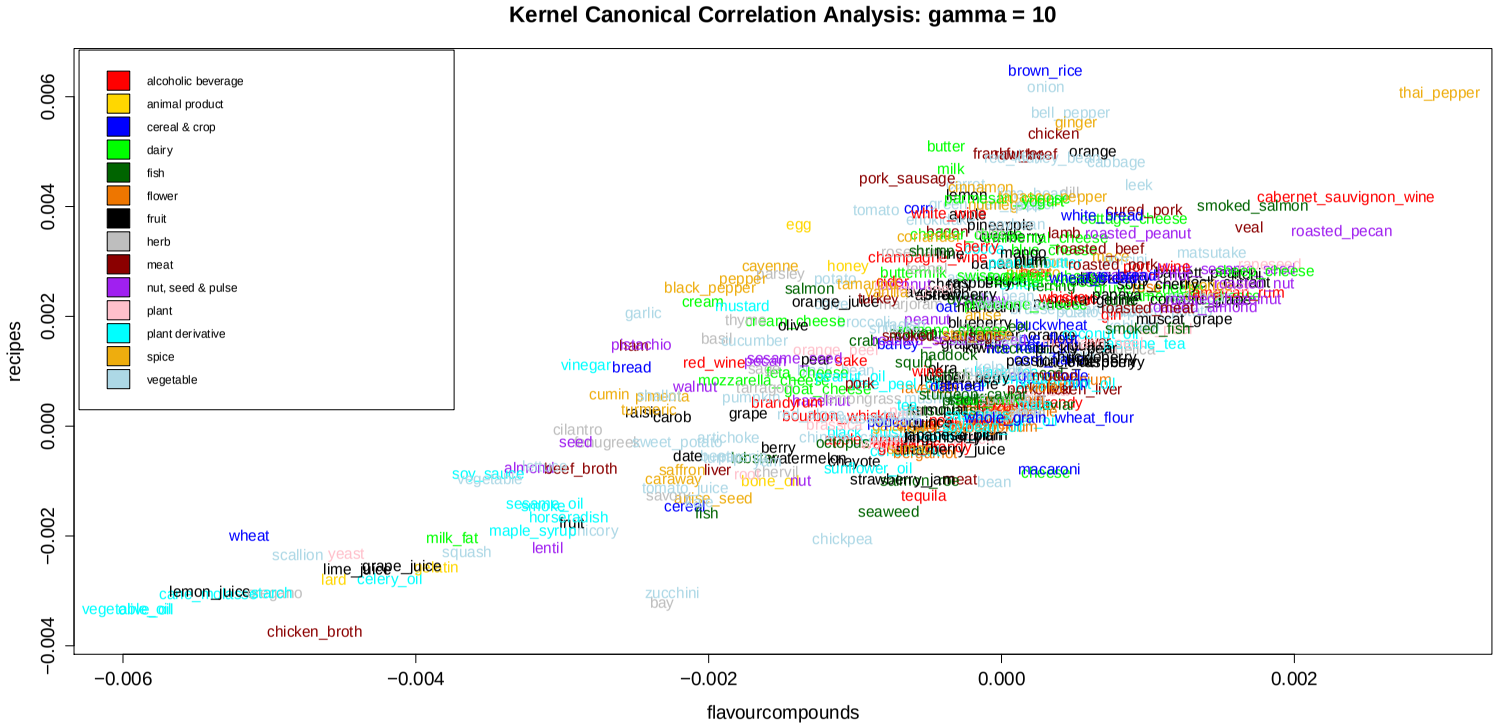


Figure 5.7: Kernel canonical correlation analysis of the recipes and the flavour components.

Chapter 6

Matrix decomposition

In this chapter, a first predictive model will be built to find good ingredient combinations. In this attempt, only the ingredient combinations found in the recipe data will be used to predict good combinations. Later, the information found in the flavour data will be added to the model, since ingredients with the same flavour compounds are supposed to go well together. The method used to build the predictive model in this chapter is matrix decomposition.

6.1 Data

The model will be built using the binary data set containing 56,498 recipes (Section 4.2.1). The recipes containing only a single ingredient are left out of the data set, since those recipes do not contain any information about ingredient combinations and so will not be useful during the model building sections. Another reason to eliminate those recipes is that there will be no remaining ingredients after erasing one of the containing ingredients, for model selection and evaluation. The total number of recipes containing only a single ingredient is 354, resulting in a data set of 56,144 recipes to build the models.
In a first step 240 recipes are randomly chosen from the 56,144 recipes. These recipes are then divided into twelve groups of twenty recipes. Eleven of these groups form the train and tune data. The remaining group is the test data. The model building will be explained is more detail in Section 6.3.

6.2 Used techniques

There exist several types of matrix decomposition. A visual representation of a matrix decomposition technique can be found in Figure 6.1. Matrix decomposition is an approximation of a matrix by multiplying two matrices with a low rank. It is difficult to choose the most suitable method, since the structure present in the data is still unknown. Therefore three different types of matrix decomposition are examined and compared. The aim is to find the method

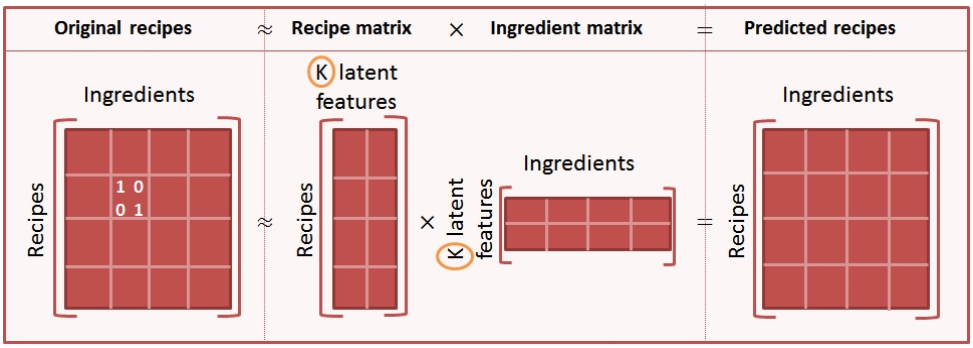


Figure 6.1: Schematic representation of matrix decomposition. In a first step the original matrix is split into two matrices, the Recipe matrix and the Ingredient matrix. The dimensions of these two matrices are determined by the number of latent features, K, chosen during the decomposition. In a second step, the product of these two matrices is calculated resulting in a new matrix, similar to the original matrix.

that is most appropriate to predict ingredient combination based on information found in existing recipes. By training and testing a model using each of the three different techniques of matrix decomposition, the type of matrix decomposition that predicts the ingredients in the recipes best can be found. This final model will be studied in more detail. The three methods that will be studied are:

- Singular value decomposition (SVD)
- Non-negative matrix factorization (NNMF)
- Independent component analysis (ICA)

6.2.1 Singular value decomposition

Friedman et al. (2008) tells that singular value decomposition is a standard decomposition method in numerical analysis. With this techniques the matrix Y is decomposed as:

$$Y \approx U\Sigma V^T, \tag{6.1}$$

where Y is an $m \times n$ matrix. In this work Y is the recipe data with m and n equal to 56,498 and 381 respectively. U and V contain respectively the left and right singular vectors of Y in their columns. U is an $m \times m$ orthogonal matrix (i.e. $U^TU = I_m$). V is an $n \times n$ orthogonal matrix. Σ is an $m \times n$ diagonal matrix and contains the singular values (d_i) of Y on its diagonal. The singular values are ordered from most important (explaining most of the variation found in the data) to least important: $d_1 \geq d_2 \geq ... \geq d_n \geq 0$.
In this work $U\Sigma$ delivers the *Recipe matrix* ($m \times n$), V^T gives the *Ingredient matrix* ($n \times n$)

in Figure 6.1. There are maximum n latent features. The number of features, K, will be changed by setting the $n - K$ last singular values in the Σ-matrix to zero. Leaving only the first K singular values to take part in the *recomposition* of the recipe matrix. When the decomposition technique used is SVD the K range goes from 2 to 100. This method is mostly used on Gaussian distributed data, and as the recipe data is not Gaussian, the chance that this method will be good at predicting ingredient combinations based on binary data is rather small. The principal component analysis is also based on singular value decomposition.

6.2.2 Non-negative matrix factorization

With non-negative matrix factorization the data and components are assumed to be non-negative, meaning that only positive values are found. The matrix Y is decomposed as:

$$Y \approx WH, \tag{6.2}$$

where Y is an $m \times n$ matrix, W is an $m \times K$ matrix and H is an $K \times n$ matrix. The value of K can be a lot smaller than n and m, where $K \leq \max(n, m)$. As mentioned before, none of the three matrices (can) have negative elements. To find the values of W and H, the following function is maximized:

$$L(W, H) = \sum_{i=1}^{m} \sum_{j=1}^{n} \left[Y_{ij} \log(WH)_{ij} - (WH)_{ij} \right]. \tag{6.3}$$

A big difference with the previous method is that only the first K features can be calculated. The method will just divide the information into the K features. A second difference is that the features are not ordered from most important to least important, so it is not possible to just take the first two features to account for the biggest part of variation. This should be taken into account when evaluating the results. This technique is also used in Lee and Seung (1999).

Although only K latent features need to be calculated, NNMF is a very time consuming method since it needs to do iterations to maximize Equation (6.3). Therefore the K range is kept very small. And the optimal value of K is found after several runs with different ranges of values of K. The K-range going from two to fifteen seems to give the best results, so this range will be used to build the model.

6.2.3 Independent component analysis

The independent component analysis assumes that the S_i components are *statistically independent*. The matrix Y is decomposed as:

$$Y \approx SB, \tag{6.4}$$

where Y is an $m \times n$ matrix. The ICA considers this matrix as a linear combination of non-Gaussian (independent) components. The S-matrix contains the independent components of

Y. The S-matrix is also known as the estimated source matrix. The B-matrix represents an estimated linear mixing matrix. $S = YCW$, with W a matrix that maximizes the negative entropy, or in other words maximizes the non-Gaussian character, which ensures that the estimations are uncorrelated. The C-matrix will limit the number of components, this will allow to only take into account the first K features as preferred in the model building.
This method allows to only calculate K features, just as the NNMF. The information found in the data is summarized in those K features. Just as in the NNMF, the features are not ordered based on their importance. The total range of K goes from 2 to 100, but it is divided into subranges of fifteen values to reduce the duration of the model building. The optimal range is found by running the model with the different subranges and selecting the range that delivers the best results. This range is used to build the model.
The previous chapter showed that the Gaussian kernel was not suited to represent the recipe data. This means that the recipe data does not follow the Gaussian distribution. Independent component analysis is used for non-Gaussian processes, meaning that this method should be more appropriate to predict ingredient combinations, than the singular value decomposition method, which is mostly used on data with a Gaussian distribution. The difference between singular value decomposition (used in PCA) and independent component analysis can be seen in Figure 6.2.

6.3 Model building step by step

The model building process can be summarized as follows: a recipe will be randomly selected from the data (train or test recipe) and one of its ingredients (randomly picked) will be erased from the list and saved in a new list. In a next step, the model will predict values for each ingredient. The ingredient with the highest value is selected as best fitting the given ingredients. In the ideal case the erased ingredient should be predicted, but it is also desirable if this ingredient occurs 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 train data groups are examined one by one, therefore only twenty recipes will be examined at the same time. Each of the twenty recipes is copied from the data set. The ingredients of each recipe are identified. For each of the twenty train recipes, one of the present ingredients is eliminated at random. The name of this ingredient is saved in the result matrix, as is the identity number of the recipe and its origin. After removing this ingredient from the recipe (x_{ri} : one $\rightarrow$ zero), the original recipe is replaced in the data set. As the data set is binary, mathematically this action results in the replacement of a single one with a zero in the row of the recipe and the column of the eliminated ingredient. The resulting modified data set differs thus from the original set with twenty *ones* that are replaced with *zeros*, distributed over twenty of the 56,144 rows. This small difference will likely have no effect on the matrix decomposition of the data set. This is the reason that only a small number of recipes is

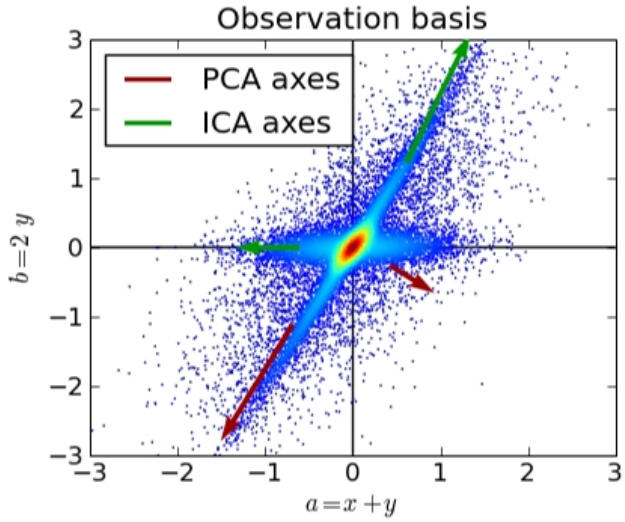


Figure 6.2: Difference between singular value decomposition (used in PCA) and independent component analysis. Both methods find the same first axis, along the longest line of data points, but not the same second axis. SVD (or PCA) finds a second axis perpendicular to the first axis. ICA however finds a second axis along the shorter line of points, which is in this case, a better representation of the data points (source: Varoquaux (2012)).

selected as training data. This data set forms the *original* recipes.

In a next step a decomposition of this data set into a product of two matrices is performed, as can be seen in Figure 6.1, by using one of the three matrix decomposition techniques that are enumerated in Section 6.2. These two matrices are called the *Recipe matrix* $(56,144 \times K)$, containing components with latent features of each recipe, and the *Ingredient matrix* $(K \times 381)$, containing latent features for each ingredient. K is the number of latent features that are calculated or taken into account. The maximum number of latent features is 381, equal to the number of ingredients in the data set. This value of K is the only parameter that can be changed to improve the predictive capacity of the model. The value of K is changed over a range of values, to determine the value of K, that leads to a model that can best predict the missing ingredient in the train recipes. Depending on the technique used to perform the decomposition of the matrix, the range of the values of K will change (Section 6.2).

The data set is reconstructed by making the product of the *Recipe* and *Ingredient matrix* (Figure 6.1). Since only K latent features are used to reconstruct the data set, the resulting matrix will not be binary. The value of K cannot be too small, however, it can not be too large either. When the value is too small, a lot of information would be lost and a too large K-value would result in overfitting of the data. The remaining ingredients in the test recipes that have a value equal to *one*, are not of interest since those ingredients are already present

in the recipe and so cannot be added anymore. The predicted value of these ingredients is changed to a very low value, making sure that they will not be suggested as preferred ingredient to combine with the remaining ingredients.
When the recipe had originally a value equal to *zero* for a certain ingredient, its new value will depend on the frequency with which the ingredient is combined with one, several, or all the ingredients present in the recipe. If the ingredient is frequently combined with (some of) the ingredients in the recipe, its value will be larger than when it is not frequently combined with the ingredients in other recipes.

6.4 Model selection and validation

6.4.1 Training and tuning

The model is trained for different values of K. After each training step, the prediction capacity will be evaluated based on the ingredient that was removed from the recipe. The twenty recipes in the train set are taken out of the predicted recipe matrix one by one and are examined. For each recipe, the ingredients are sorted based on their predicted value, from the largest to the smallest value. Since the predicted value of the remaining ingredients of the recipe are changed to a very low value, as mentioned in the previous section, these ingredients will be last in the sorted list. The higher the predicted value of an ingredient, the better this ingredient can be combined with the remaining ingredients in this recipe; the lower the predicted value is, the worse the combination will be. If the performance of the prediction is high, the eliminated ingredient should have a high predicted value, and thus this ingredient should be found in the top ingredients of this sorted list. The place of the eliminated ingredient in this list will be called the rank of the eliminated ingredient. In other words, when the eliminated ingredient has the highest predicted value, it will be on top of the sorted list and will get a rank equal to one.
In a next step the rank of the eliminated ingredient is determined for each of the twenty recipes in the train set. The mean of the rank of these twenty recipes is determined and linked to the value of K, used for the reconstruction of the recipe data in this training step. This action is repeated for each value of K considered. For each value of K, the mean rank (of the twenty recipes in the train data) is saved into a matrix. Once all the values in this matrix are found, the value of K with the lowest mean rank is selected to be the best value of K for this group of training recipes. In a next step, the twenty recipes of this training set are restored to their original form (eliminated ingredient: $x_{ri} : zero \rightarrow one$).
Once the recipe data is restored, the next set of train data is examined in just the same way. So this process is repeated ten times with the remaining ten groups of train data. Each set of train data delivers its own *best* value of K, resulting in eleven *best* values of K. The median of these values is determined and classified as *best* value of K.

6.4.2 Testing

The best value of K found during training is used to perform a decomposition and a reconstruction of the twenty recipes in the test group. Just as in the training phase, one ingredient of each test recipe is eliminated at random and saved in the result matrix. After decomposition and reconstruction the rank of the eliminated ingredient is determined. Also the top five of ingredients that can best be combined with the remaining ingredients are saved, as is the value of K found during training, the identity number and the origin of the recipes, the category of the eliminated ingredient and the rank of the eliminated ingredient if only ingredients of the same category (fruit, vegetables, spices,...) are taken into account.
This whole process of selecting 240 recipes and training and testing the model is repeated one hundred times (by means of the HPC-cluster), resulting in $2,000$ tested recipes and 100 *best* values of K.

6.5 Results

6.5.1 Performance measures

For each technique, the hundred runs (found after testing the models in Section 6.4.2) are collected in one single document, resulting in 2,000 test recipes and 100 values of K. The mean *best* value of K is determined for each technique. Besides the best value of K, two other parameters can be determined: the average general rank of the eliminated ingredient (2,000 values) and the average rank of the ingredient when only looking at ingredients of the same category. These values allow to determine the percentage of ingredients with a smaller rank than the eliminated ingredient. Thus resulting in a larger value in the predicted recipe and thus a better fit in the test recipe than the eliminated ingredient according to the tuned model (Table 6.1). These percentages will help to evaluate and compare the three techniques.
Position of the ingredient in general:

$$\text{average percentage general rank} = mean\left(\frac{\text{general rank of eliminated ingredient}}{\text{total number of ingredients}}\right). \quad (6.5)$$

Position of the ingredient in the category:

$$\text{av. percent. category rank} = mean\left(\frac{\text{category rank of eliminated ingredient}}{\text{total number of ingredients in that category}}\right). \quad (6.6)$$

For each general and category rank value the percentages of test recipes having this general or category rank is determined. For general rank these percentages are plotted against the general rank values and a cumulative curve is added to the graph as well. This last one allows to easily compare the three techniques visually and to determine the capacity of the techniques by for instance looking at the percentage of recipes with a general rank smaller or equal to ten. The latter value gives the percentages of test recipes for which the eliminated ingredient can be found in the top ten of recommended ingredients.

6.5.2 Comparing the three techniques

The results for the parameters mentioned above are given in Table 6.1 for each of the three techniques. They will be studied in more detail below.
The mean best value of K is the largest for the SVD and the smallest for the NNMF. However, these values do not say much about the performance of the techniques. They only say something about the number of components needed to capture the information on ingredient combinations hidden in the data set. Apparently NNMF can best compress this information and thus needs the smallest number of components to have the highest performance.
The average percentage general rank tells a lot more about the performance of the models. When comparing the techniques based on this parameter, it is clear that the NNMF can predict the eliminated ingredient best, followed by ICA. SVD has the worst performance. The NNMF has an average percentage general rank of 10.8%, which means that on average the eliminated ingredient can be found on the thirty-eighth place, when ordering the ingredients from best fitting the recipe to worst fitting the recipe. This is quite a good performance, when keeping in mind that predicting ingredient combinations is very difficult, since they depend on taste, flavour perception, texture, etc. (see Chapter 2). The difference in average percentage general rank between NNMF and ICA is almost equal to the difference between ICA and SVD (both approximately 18%). This may suggest that the performance of the different techniques also declines in a same degree. However to confirm this statement, the predictions of the three techniques will need to be studied in more detail.
The average percentage category rank follows the same trend as the average percentage general rank. The rank is best for the NNMF technique and worst for the SVD technique. These values should be smaller than the average percentage general rank, since the category ranks will be smaller than the general ranks as a lot of ingredients are removed from the results, since they don't belong to the same category. However, in Tabel 6.1 the average percentage category ranks are larger. This is because these values are biased by the small number of ingredients in certain categories. For instance, the category of animal products (like honey, gelatin, etc.) contains only five ingredients. When the eliminated ingredient has the lowest rank of these five ingredients, its percentage category rank is only 20% (1/5). The prediction is very good, but the percentage is rather high; this is why the value of the average percentage category rank is biased and should not be interpreted. However, the measure can still be used to compare the three techniques and to conclude that also here the NNMF techniques had the best performance. A better way to evaluate the category rank is by determining the percentage of recipes that have a category rank of one, or lower or equal to three. When the predictions are made with NNMF, 35.6% of the test recipes have a category rank equal to one. When we look at a category rank smaller than or equal to three, 62.7% of the recipes are covered. For ICA, these percentages are respectively 32.2% and 52.9% and for SVD these percentages are equal to 8.2% and 9.3%. These values give a much better view of the differ-

Table 6.1: Summary of the results for the three types of matrix decomposition (* 381 ingredients, ** 2000 test recipes).

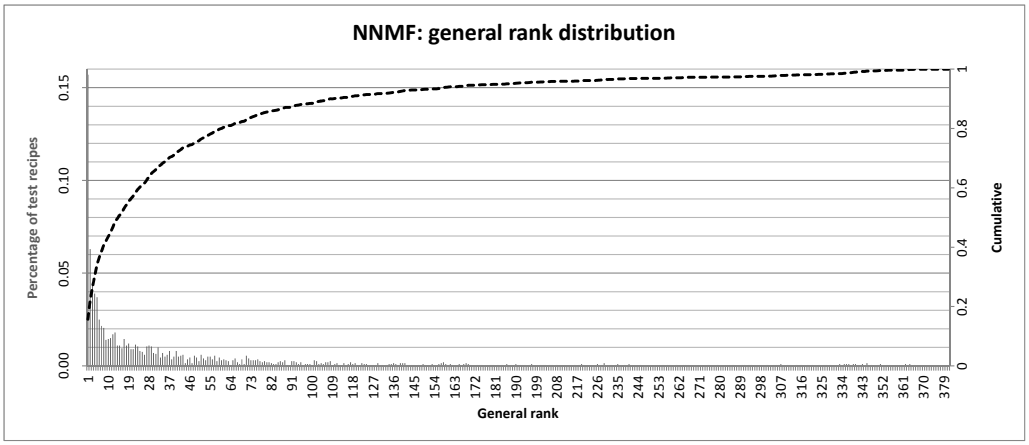
	All categories			Within category	
technique	average % general rank*	st.dev. general rank	% recipes: ingredients in top 10**	average % category rank	average value of K
NNMF	10.8%	17.5%	43.6%	15.8%	5.2
ICA	28.5%	38.7%	39.7%	31.9%	4.3
SVD	46.5%	29.6%	6.4%	48.6%	27.8

ence in predicting capacity between the different techniques. It is clear that the difference in performance is much smaller between NNMF and ICA than between ICA and SVD.
A similar value can be determined for the general rank. Since here all the ingredients are together in one group, the value of general rank is chosen higher than three. For the general rank, the percentage of ingredients having a general rank smaller or equal to ten is determined. Or, in other words, the eliminated ingredient can be found in the top 10 of best fitting ingredients. These values can be found in Table 6.1 as well. These values confirm what was found in the previous paragraph: NNMF has the best performance when predicting ingredient combinations in recipes. But the difference between ICA en SVD is much clearer than when just looking at the average percentage general rank. It is also very clear that SVD is definitely not a good technique to predict ingredient combinations.
This can better be seen when plotting the percentage of recipes with a certain general rank against the different values of the general rank. The plot of each technique can be found in Figure 6.3. Since there are 381 ingredients and each test recipe contains at least one ingredient, the largest general rank that can be found is 380. To be able to compare the techniques even better a cumulative curve of the recipes is added to the graphs as well. This curve shows for each general rank the percentage of recipes that have a general rank smaller or equal to this rank. When only looking at the cumulative curves, it is immediately clear that NNMF has the best performances of the three techniques. This cumulative curve increases exponentially in the small general ranks and quickly reaches a value close to one. The curve has a really low slope in the higher values of the general rank. This technique also has the highest percentage of test recipes with a general rank equal to one. The linear character of the cumulative curve of the SVD shows that the predictions done by this technique are almost done randomly. The graph of the ICA technique is very interesting. It shows that most of the predictions done are almost as good as NNMF, however the technique has problems with predicting the eliminated ingredient in certain recipes, since the percentages go up again when the largest ranks are reached. It is not clear what is so special about these recipes that the prediction of the eliminated ingredient is so bad. It contains no rare ingredients and the

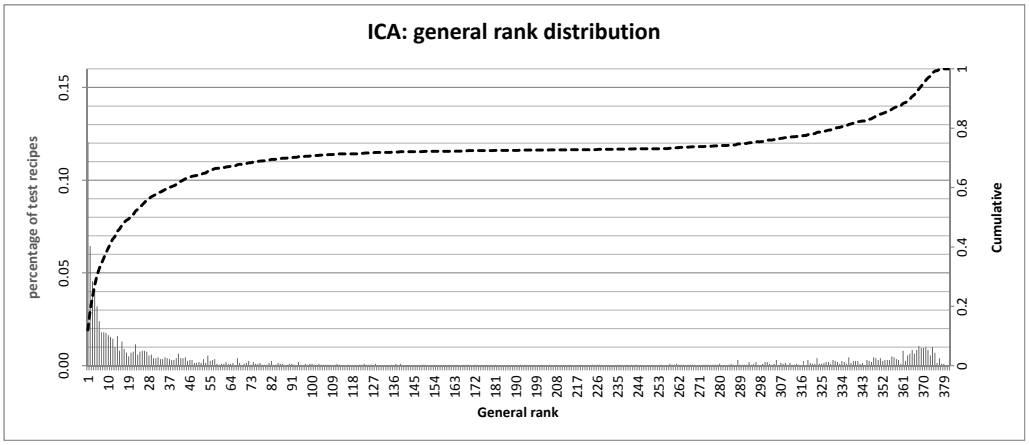
number of ingredients in these recipes is not particularly smaller or larger than recipes that have good predictions.
It can be concluded that the non-negative matrix factorization is the best technique of matrix decomposition to use when predicting ingredient combinations. Independent component analysis also is a quite good technique, however, this technique shows some weaknesses in predicting ingredient combinations for certain recipes. It is clear that singular value decomposition is not at all a good technique to use for this application. Its predictions are better than random (average percentage general rank is smaller than 50%) but not at all good enough to be used to predict ingredient combinations.

6.5.3 Comparing Western, Eastern and Southern diets

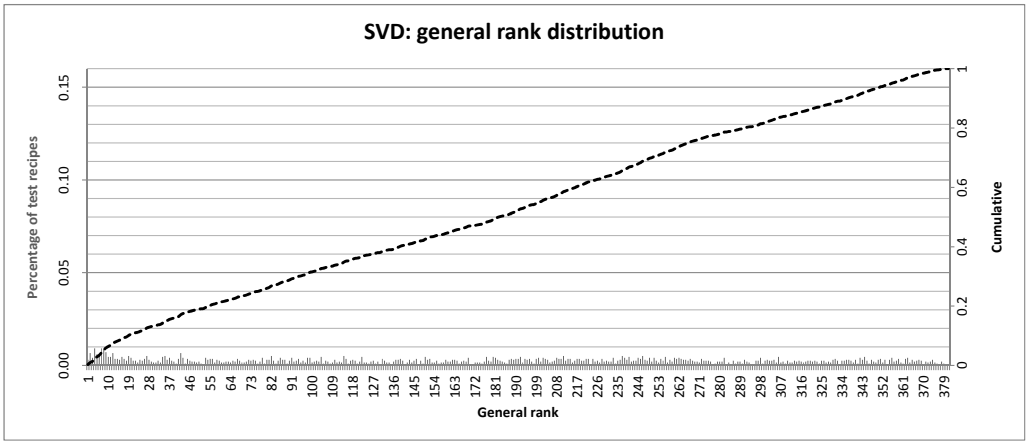
The unsupervised learning techniques showed that there was a difference in the way of combining ingredients in recipes between the different parts of the world. To see if this is also important when predicting eliminated ingredients, the data set is divided into Western, Eastern and Southern recipes just like in Section 5.2.1.
Based on the conclusion in Section 4.3.3, which states that the origin of the recipes that should be coming from North America is sometimes doubtful, the North American recipes are reclassified. This is done by dividing all the recipes into the 4 clusters that were found in Section 4.3.1 and training a random forest model to predict the origin cluster based on the ingredients found in each recipe. The train data contained the recipes of all the continents except for North America. 2,000 recipes of the 48,581 North American recipes were randomly chosen and also put into the train data. The remaining North American recipes formed the *test* data. This allowed to reclassify the North American recipes. The relabeling was done eleven times and each time the new labels for the North-American recipes were saved. Each North-American recipe was then classified into the cluster (West, East or South) that was predicted most for that recipe during these eleven relabellings. Resulting in a decrease of Western recipes and an increase of Eastern and Southern recipes.
It is important to note that the total number of ingredients per category is different for the three origins. This will have a influence on the category rank percentage. For each origin the number of unused ingredients is determined, as is their category, and these results are taken into consideration when calculating the rank percentages. The Eastern recipes contain only 301 different ingredients, the Southern 334 and the Western 363.
The same process of model development was used as before, but in all three cases the matrix decomposition was done by using the non-negative matrix factorization technique, since this technique scored best in the previous section. During training the value of K was varied from two to fifteen, just as in the previous section, and only twenty recipes were taken as train data at the same time. The models are just as in the previous section trained and tested for one hundred times resulting in again 2,000 test results (with 100 values of best K), which can be



(a) Non-negative matrix factorization (NNMF)



(b) Independent component analysis (ICA)



(c) Singular value decomposition (SVD)

Figure 6.3: General rank distribution after matrix decomposition

Table 6.2: Summary of the results for the different parts of the world, using NNMF (* 381 ingredients, ** 2000 test recipes).

	All categories			Within category	
technique	average % general rank*	st.dev. general rank	% recipes: ingredients in top 10**	average % category rank	average value of K
World	10.8%	17.5%	43.6%	15.8%	5.2
East	8.4%	14.0%	50.0%	12.4%	4.2
South	8.6%	15.2%	50.5%	12.9%	4.3
West	9.0%	15.9%	50.8%	14.5%	3.4

analyzed. Just as in the previous section, the average percentage general rank, the average percentage category rank and the percentages of recipes having a general rank smaller or equal to ten are determined. The results of the Western, Eastern and Southern recipes can be found in Tabel 6.2 together with the results found when using the total data set (world). There are no large differences between the best values of K of the different origins. The value is lowest for the Western recipes and highest when taking into account the whole data set, but the differences are rather small. The average percentage general rank is lowest for Eastern recipes, but not that different from the values of the Southern and Eastern recipes. However it is clear that the prediction capacity can be improved when taking into account the origin; as all three average percentage general rank values are smaller than the one found for the world. This again proves that the ingredient combinations are different for the different origins and it is easier to predict the eliminated ingredient when only taking into account recipes with the same origin and thus the same way of combining ingredients. The same conclusion can be found when looking at the percentage of recipes with a general rank smaller or equal to ten. These percentages are not to different between the different origins, but are clearly larger than the one found with the whole data set. Here the difference is even clearer: 50% instead of 43.6%, meaning that for half of the test recipes the eliminated ingredient can be found in the top ten of recommended ingredients to add to the remaining ingredients.

6.5.4 A closer look at the components

To get a better view at how the non-negative matrix factorization makes the decisions to predict certain ingredients and based on what it groups the ingredients together, the latent features of the ingredient matrix (Figure 6.1) are studied in more detail. As mentioned before, NNMF does not order the features from most important to least important, meaning that it is not possible to just take the first two features and study those. It is possible to tell the technique to calculate two latent features, and in that way force the technique to compress

all the information in two features. However, to give the technique some more freedom, the model is asked to calculate three features and the features are compared two by two. This method can be compared with the principal component analysis used in Chapter 5.
The three plots can be found in Appendix A. When plotting the first and second feature (Figure A.1), it is clear that the ingredients are divided based on the type of recipe: left in the graph there are *sweet* ingredients, mainly found in desserts and more on the right there are ingredients found in the main course. Ingredients for which the first feature is equal to zero, correspond to sweet dishes. Some examples are egg, vanilla, cinnamon, walnut, raisin, yeast, ect. This can be more clearly seen on the zoom of the plot (Figure A.2). The larger the value of the first feature, the more an ingredient fits in a main course instead of a dessert, for instance olive oil, tomato and garlic and onion with the largest value of the first feature. When looking at the plot of the first and third feature (Figure A.3), it seems like the third feature is created based on the origin of the recipes. For instance, ingredients with a large value in feature three are ingredients mostly found in Western recipes: butter, milk, wheat, cream, vanilla, while the ingredients with a very small value (almost equal to zero) are the ingredients mostly found in Eastern recipes: for instance tomato, garlic, olive oil, cumin, bell pepper. It seems like the model divides the recipes into groups: type of recipe, origin, etc. and groups together the ingredients within the groups of recipes.
As the model has a best value of K of approximately five, the non-negative matrix factorization is done again with five features this time. As it is not possible to plot combinations of these components against each other, the twenty ingredients with the highest values are determined for each feature. These ingredients are listed in Table A.1. The first feature contains ingredients from Southern recipes. The second feature contains ingredients commonly found in recipes from Eastern regions. The ingredients in the third feature are ingredients that are mostly used in main courses and not in desserts, while in the fifth feature, it is the other way around. The same conclusion can be made as before. The model selects ingredients based on type of recipe (main course or dessert) and origin of the recipes.

6.6 General conclusion

It can be concluded that, of the three techniques that have been studied, non-negative matrix factorization is the best technique to predict ingredient combinations in recipes. This can partially be explained by the fact that the data contains no negative values, the ingredient is either present or not. For the second part of the explanation, it is important to look at the results of unsupervised learning techniques: these show that ingredients can be clustered based on combinations of ingredients found in recipes. This means that the latent features created by the matrix decomposition method represent groups of ingredients, where each feature has high values for certain ingredients and low values for others, but all positive. Multiplying the low rank matrices will result in a weighted positive combination of existing

ingredient combinations, and this will result in a *new* recipe. This leads to the conclusion that non-negative matrix factorization is appropriate to predict ingredient combinations. Its performance is acceptable as the average percentage general rank is equal to 10.8%. This means that in 50% of the recipes the eliminated ingredient can be found in the top 40 of best fitting ingredients with the remaining ingredients in the recipe.
A second conclusion is that there is no real difference in performance of the model between the different origins, however, the performance is better when only recipes of the same origin are considered, compared to when the whole data set is used to build the model. It is possible that the four models are totally different, for instance in the way of predicting ingredient combinations. So it might be a good idea to build three separate models: one for Western recipes, one for Eastern recipes and one for Southern recipes and let the user decide which origin he wants to chose for his *new* recipe.
When looking at the features of the non-negative matrix factorization model, it is clear that the model takes into account the type of recipe (dessert or main course) that can be formed with the given ingredients and the origin of recipes containing the given ingredients. The model classifies the recipes into groups and looks for the most logical combinations of ingredients within a certain group.

Chapter 7

Two-step recursive least squares model

In this chapter, a second type of predictive model is built. The difference with the model in the previous chapter is that this model will not only use the recipe data set, but will also take into account the information contained in the flavour data set. It will be studied whether or not this can lead to a better performance. For this model, the two-step recursive least squares method is used to predict the ingredient combinations. This allows to use not one, but two data sets to make predictions. Just as in the previous chapter, the aim of the model is to predict which ingredients can be added to a given set of ingredients to form a good dish.

7.1 Data

For this model, both data sets containing information on the ingredients are used. So not only the recipe data, but also the flavour data will be used. The recipes containing only one single ingredient are just as before removed from the data set, since these recipes do not give information on ingredient combinations, and are not useful for the model building. However, this time also recipes containing two ingredients are removed. This is because those recipes contain only one ingredient after elimination one for tuning or testing, and combining ingredients with only one remaining ingredient does not seem useful. This reduces the number of recipes in the data set from 56,498 to 55,001. The flavour data is reduced to only those ingredients that are also present in the recipes (1525 $\rightarrow$ 381 ingredients). When studying the flavour components present in these ingredients, it can be concluded that 86 of the 1,107 flavour components are not present in one of the recipe ingredients. These flavour components are removed from the data set as well, resulting in a total of 1021 flavour components.
The binary form of the recipe data X_u $(55,001 \times 381)$ is used to build the model, as is the binary data set of the flavour components X_v (381×1021). The recipes will be divided into train recipes, tune recipes and test recipes. For the standard version of the model, the data

sets will be brought into the model in the linear kernel form, as this was the best kernel method in Chapter 5:

$$K_u = X_u X_u^T, \text{with } X_u \text{ the binary recipe data set,} \tag{7.1}$$

$$K_v = X_v X_v^T, \text{with } X_v \text{ the binary flavour data set.} \tag{7.2}$$

In later sections, K_u and K_v will be replaced with other kernel matrices to optimize the model. These models will be compared to the standard version using two linear kernels.

7.2 Model building step by step

To build this second model, the kernel version of two-step RLS is used. This type of model is totally different from the matrix decomposition used to build the first model. Before explaining how the model is built, it is important to know how the equation of the kernel two-step RLS looks like, to know how the technique works. In the previous section, we determined that K_u and K_v will be the linear kernels of the recipe data and the flavour data. The following two-step RLS equation shows how the recipes can be predicted based on these two kernels:

$$Y \approx K_u W K_v. \tag{7.3}$$

The Y-matrix in this equation contains *binary* representations of the predicted recipes, which means that the Y-matrix is equal to the X_u matrix. The recipe data thus has a double function: it contains the recipes that should be found by the model, but it also contains the information on ingredient combinations needed to predict recipes.
Normal RLS trains a set of independent linear models, here this would mean that for each ingredient a linear model is trained. Two-step RLS on the other hand performs two regressions (of a transposed matrix), making the independent linear models trained in the first step, dependent during the second step. In the first step, the two-step RLS method will complete the training set. In the second step a model is build for the target task. The biggest advantage of the two-step RLS approach is that the model trained with auxiliary data, can be re-used when new target tasks appear. More information on the two-step RLS method can be found in Pahikkala et al. (2014).
Now that the two-step RLS equation is known, the model building can start. In a first step, the recipe data is divided into train, tune and test data. This is done by dividing the recipes into several groups, some groups become train ($X_{u,train}$), other groups become tune ($X_{u,tune}$) or test ($X_{u,test}$) data. Which groups become train, tune or test data is determined through cross-validation. The model will be trained and tuned five times in a row. This is done with five groups of recipes, in the first step, the first group of recipes form the tune data, while the four other groups form the train data. In the second step, the second group of recipes becomes the tune data and the first, third, fourth and fifth groups become the train data and

so on.
Once the data is prepared, the model can be **trained**. During training the W-matrix in Eq. (7.3) is determined by giving Y, K_u and K_v; this W-matrix will be used during tuning. The model is trained with the part of the recipe data selected as mentioned above ($X_{u,train}$). The corresponding linear kernel is $K_{u,train} = X_{u,train}X_{u,train}^T$. This kernel is used to determine the W-matrix. To prevent overfitting, the matrix of coefficients is estimated as follows:

$$W = (K_u + \lambda_u I)^{-1} Y (K_v + \lambda_v I)^{-1}. \tag{7.4}$$

Different values of λ_u and λ_v, the hyperparameters, are selected for tuning. For each pair of λ-values, a W-matrix is trained. This will allow to determine the optimal values of λ_u and λ_v, during **tuning**, that result in the best predictions. The tuning step starts with predicting $Y_{tune,predict}$ using Eq. (7.3), with $K_{u,tune}$ equal to $X_{u,tune}X_{u,train}^T$. By using $X_{u,train}$, $K_{u,tune}$ has the same dimension as W, which is needed to multiply the matrices. But first one ingredient of each tune recipe is eliminated from the data by replacing the *one* in the data (in the row of the tune recipe and the column of the eliminated ingredient) with a *zero*, just as in the previous chapter. The names of the eliminated ingredients are saved and will be used to evaluate the tune results. Also the remaining ingredients are determined, since these ingredients are not of interest and their predicted values will be removed during the evaluation. As the model is trained and validated five times in a row for tuning (as mentioned above), five optimal λ_u and λ_v values are found. The method of determining the optimal values is explained in the next section. The median of these values is taken as optimal value for testing. All five values are saved as well.
Once the optimal values of λ_u and λ_v are found, the model is **tested**. The testing method is similar to the one used in the previous chapter. The main difference is that the number of selected test recipes is much larger than with the matrix decomposition model. This is because the test data is not present during training and tuning of the model. With the previous model, the test data could not be separated from the train or tune data as the whole data set matrix had to be used for decomposition. This also means that with this second model it will be much easier to predict new recipes, as the W-matrix will always be the same and does not need to be recalculated when a new recipe is added.
Just as with the tune data, one ingredient of each test recipe is removed by replacing the *one* in the data with a *zero*. These ingredients are saved in the result matrix. To determine the W-matrix for testing the model, the train and tune data are united into $X_{u,traintune}$ to maximize the information captured in the W-matrix: W is calculated using Eq. 7.4, with $K_{u,train,tune} = X_{u,train,tune}X_{u,train,tune}^T$ and the optimal values of λ_u and λ_v (found during tuning). This W-matrix is then used to determine $Y_{test,predict}$ using Eq. 7.3, with $K_{u,test} = X_{u,test}X_{u,train,tune}^T$. The prediction results will be used to evaluate the model.

7.3 Model evaluation

7.3.1 Training and tuning

The model is trained for different values of λ_u and λ_v. After each training the model is tuned to evaluate the prediction capacity of the trained model based on the ingredients that were left out of the tune data. This time it is not the rank of the eliminated ingredient that is used to evaluate the capacity of the model as in the previous chapter, but the area under the (Receiver Operating Characteristics) ROC curve, or AUC method. On a ROC curve, the number of true positives is plotted on the Y-axis and the number of false positives on the x-axis. Some examples of ROC curves are given in Figure 7.1. The AUC allows to not only make sure that the eliminated ingredient has a large value in the prediction, but also that the ingredients that are not present have a low value in the prediction. When the prediction is done well, the AUC-value should be near one. AUC-values go from zero to one, where zero means that the predicted labels are the opposite of the actual labels, AUC of 0.5 means random labeling and one is perfect. More information on AUC-values and ROC-curves can be found in Huang and Ling (2005).
First, the remaining ingredients present in the tune recipe are removed from $Y_{tune,predict}$. Then, the AUC-value of the tune recipe is determined: the eliminated ingredient forms the true positive result, all the other ingredients are negative results. A preprogrammed function is used to determine the AUC of each tune recipe based on the values of $Y_{tune,predict}$. Once an AUC-value is calculated for each tune recipe, the average of these values is determined and will be used to find the optimal values of λ_u and λ_v: for each pair of λ-values, a W-matrix is trained and $Y_{tune,predict}$ values are predicted for which the mean AUC-value is determined. The pair of λ-values with the highest average AUC (closest to one) is crowned the optimal pair of λ-values. Figure 7.2 shows for each pair of λ-values the average AUC-value. From the graph it is also clear that the value of λ_u is more important to optimize the performance of the predictive model than the value of λ_v.
This is done for each combination of train/tune data, resulting in five pairs of optimal λ-values. The median of these values is taken for each λ and these are the values that will be used during the testing part of the model building.

7.3.2 Testing

The optimal values of λ_u and λ_v are used to build the W-matrix as explained in Section 7.2. The $Y_{test,predict}$-matrix is evaluated similarly as done in Chapter 6 to make it easier to compare both models. This means that the rank of the eliminated ingredients of the test data is determined. Each recipe is evaluated separately.
First the predicted values of the remaining ingredients of each test recipe are replaced by very low values, making sure that these ingredients will not be suggested as best fitting.

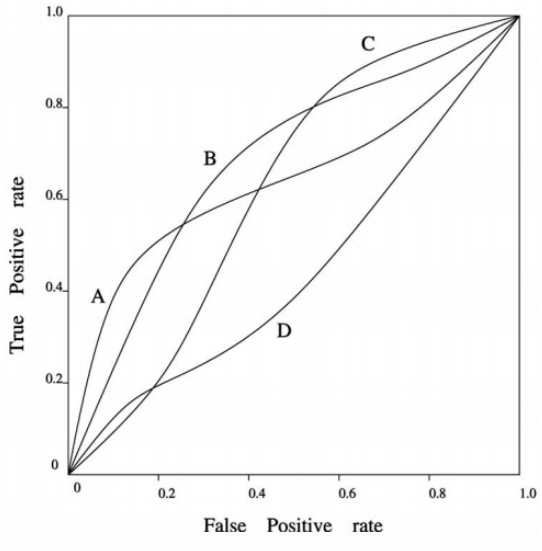


Figure 7.1: Four examples of ROC curves. A ROC curve dominates an other ROC curve, when for each value of false positives (FP) the value of true positives (TP) is higher for the first ROC curve than for the second. In this figure curve A and B dominated D. This means that the AUC-value of A and B will be higher than that of D. (source: Huang and Ling (2005))

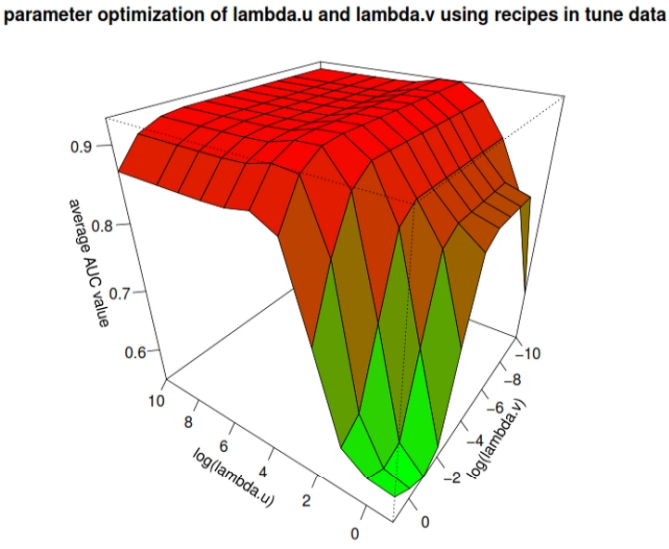


Figure 7.2: Average value of AUC of recipes in tune data for optimizing the value of λ_u and λ_v. Here the optimal value of $\log(\lambda_u)$ is three; the optimal value of $\log(\lambda_v)$ is minus two.

In a next step the predicted values of each recipe are ordered from highest value to lowest value. Ingredients that fit the remaining ingredients well, will have a high value, ingredients that do not make a good combination with the remaining ingredients will have a low value. The rank of the eliminated ingredient in this ordered list is determined and saved in the result matrix, next to the name of the eliminated ingredient. Also, the top five ingredients of best fitting ingredients is saved in the result matrix. Just as in the previous chapter the rank is also determined when only the ingredients of the same category as the eliminated ingredient are taken into account.The category and this rank are saved into the result matrix as well. Since it is possible to take a large number of test recipes (which was not possible in the previous chapter, since the test data could not be separated from the train and tune data), this whole process needs to be done only once to have the number of results that is needed to make reliable conclusions.

7.4 Results

7.4.1 Performance measures

The same performance measures are used to evaluate the predicting capacity of the two-step RLS model as were used to evaluate the matrix decomposition model. Just as a reminder the parameters will be repeated.
The first two parameters are the average percentage general rank (Eq. (6.5)) and the average percentage category rank (Eq. (6.6)), giving the average percentages of ingredients which have a smaller rank than the eliminated ingredient. Besides those parameters, the percentage of recipes having a certain general rank or category rank is determined for each general rank value and category rank value. These percentages can be plotted against the rank values and a cumulative curve can be added. But these percentages also allow to determine the percentage of recipes having a general rank smaller or equal to ten or a category rank equal to one or smaller or equal to three. All these parameters allow to evaluate the performance of a model and to compare two models with each other.

7.4.2 The impact of scaling

In this section it will be investigated whether or not the eliminated ingredient can be predicted better if the data sets are standardized (= feature scaling) before determining the linear kernel or not. Or, in other words, it will be determined whether or not it is necessary to scale the data first before building (training, tuning and testing) the model.
When feature normalization is done, the values of each feature in the data have zero-mean and unit-variance. This is done by subtracting the mean of the feature from each value of the feature and then dividing each new value by the standard deviation of the feature.
First the influence of scaling on the flavour data is examined. The model is built twice: once

Table 7.1: This table tells whether or not scaling the flavour data will improve the prediction capacity of the two-step RLS model. Both versions of the model are built with $K_u = X_u X_u^T$ and $K_v = X_v X_v^T$. However for the second version X_v was scaled before determining K_v (* 381 ingredients, ** 3667 test recipes).

	All categories		Within category	
technique	average % general rank*	% recipes: ingredient in top 10**	average % category rank	% recipes: ingredient in top 3**
non-scaled X_v	6.7%	57.5%	12.3%	72.8%
scaled X_v	6.6%	57.5%	12.2%	72.7%

with scaled flavour data (X_v) and once without scaling the flavour data. In both models the recipe data is not scaled, this is done to just study the effect of scaling the flavour data. Both models are built with exactly the same train, tune, test data. This to be able to compare the results of both models. Comparing the performance of both models will be done based on the average percentage general rank, the average percentage category rank, the percentage of recipes with a general rank smaller or equal to ten and the percentage of recipes having a category rank smaller than or equal to three. These results can be found in Table 7.1.
There are no real differences in value of these parameters between the two models. The values of the model with the scaled flavour data are only slightly better, but the differences are actually not worth mentioning. It can be concluded that it is not necessary to scale the flavour data when building the model, since it does not (really) improve the results.
The recipe data should be scaled before dividing it into train, tune and test groups, because it is possible that a certain ingredient might not be present in the train, tune or test data, making it impossible to scale the data as the standard deviation becomes zero and it is not possible to divide by zero.

7.4.3 Do the flavour data improve the predictions or not?

To study whether or not it is better to add the information on ingredient combinations captured in the flavour data to the model when predicting the eliminated ingredient, two models are built. The two models are built identically, except for the K_v-matrix. The first model contains the K_v-matrix as described in Eq. (7.2), with the non-scaled X_v-matrix. In the second model K_v is replaced with an identity matrix I (381×381). Just as in the previous section, the two models are compared based on the average percentage general rank, the average percentage category rank, the fraction of recipes with a general rank smaller or equal to 10 and the fraction of recipes having a category rank smaller or equal to three. The two models are trained, tuned and tested with exactly the same data, making sure that the

Table 7.2: This table contains the results of two versions of the two-step RLS model. Each version of the model is built with $K_u = X_u X_u^T$, but K_v is different for the two versions as given in the table (* 381 ingredients, ** 3667 test recipes).

	All categories		Within category	
technique	average % general rank*	% recipes: ingredient in top 10**	average % category rank	% recipes: ingredient in top 3**
$K_v = X_v X_v^T$	6.7%	57.5%	12.3%	72.8%
$K_v = I$	5.8%	60.2%	11.2%	75.6%

results can be compared without any doubt. The parameter values for both models can be found in Table 7.2.

The model, containing an identity matrix where the K_v-matrix should be, scores for all four parameters best. It has a lower average percentage general rank and a higher percentage of recipes having a general rank smaller or equal than 10. This means that the eliminated ingredient in a test recipe is predicted better with the model containing the identity matrix, than with the model containing K_v. The same can be seen when looking at the values of the category parameters. Apparently, it is not better to add the information of the flavour component to the model when predicting eliminated ingredients, as it lowers the prediction capacity of the model.

However, this does not mean that the information in the flavour data is not useful. It could be possible that the chance of finding new ingredient combinations, leading to tasteful recipes, increases when the flavour components are added to the model as well. Research has already proven that ingredients with similar flavour components go well together. So adding this information to the model could improve the creativity of the model in making good ingredient combinations. This is examined in Section 7.4.6.

7.4.4 The impact of secondary ingredient interactions

In this section the secondary interactions between ingredients in recipes are added to the model as well. This means that for each combination of ingredients a value is added to the recipe data as well, turning the recipe data into an (55001×145542) matrix. However, because of the large dimensions of the recipe data, when also adding information on ingredient combinations for each recipe, it becomes impossible to calculate the linear kernel ($K_u = X_u X_u^T$). To solve this problem, the polynomial kernel will be used instead of the linear kernel. This kernel allows to take into account interactions of higher degrees. The equation of the polynomial kernel is as follows:

$$k(x_i, x_j) = (\langle x_i, x_j \rangle + \text{offset})^{\text{degree}}. \tag{7.5}$$

Table 7.3: Evaluation whether or not secondary interactions between ingredients in recipes improves the prediction capacity of the two-step RLS model. Both version are built with $K_v = X_v X_v^T$, but with a different matrix for K_u as given in the table (* 381 ingredients, ** 3667 test recipes).

	All categories		Within category	
technique	average % general rank*	% recipes: ingredient in top 10**	average % category rank	% recipes: ingredient in top 3**
$K_x =$ linear kernel	6.7%	57.5%	12.3%	72.8%
$K_x =$ polynomial kernel	5.9%	58.9%	11.4%	74.7%

In this case the offset equals 1 and the degree equals 2. For $K_{u,train}$ both x_i and x_j are recipes from the train data, but for $K_{u,tune}$, x_i are tune recipes and x_j are train recipes. In this model K_v equals $X_v X_v^T$, just as in the standard version of the model.
This model will be compared with the model where X_u is determined using the linear kernel. To make sure the results of both models can be compared, the models are trained, tuned and tested with exactly the same data. The results can be found in Table 7.3. From the results it is clear that the eliminated ingredient can be better predicted when secondary ingredient interactions are taken into account as well. The average general rank is lower, as is the average category rank and for more recipes the eliminated ingredient can be found in the top 10 of best fitting ingredients in general and top 3 of best fitting ingredients per category. It can be concluded that using the polynomial kernel instead of the linear kernel does improve the performance of the model.
Since the performance of the model improved when K_v was replaced with an identity matrix, and it improved when K_u was built using the polynomial kernel instead of the linear kernel, a new model is built having both changes. This model is also trained, tuned and tested with the same recipes as the other models and the results are compared with the standard version of the model. All the results are brought together in Table 7.4.
Looking at the results, it can be concluded that the performance of the model is even better when both changes are done in the model, instead of only one of the changes.

7.4.5 Two-step RLS vs. Matrix decomposition

In this section the best model found in the previous chapter is compared with the model built in this chapter. The matrix decomposition technique which delivered the best results in the previous chapter was non-negative matrix factorization. The two-step RLS model is built with both data sets and without scaling the data. However it is important to keep in mind that the tuning step is different for both models.
The two techniques are compared based on the same four parameters that are used in the

Table 7.4: Summary of the results of four versions of the model. The first version is the standard version, where both K_u and K_v are calculated using the linear kernel function. For the second version K_v stays the same, but here K_u is calculated using the polynomial kernel function. For the third function K_u stays the same, but here K_v is replaced with an identity matrix. In the fourth version both changes are done. (* 381 ingredients, ** 3667 test recipes)

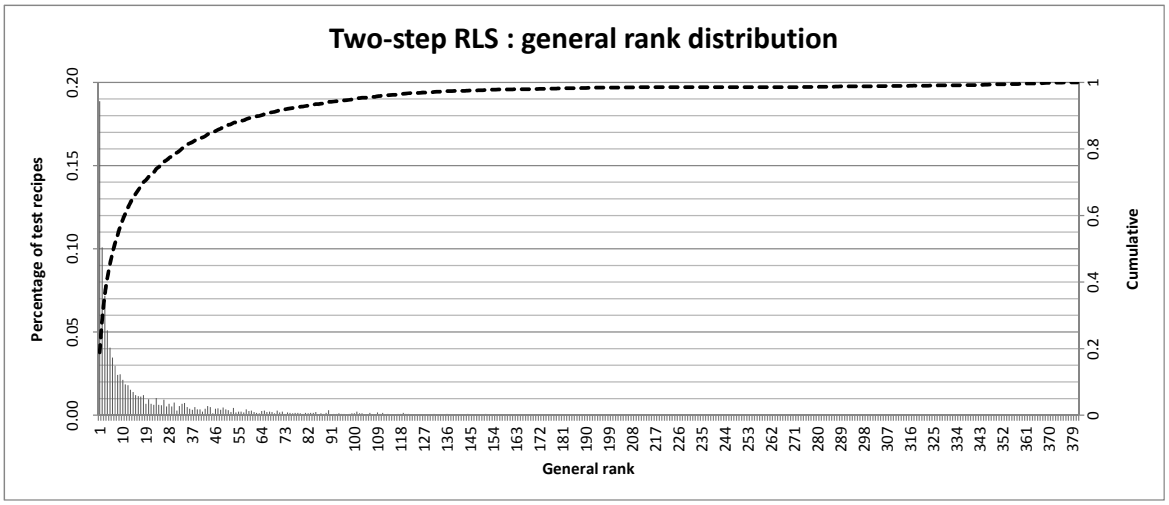
	All categories		Within category	
technique	average % general rank*	% recipes: ingredient in top 10**	average % category rank	% recipes: ingredient in top 3**
K_x =lin. and K_v=lin.	6.7%	57.5%	12.3%	72.8%
K_x =pol. and K_v=lin.	5.9%	58.9%	11.4%	74.7%
K_x =lin. and K_v=I	5.8%	60.2%	11.2%	75.6%
K_x =pol. and K_v=I	5.0%	61.3%	10.3%	94.9%

previous two sections: the average percentage general rank, the average percentage category rank, the percentage of recipes with a general rank smaller or equal to ten and the percentage of recipes having a category rank smaller or equal to three. The values of these parameters for both models can be found in Table 7.5. However, in this section also the plots of the recipe percentages against the general rank values will be compared. These graphs are shown in Figure 7.3.

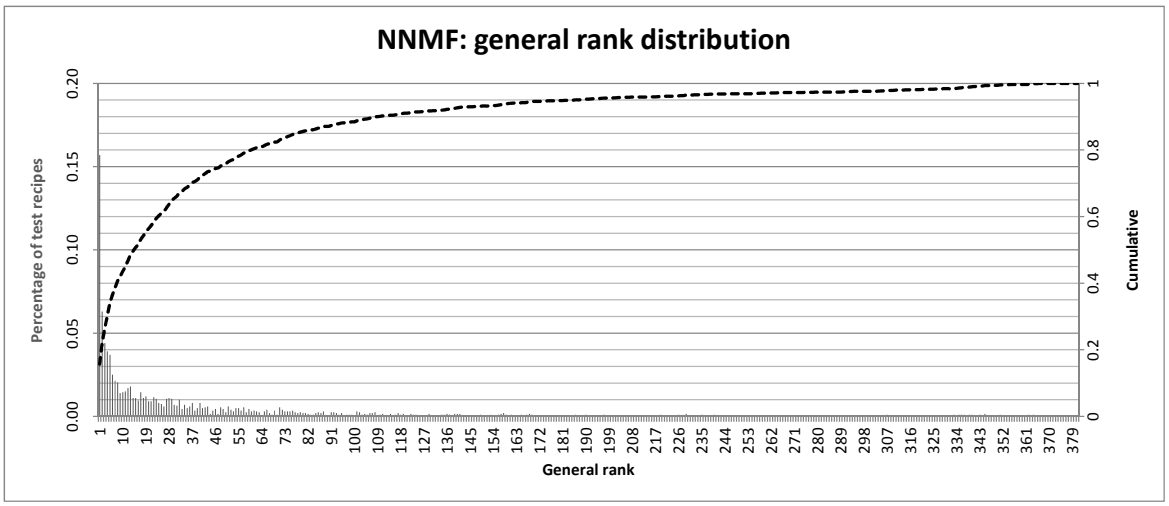
Looking at the values in Table 7.5, it is clear that the performance of the two-step RLS model is still a lot better than that of the NNMF model, despite the fact that this was the best model in the previous chapter. As expected, the two-step RLS technique can predict ingredient combinations better than a matrix decomposition technique. The same conclusion can be made when comparing the graphs in Figure 7.3. The cumulative curve of the two-step RLS model goes faster to one than that of the NNMF, meaning that more recipes have a smaller general rank. There are also more recipes with a general rank equal to one in the two-step RLS graph than in the NNMF graph. It is clear that the performance of the two-step RLS model is higher when predicting ingredient combinations in recipes.

7.4.6 The different versions of the model in practice

Testing which model can best bring back an eliminated ingredient is one way to test the performance of the different versions of this two-step recursive least squares model. However, that is not the aim of the model. The model is built to predict ingredient combinations for a given set of remaining ingredients in a refrigerator. So in this section the different versions are evaluated based on their top five of best fitting ingredients for a given set of ingredients.



(a) Two-step recursive least squares (Two-step RLS).



(b) Non-negative matrix factorization (NNMF).

Figure 7.3: Comparing the prediction capacity of the NNMF-model and the two-step RLS model.

Table 7.5: Comparing the prediction capacity of the NNMF-model and the two-step RLS model, based on the different performance measures. For the NNMF-model the results are determined using 2000 test recipes and for the two-step RLS model using 3667 test recipes.

	All categories		Within category	
technique	average % general rank	% recipes: ingredient in top 10	average % category rank	% recipes: ingredient in top 3
Two-step RLS	6.7%	57.5%	12.3%	72.8%
NNMF	10.8%	43.6%	15.8%	62.7%

To the four versions of the model, given in Section 7.4.4, a same set of ingredients is given and the four lists of best fitting ingredients are compared.
Firstly, the models are given ingredients that form a dessert, the ingredients given to the model are egg, cocoa and cream. All three ingredients are used in for instance chocolate mousse. The four lists of top five best fitting ingredients can be found in Table 7.6(a). It is clear that the last two version, both built with $K_u = (\langle x_{u,i} x_{u,j} \rangle + 1)^2$, are good in predicting an eliminated ingredient, but are not that great in suggesting ingredients: onion and garlic are not really fitting the dish. However the first two version both give an acceptable list of ingredients. All ingredients in this list can be used in a dessert, where vegetable oil is the less common ingredient found in desserts.
In a second round the models are given three new ingredients. This time they have to give the five ingredients that best fit chicken, rice and cream. These are ingredients found in a main dish. The results of the four models can be found in Table 7.6(b). Versions three and four give almost the same five ingredients as in the previous case, all ingredients that are frequently used in recipes. Versions one and two have changed their predicted ingredients. The ingredients fit the given ingredients rather well. Both models predict chicken broth and onion, which are commonly combined with chicken and rice. The second version, containing only information on ingredient combinations in existing recipes, also predicts mushrooms. The first version, containing also information on the flavour components, predicts brown rice to be combined with the set of given ingredients. However, as there is already rice in the given ingredients, brown rice will not be chosen by the user. This ingredient is probably suggested becomes it has a high number of shared flavour components with rice.
A third and last round contains tomato, beef and wheat, based on spaghetti, where the pasta still needs to be made, starting with wheat. The question is, will the different versions of the two-step RLS model also make a spaghetti-like dish, or will the suggested ingredients be something totally different? The results are given in Table 7.6(c). Just as in the previous cases the first two versions of the model give better fitting ingredients than the last two versions,

which again give ingredients with the highest likelihood of appearing in a recipe. Versions one and two give similar ingredients, however, version one suggests raw beef, while beef was already given to the model. This is because it shares a lot of flavour components with beef. It can be concluded that versions three and four of the model, which also take into account secondary interactions between ingredients in recipes, are not the best versions of the model to predict creative ingredient combinations. The models predict wheat, butter, egg, onion, garlic, independent of the given set of ingredients. So these version will not be selected as final version of the model to be used by other people. Versions one and two predict most of the time similar ingredients, however version one also adds ingredients with similar flavour components. This is not always useful, as it predicted brown rice, while given rice and raw beef, while given beef. These are not combinations that will be made in practice. Therefore version two, built with $K_u = X_u X_u^T$ and $K_v = I$, is selected as best version of the model.

7.4.7 Adding an additional ingredient to the model

This section examines what happens to the list of ingredients when one ingredient is added to the set of given ingredients. In the previous section, the version of the model built with $K_u = X_u X_u^T$ and $K_v = I$, is selected as *best* version. Therefore only that version of the model will be used from now on.
The experiment starts by giving only one ingredient to the model. The selected ingredient is egg, this ingredient can be used in both desserts and main dishes. In a next step, the ingredient that could best be fitted with egg is selected as additional ingredient to the recipe and thus is given to the model as well. This step is repeated once more. The results of these three runs are given below:

- Egg: wheat, vegetable oil, vanilla, milk, bread
- Egg + wheat: butter, milk, vanilla, yeast, vegetable oil
- Egg + wheat + butter: vanilla, milk, yeast, lard, cinnamon

It is clear that the list of fitting ingredients changes when an ingredient is added to the set of given ingredients. The ingredients do not just move up to a higher rank and a new ingredient is not added at the bottom of the list. After each addition the model makes new predictions, which is rather logical as $X_{u,new}$ changes, and thus the model will calculate new predicted values for each ingredient.
In the next step the model is run twice: in one scenario an ingredient typical for dessert is added, in the other an ingredient commonly found in main dishes is added. The results can be found below:

- Egg + wheat + butter + cocoa: vanilla, milk, lard, cane molasses, yeast
- Egg + wheat + butter + onion: milk, vanilla, yeast, lard, vegetable oil

Table 7.6: Version 1 is the model built with $K_u = X_u X_u^T$ and $K_v = X_v X_v^T$, version 2 is built with $K_u = X_u X_u^T$ and $K_v = I$, version 3 is built with $K_u = (\langle x_{u,i} x_{u,j} \rangle + 1)^2$ and $K_v = X_v X_v^T$ and finally version 4 is built with $K_u = (\langle x_{u,i} x_{u,j} \rangle + 1)^2$ and $K_v = I$.

(a) Top five of best fitting ingredients, when given **egg, cocoa and cream**.

Version 1	Version 2	Version 3	Version 4
butter	vanilla	butter	butter
wheat	wheat	wheat	wheat
vanilla	milk	garlic	garlic
milk	butter	garlic	garlic
cane molasses	vegetable oil	milk	milk

(b) Top five of best fitting ingredients, when given **chicken, rice and cream**.

Version 1	Version 2	Version 3	Version 4
brown rice	chicken broth	butter	butter
onion	onion	onion	onion
chicken broth	butter	garlic	garlic
butter	mushroom	egg	egg
milk	milk	wheat	wheat

(c) Top five of best fitting ingredients, when given **tomato, beef and wheat**.

Version 1	Version 2	Version 3	Version 4
raw beef	onion	butter	butter
onion	egg	egg	egg
egg	yeast	garlic	garlic
yeast	garlic	onion	onion
garlic	butter	olive oil	olive oil

It would be best when the first list contains all *sweet* ingredients, and the second list contains all *main* ingredients. The first list does contain only ingredients found in desserts, however, the second list also contains a lot of *sweet* ingredients. Apparently egg, wheat and butter are mostly used together in desserts, and not that much in main dishes, which makes it difficult for the model to predict ingredients for main dishes An option here could be to ask ingredients from a certain category like meat or vegetable: when giving egg, wheat, butter and onion to the model and asking for meat or vegetables, the model returns:

- Meat: beef, chicken broth, bacon, chicken, ham
- Vegetable: potato, tomato, celery, mushroom, carrot

It can be concluded that adding the best fitting ingredient to the set of given ingredients, the suggested ingredients will not just move up a place, but new predictions will be done based on the presence of the new ingredient. It can also be concluded that the model will not always predict ingredients that are wanted, for instance the model returns ingredients for desserts instead of main dishes. But then there is always the option to request ingredients from a certain category.

7.4.8 Predict more rare or tasteful ingredients

In this section the predicted values that are given by the model are scaled before ordering them, resulting in the best fitting ingredients, to get rather rare ingredients or ingredients with a more distinctive flavour. Just as in the previous section only the second version of the model will be used.
To force the model into suggesting ingredients that are used less frequently, the predicted value of each ingredient is divided by the number of times that the ingredient is found in the recipe data. This is done as follows:

$$Y.pred\_scaled[i] = Y.pred[i] / \log(presence[i] + 1), \tag{7.6}$$

where i is an ingredient and *presence* is a vector containing for each ingredient the number of times it is present in a recipe from the recipe data. Since it is possible that an ingredient is only present once, each value is increased by one, this to make sure that the argument of the logarithmic function is not equal to zero.
To be able to scale the predicted values, to get more tasteful ingredients, a measure is needed that is correlated with the taste of the ingredients. In Section 5.3 the ingredients are grouped together based on their flavour components, which are good indicators for tastefulness of an ingredient. It seems that in Figure 5.4 ingredients with a lot of flavour like cheese, wine, tea are located at the edges of the figure and ingredients with less flavour are located in the middle of the figure, around the origin (0,0). Therefore the distance of each ingredient to the origin of the graph is determined and this distance will be used to measure the tastefulness of the ingredients. The predicted value of each ingredient is multiplied by the logarithm of the distance to the origin in Figure 5.4, increased by one:

$$Y.pred\_scaled[i] = Y.pred[i] * log(tastefulness[i] + 1) \tag{7.7}$$

The model is given chicken, rice and cream, just as in the second round of Section 7.4.6. First the top five is given without scaling the predicted values, then the predicted values are scaled for tastefulness and rareness, to get a top five with more rare ingredients and a top five with more tasteful ingredients. The three lists are given in Table 7.7.
When comparing the list with more rare ingredients, butter and milk are left out, which is logical as these are ingredients that are used in a lot of recipes. Gelatin and soy sauce are

Table 7.7: Top five of best fitting ingredients, when given **chicken, rice and cream** to the version of the two-step RLS model, with $K_u = X_u X_u^T$ and $K_v = I$. First when the predicted values are not scaled, second with predicted values scaled to predict more rare ingredients and last with predicted values scaled to predict more tasteful ingredients.

Non-scaled	Rare	Tasteful
chicken broth	chicken broth	cream cheese
onion	onion	cheese
butter	mushroom	strawberry
mushroom	gelatin	pepper
milk	soy sauce	cayenne

added, these are not that rare ingredients, but less common as butter and milk. So the aim to predict less common ingredients is fulfilled. The third list contains definitely ingredients that are more tasteful, for instance cheese, pepper, cayenne. These are also ingredients that form creative combinations with the given ingredients, for instance strawberry with chicken, rice and cream.
It can be concluded that scaling the predicted values to get more rare, or more tasteful ingredients is definitely possible, as long as there is a good measure to determine this characteristic of each ingredient.

7.4.9 A closer look at the *model matrix*

Just as in the previous two sections only the second version of the model will be examined. The model can be simplified as followed:

$$\begin{aligned} Y &= K_{u,newrecipe} W K_v, \\ &= X_{u,newrecipe} X_u^T W K_v, \\ &= X_{u,newrecipe} M, \end{aligned}$$

with $M = X_u^T W K_v$, which will be called the *model matrix* and X_u the binary recipe data set. The model matrix is a square matrix of order 381, meaning that both features and observations are corresponding with the 381 ingredients. Or, in other words, the matrix contains for each ingredient combination a value that says how well two ingredients go together.
This model matrix may contain information on how the ingredients are predicted, which parameters (origin, type, ingredients) have the most influence when determining ingredient combinations, etc. To collect this information the model matrix will be analyzed.
The model matrix is different for each version of the two-step RLS models. Only the model matrix of the version that was selected as best performing will be studied in more detail.

When looking at the values in the matrix, it is clear that the presence of some ingredients will not influence the result of the prediction, as these ingredients have a value of zero for each feature in the model matrix. This means that the model will ignore this ingredient and only look at the other given ingredients to suggest a top five. Some examples of such ingredients are angelica, beech, geranium, holy basil, etc. These are all ingredients that are very rare and thus not present in a lot of recipes. As the model matrix is built with only one third of the recipe data, because of memory issues, it is possible that these ingredients were not present in the part of the recipe data used to build the model matrix. The model matrix would have more information on these ingredients if the flavour data was taken into account as well, but as seen above, that version of the model is less useful in practice.
Some value are positive and others are negative. When a value is negative, this means that the presence of the given ingredient, found in the row of the matrix, prevents the ingredient found in the column of the matrix to be selected as best fitting ingredient. This means that these two ingredients do not make a good combination. When the value is positive, the two ingredients do make a good combination, and the higher the value, the better the combination. An example of a combination with a high value is apple and cinnamon or chicken and chicken broth.
As the model only takes into account information on ingredient combinations found in recipes, it is normal that an ingredient commonly found in recipes of Eastern origin, for instance soy sauce, has the highest values for other *Eastern* ingredients such as sesame oil, ginger, sake, scallion, garlic, etc. The same reasoning can be done for type of recipe. Ingredients commonly found in dessert group together. An example is vanilla, that has the highest values for cocoa, egg, milk, wheat, butter, cream, cane molasses.
It can be concluded that the model takes into account the type of recipe that can be made with the given ingredients and the origin of the ingredients. A second conclusion is that rare ingredients have no influence on the results of the model. Less rare, but still not that common, ingredients (e.g. black tea) have less say in the results, as these ingredients have lower values than typical ingredients (e.g. butter).

7.5 Website

Building a model that can predict ingredient combinations is one thing, however, if nobody can use the model, then why build a model in the first place? With this idea in mind a brainstorm began and the end conclusion was that making a website where people can use the model in practice was the best idea. The link to the website is *http://www.kermit.ugent.be/ingredient-suggester*.
The model on the website is the version built with $K_u = X_u X_u^T$ and $K_v = I$, as this was the version that suggested the most promising top five. The code behind the website is written in PHP. The model matrix is determined using R and the HPC-cluster. This matrix is loaded

into the memory of the website. In the memory there is also a list containing all the 381 ingredients that are found in the recipe data and a list containing for each ingredient the corresponding category. These are all needed to run the model and to build the website.
The user is allowed to insert five ingredients into (the memory of) the website, that are for instance left over in his/her refrigerator. This is done by typing the name of the ingredient they want to add and the website will give the possible ingredients through auto-complete, as the users can only choose between the 381 ingredients present in the list in the memory of the website. When the users have added five ingredients or less to the list of ingredients to give to the model, the user pushes the button on the screen and the model starts predicting the five ingredients that best fit the given ingredients. The process that happens behind the screen is as follows:

1. a vector with 381 zeros is created and is called *new_recipe*
2. the column number of the given ingredients are determined
3. at those places in the new_recipe vector the zero is replaced by one
4. the new_recipe vector is multiplied with the model matrix M
5. the values in the resulting vector corresponding with the given ingredients are replaced by -999, as these ingredients are not wanted
6. the values in the vector are ordered from high to low
7. the five ingredients that correlate with the five highest values are returned to the screen and can be seen by the user

As the *new_recipe* vector is binary, this process can be shortened. There is no need to create a new vector. The only things that need to happen are determining the column number of the selected ingredients and compute for each column of the model matrix the sum of the values that are located on the rows having those numbers. Which is the same as multiplying the two matrices, but quicker.
The user can also select from which category the suggested ingredients should come, if he/she wants to. Than the process is the same except that only the predicted values of ingredients from the selected category are taken into account.
Some screen shots of the website are shown in Appendix B.

7.6 General conclusion

A first conclusion that can be made is that the model based on two-step recursive least squares has even better results than the model based on non-negative matrix factorization. This means that it can better predict an eliminated ingredient of a recipe. For almost 60% of

the recipes, the eliminated ingredient can be found in the top ten of best fitting ingredients. But, more importantly, this two-step RLS method allows to predict fitting ingredients for a new recipe very easily, which was not an option with the matrix decomposition method.
Secondly, it can be concluded that it is not necessary to scale the flavour data before building the model, as the results of the model do not improve when the data is scale compared with the model where the data were not scaled.
The results of the model improve even more when the flavour data is left out of the model: the eliminated ingredients gets a higher place in the list of good fitting ingredients. And the same degree of improvement in predicting the eliminated ingredient can be found when secondary interactions between ingredients in recipes is added to the model as well. Combining these two changes to the model results in an even better performance of the model in predicting the eliminated ingredient.
However, when testing the different versions of the model in practice, it becomes clear that not all versions can be used to suggest ingredients to add to a given set of ingredients. When adding the secondary ingredient interaction to the model, the model predicts only ingredients that are commonly used in recipes, such as wheat, butter, egg, onion, garlic, ect. The list of best fitting ingredients does not really change when the set of given ingredients changes. This is not wanted, so even though the versions of the model containing also secondary interactions are good at predicting the eliminated ingredient, they are not suitable for suggesting ingredient combinations. Therefore, these versions of the model are eliminated.
The other two versions of the model both give different lists of suggestions for different sets of ingredients, a big improvement in comparison to the previous two versions. The two versions suggest some similar ingredients, but not all. The version containing also information on the flavour components of each ingredient suggests ingredients with similar flavour compositions as the given ingredients. This is good, as it is already proven in literature that ingredients with shared flavour components make good combinations. However, it is not always wanted, as the model suggested brown rice, when given rice and raw beef when given beef. These ingredients will not be used in combination. Because of this, the version of the two-step RLS model containing only primary interactions and no information on flavour components is selected as best version.
It is possible to force the model to predict more rare ingredients or more tasteful ingredients, as long as there is a good measure to collect these characteristics for each ingredient.
A last conclusion is that the more common the use is of an ingredient, the more influence this ingredient has on the results of the model. So rare ingredients have (almost) no influence on the suggested ingredients.

Chapter 8

General conclusions

The aim of this work was to build a model that predicts the ingredients that can best be combined with a given set of ingredients to create a good dish.
Canonical correlation analysis showed that the flavour components found in ingredients are correlated with the use of these ingredients in recipes. This means that some ingredients are combined based on their similarity in flavour. Therefore, it could be a good idea to add the information on ingredient combinations, captured in the flavour data, to the model.
During this work two types of models were built: a first model was based on matrix decomposition and for the second model we used the two-step recursive least squares technique. Multiple versions of both model types were built.
As mentioned above, the first model is based on matrix decomposition. By approximating a matrix by a product of two low-rank matrices, the decomposition technique has to search for patterns in the data. Each technique uses a different approach to split the matrix. Non-negative matrix factorization appeared to be the best technique to find patterns in the recipe data and is thus most suited to predict an eliminated ingredient of a recipe, as was done during tuning and testing of the model. Singular value decomposition has the lowest performance. This technique is apparently not capable of finding the right patterns in the data.
Supervised learning techniques show that the recipes could be divided into three groups, based on the origin of the recipes. Based on this information three models were built using only parts of the recipe data. There is no real difference in performance of the three models, however, the performances are better compared to when the whole data set is used to build the model.
Replacing the matrix decomposition model with the two-step RLS model allows to add the flavour data to the model as well. In this way, the model gets more information on ingredients that are rather rare in recipes, and thus the chance of these ingredients to be selected as best fitting ingredients will increase. This could result in more creative ingredient combinations, which was part of the aim of this work. However, adding the flavour components to the model made the model predict similar ingredients as the one that was given. For instance, when

rice was given to the model, it would suggest brown rice, and when given beef, it gives raw beef as output, as both of these ingredients have almost the same flavour composition. These are obviously not combinations that will be made in practice. This problem could be solved by tuning the model in such a way that ingredients that are too much alike receive a lower value.

If the model could recognize the type of recipe better, it would know that when egg is given in combination with cocoa, it has to suggest ingredients found in desserts. On the other hand, when egg is given in combination with onion, the model should no longer look for dessert ingredients, but it should suggest ingredients found in main courses. This change in the model would result in suggesting ingredients that are more suitable for the type of dish the user wants to make. The objective could be accomplished by adding the secondary ingredient interactions, found in recipes, to the model by using a polynomial kernel instead of a linear kernel. These interactions would tell the model to look for combinations in the dessert recipes when the secondary interaction egg and cocoa is present. However, this addition only led to suggesting ingredients that are used a lot. The reason for this is probably because these ingredients have secondary interactions with a lot of ingredients. Or in other words: a lot of ingredients have secondary interactions with these ingredients and thus these ingredients have a high chance of being selected. It is possible that this effect could be reduced by scaling the recipe data before calculating the kernel matrix or by choosing a lower degree in the polynomial kernel, for instance one point five instead of two.

Since part of the aim was that people could use the model, the two-step RLS model, built with a linear kernel of the recipe data, but without the flavour data, is incorporated into a website. Print screens can be found in Appendix B. This model is able to find an eliminated ingredient from a recipe, when all remaining ingredients of the recipe are given to the model. In 60% of the cases the eliminated ingredient can be found in the top ten of best fitting ingredients. The model is also able to suggest ingredients that truly fit the given ingredients. For instance, when the model is given chicken, rice and cream, the ingredients found in the top five of suggested ingredients are chicken broth, onion, butter, mushroom and milk. Further optimization of the model would lead to even better recommendations.

The model could still be improved by adding more recipes to the recipe data and bringing in new ingredients that can be selected by the users. As mentioned above, lowering the degree of the polynomial kernel and increased tuning when the flavour data is added, could result in better suggestions of the model. The final model could be validated through sensory testing.

Bibliography

Y.Y. Ahn, S.E. Ahnert, J.P. Bagrow, and A.L. Barabési. Flavor network and the principles of food pairing. *Scientific Reports*, 1(196), 2011.

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

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

N. Bakalar. Partners in flavour. *Nature*, 486:4–5, June 2012.

D. Barber. *Bayesian Reasoning and Machine Learning*, chapter 13: Machine Learning Concepts, pages 279–292. Cambridge University Press, 2012a.

D. Barber. *Bayesian Reasoning and Machine Learning*, chapter 15: Unsupervised Linear Dimension Reduction, pages 299–322. Cambridge University Press, 2012b.

C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

J. Chandrashekar, M.A. Hoon, N.J.P. Ryba, and C.S. Zuker. The receptors and cells for mammalian taste. *Nature*, 444:288–294, 2006.

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

B. De Meulenaer. Proteïnen. In *Levensmiddelenchemie*, 2011. Course at University of Ghent: faculty of Bioscience engineering.

J. Delwiche. The impact of perceptual interactions on perceived flavor. *Food Quality and Preference*, 15:137–146, 2004.

The Exploratorium. Your sense of smell. Source: The accidental scientist: Science of cooking: https://www.exploratorium.edu/cooking/seasoning/taste/smell.html. consulted: 12th of February 2014.

P. Forbes and M. Zhu. Content-boosted matrix factorization for recommender systems: Experiments with recipe recommendation. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 261–264, New York, NY, USA, 2011. ACM.

J. Freyne and S. Berkovsky. Intelligent food planning: Personalized recipe recommendation. In *Proceedings of the 15th International Conference on Intelligent User Interfaces*, IUI '10, pages 321–324, New York, NY, USA, 2010. ACM.

J. Friedman, T. Hastie, and R. Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, 2008.

J. Huang and C.X. Ling. Using auc and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Angeneering*, 17(3), March 2005.

A. Karatzoglou, A. Smola, and K. Hornik. kernlab - an s4 package for kernel methods in r. *Journal of Statistical Software*, 2004. longer version available on http://cran.r-project.org/web/packages/kernlab/vignettes/kernlab.pdf.

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

D.D. Lee and H.S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

G. Morrot, F. Brochet, and D. Dubourdieu. The color of odors. *Brain and language*, 79: 309–320, 2001.

M. Moyer. The food issue. *Scientific American*, 309(3), September 2013.

T. Pahikkala, M. Stock, A. Airola, T. Aittokallio, B. De Baets, and W. Waegeman. A two-step learning approach for solving full and almost full cold start probblems in dyadic prediction. *Workshop on the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, submitted*, 2014.

J.. Prescott. Psychological processes in flavour perception. In A.J. Taylor and D.D. Roberts, editors, *Flavor Perception*, chapter 9. Blackwell Publishing Ltd, 2004.

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

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

D.V. Smith and R.F. Margolskee. Making sense of taste. *Scientific American*, 284:85–92, 2006.

J. Sobecki, E. Babiak, and M. Slanina. Application of hybrid recommendation in web-based cooking assistant. In *Proceedings of the 10th International Conference on Knowledge-Based Intelligent Information and Engineering Systems - Volume Part III*, KES'06, pages 797–804, Berlin, Heidelberg, 2006. Springer-Verlag.

R.J. Stevenson. *The Psychology of Flavour*. Oxford University Press, 2009.

R.J. Stevenson and R.A. Boakes. Sweet and sour smells: learned synesthesia between the senses of taste and smell. In G.A. Calvert, C. Spence, and B.E. Stein, editors, *The Handbook of multisensory processes*, pages 69–83. Cambridge, MA : MIT Press, 2004.

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

Y. van Pinxteren, G. Geleijnse, and P. Kamsteeg. Deriving a recipe similarity measure for recommending healthful meals. In *Proceedings of the 16th International Conference on Intelligent User Interfaces*, IUI '11, pages 105–114, New York, NY, USA, 2011. ACM.

G. Varoquaux. Identifying relevant variables: PCA and ICA, Dimension reduction and interpretability. website: http://gael-varoquaux.info/scientific_computing/ica_pca/, March 2012. consulted: 1st of May 2014.

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

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

Appendices

Appendix A

Model 1: latent features

A.1 Five latent features (K=5)

Table A.1: Top twenty: ingredients with highest value for the five features.

Feature 1	Feature 2	Feature 3	feature 4	feature 5
olive oil	garlic	onion	egg	butter
garlic	cayenne	pepper	wheat	milk
tomato	vegetable oil	vinegar	vegetable oil	wheat
basil	scallion	tomato	cinnamon	cream
black pepper	black pepper	beef	vanilla	vanilla
parsley	soy sauce	celery	cane molasses	cocoa
macaroni	ginger	mustard	lard	cane molasses
parmesan cheese	cumin	carrot	walnut	cream cheese
oregano	rice	green bell pepper	yeast	yeast
bell pepper	cilantro	potato	nutmeg	pecan
thyme	bell pepper	tamarind	milk	starch
lemon juice	chicken	cane molasses	raisin	corn
rosemary	sesame oil	corn	buttermilk	milk fat
white wine	corn	vegetable oil	apple	gelatin
bread	fish	cheddar cheese	lemon juice	almond
cheese	coriander	mushroom	bread	coconut
mozzarella cheese	tomato	bread	almond	cheese
olive	lime juice	bacon	cocoa	cheddar cheese
chicken broth	shrimp	parsley	pecan	potato
mushroom	soybean	chicken	mustard	black pepper

A.2 Three latent features (K=3)

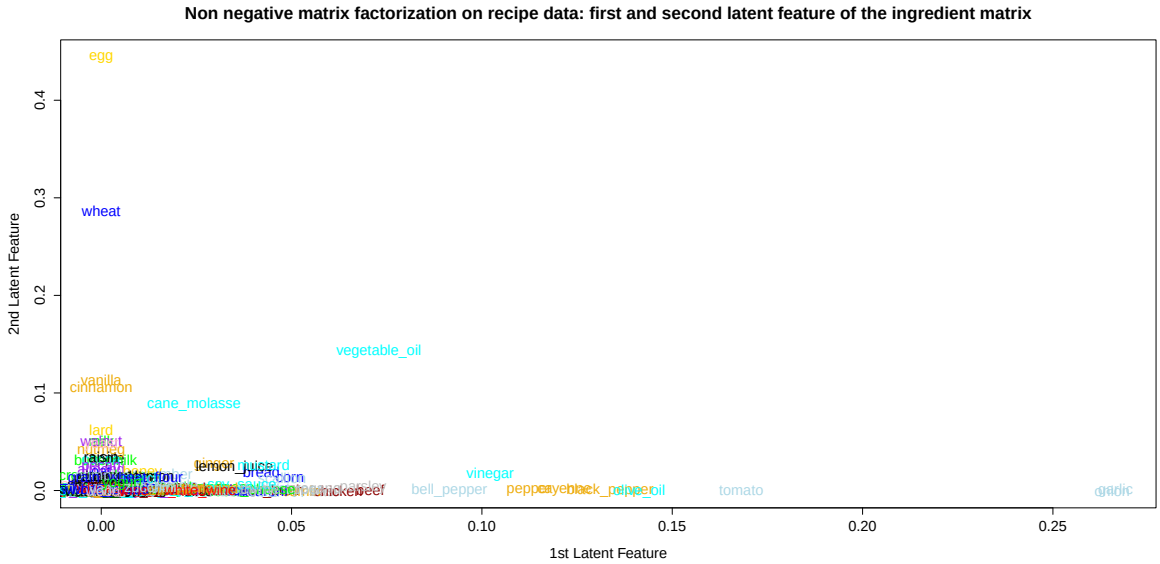


Figure A.1: Non-negative matrix factorization: first and second feature.

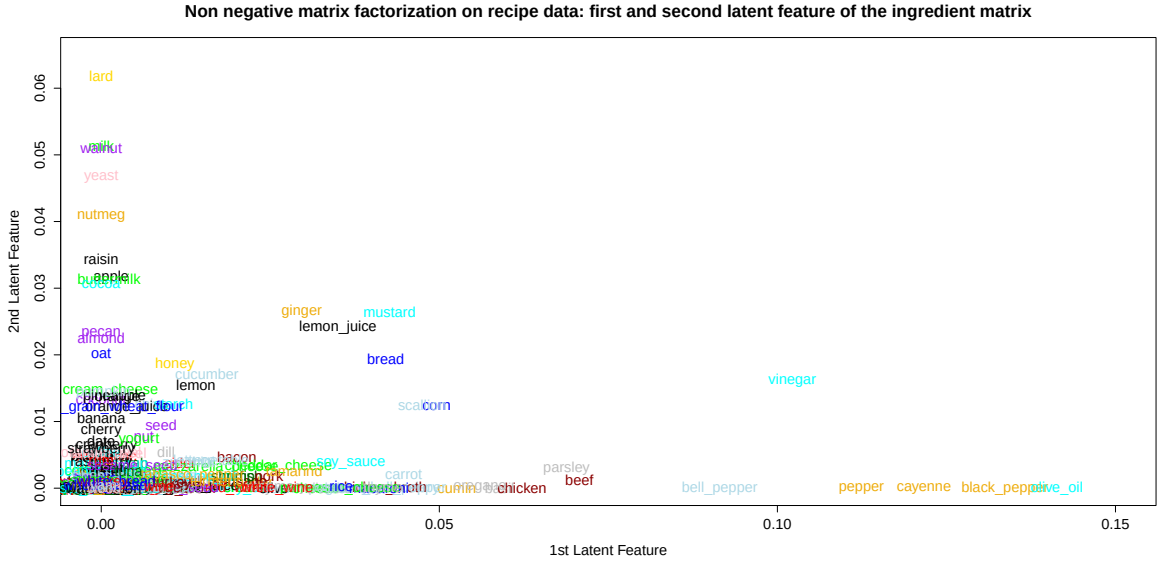


Figure A.2: Non-negative matrix factorization: first and second feature: zoom.

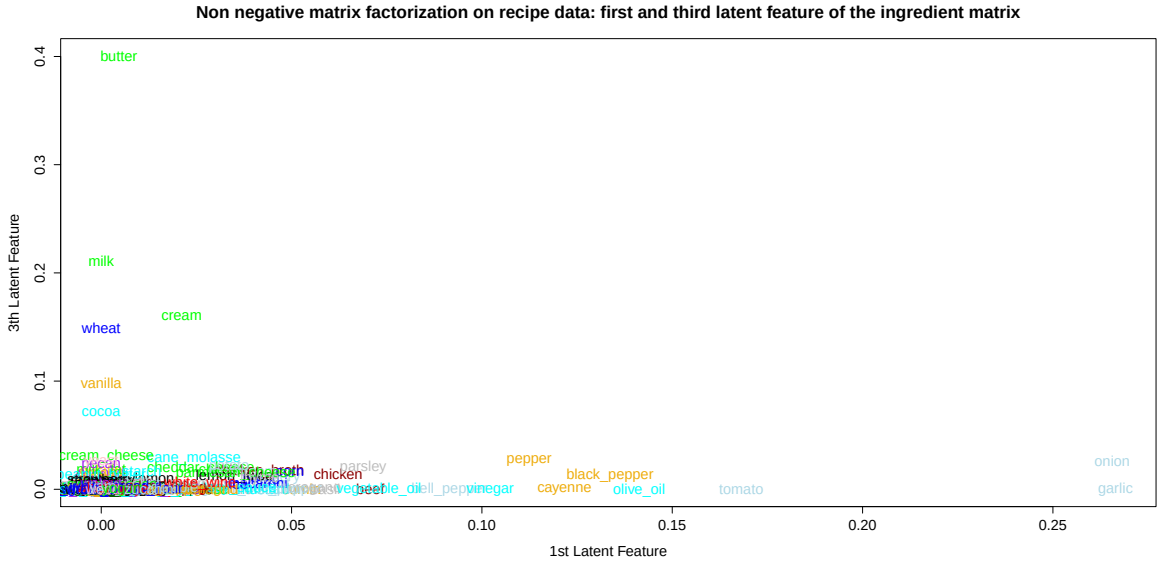


Figure A.3: Non-negative matrix factorization: first and third feature.

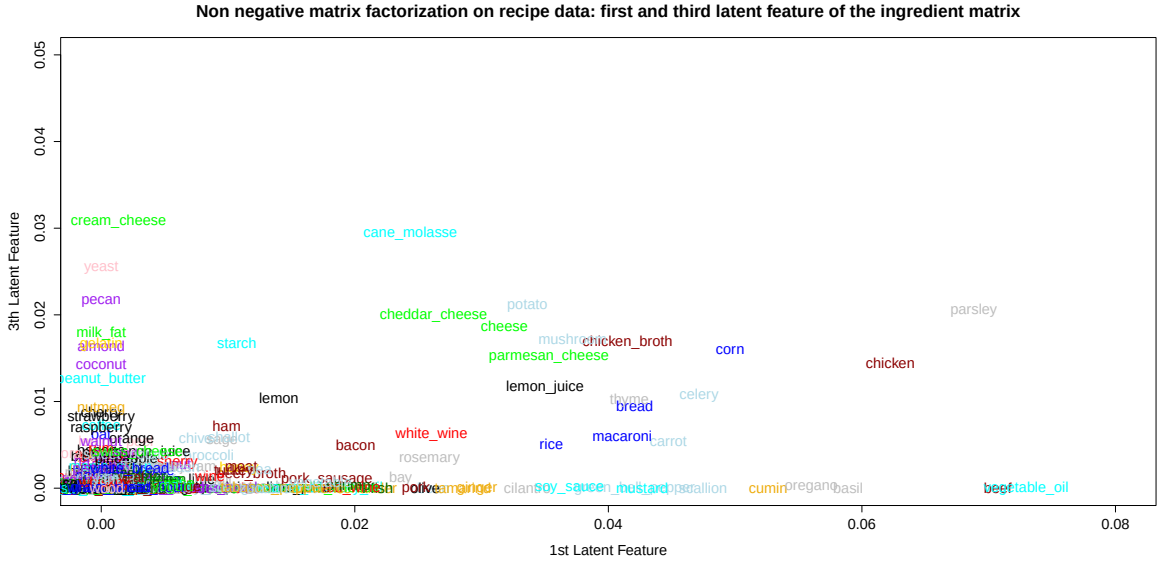


Figure A.4: Non-negative matrix factorization: first and third feature: zoom.

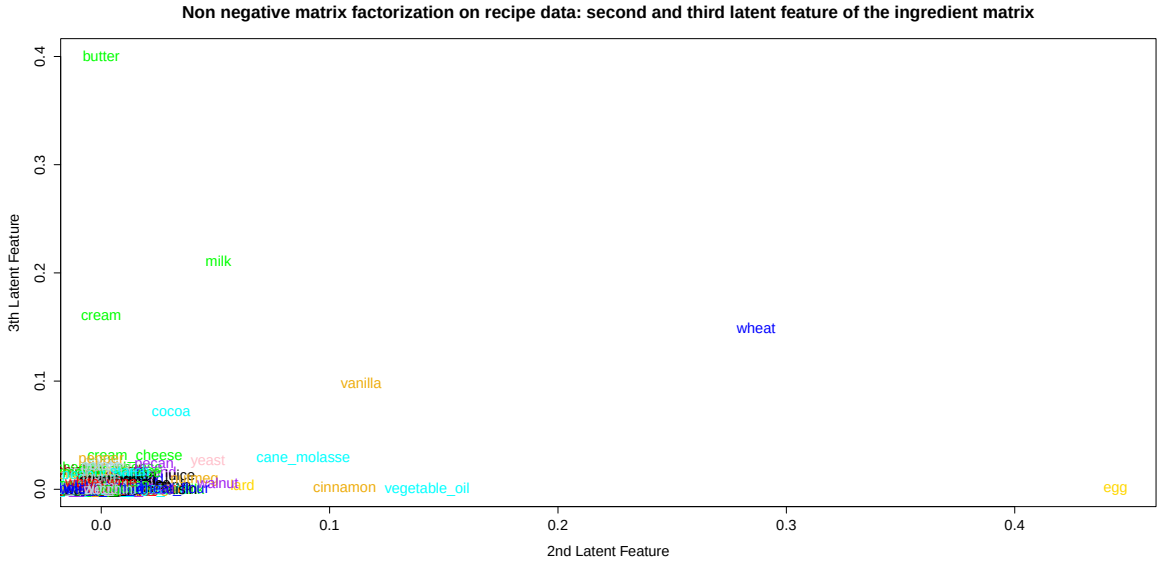


Figure A.5: Non-negative matrix factorization: second and third feature.

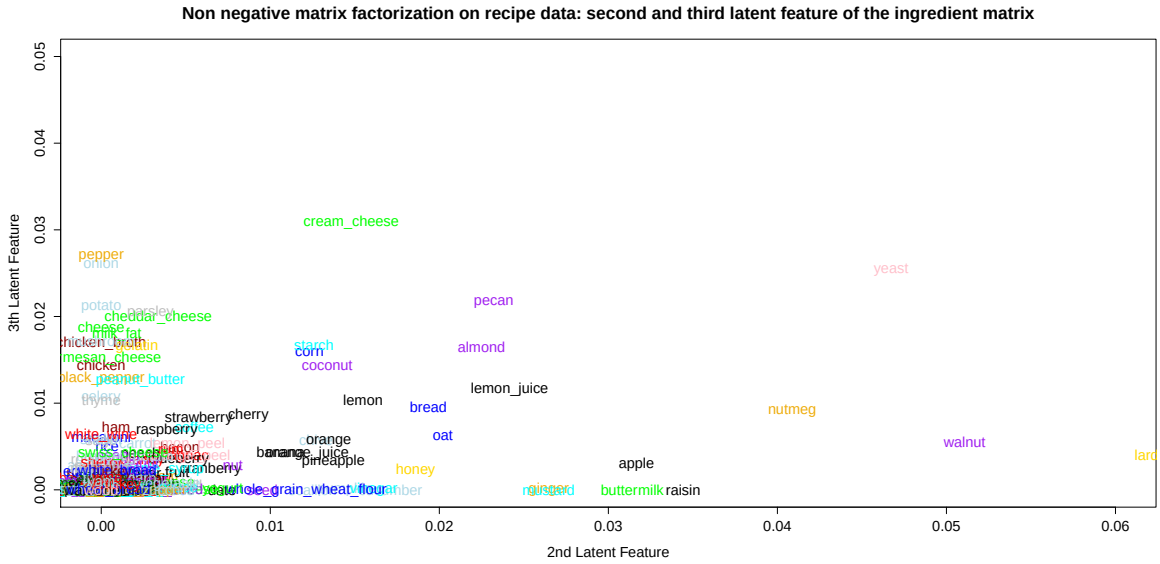


Figure A.6: Non-negative matrix factorization: second and third feature: zoom.

Appendix B

Model 2: Website

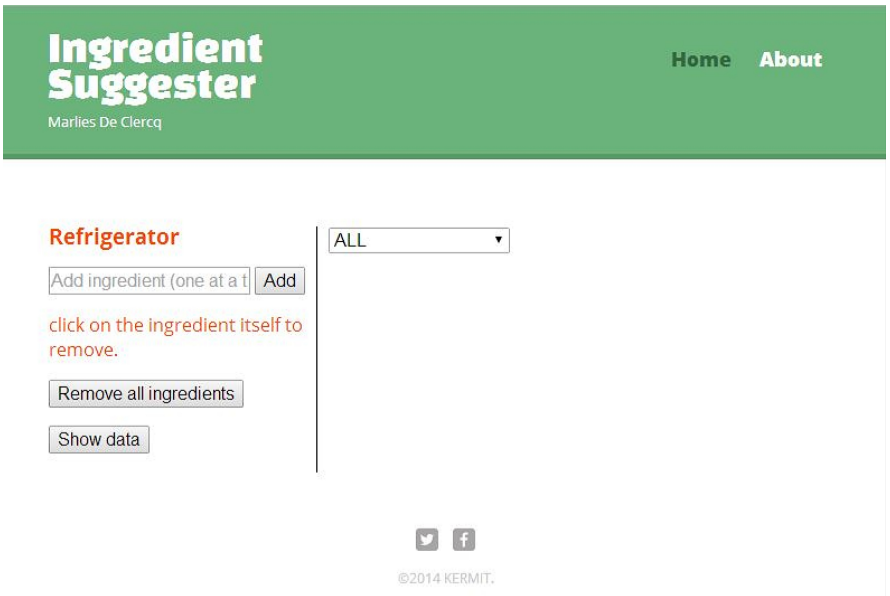


Figure B.1: When opening the website, it looks like this.

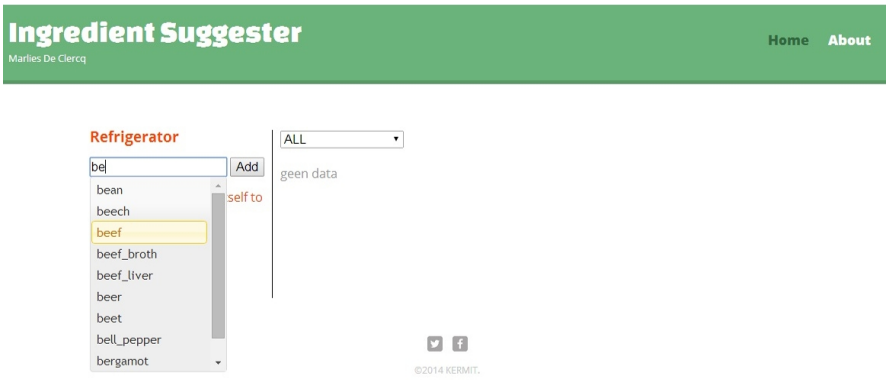


Figure B.2: The user can insert an ingredient by typing the name of the ingredient in the box, selecting the ingredient from the list and press the *add* button.

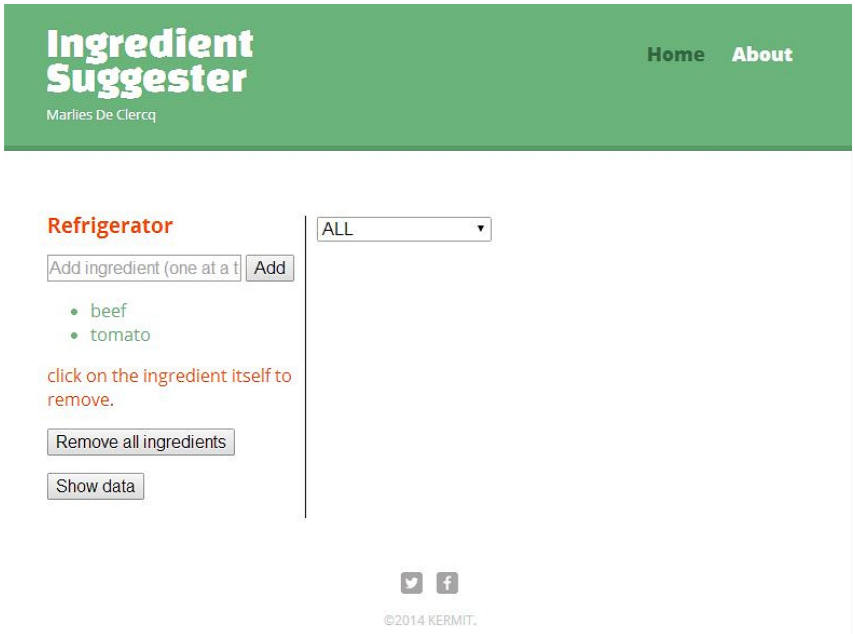


Figure B.3: When an ingredient is added to the list, the name of the ingredient is added in green beneath the box to add new ingredients. By clicking on the name, the ingredient can be removed from the list. When the user is satisfied with the list of ingredients he/she entered (max. 5), he/she can push the button *show data*.

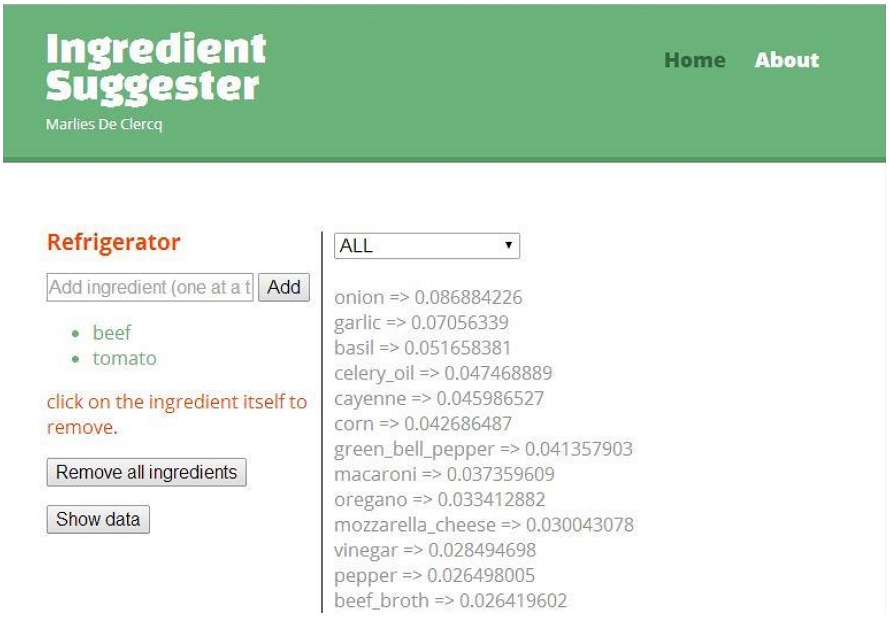


Figure B.4: The website returns a list of five names of ingredients that fit the given ingredients best. In this picture the model still returns more ingredients and the corresponding predicted values. This will be adjusted in the final website.

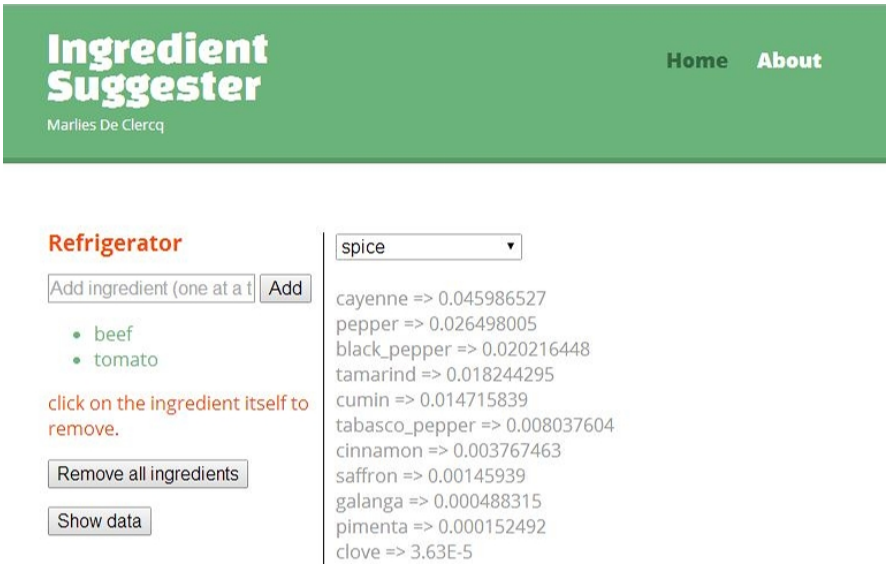


Figure B.5: The user can also choose to select a certain category the suggested ingredients should belong to. The website returns a list of five names of ingredients coming from the selected category that fit the given ingredients best.