Recipe reconstruction with genetic algorithms

Dr. R.J. Vlek, dr. D.J.M. Willems

CONFIDENTIAL

WAGENINGEN
UNIVERSITY & RESEARCH

Recipe reconstruction with genetic algorithms

Authors: Dr. R.J. Vlek, dr. D.J.M. Willems

This study was carried out by Wageningen Food & Biobased Research, funded and commissioned by National Institute for Public Health and the Environment. The work was conducted in an objective way by researchers who act impartial with respect to the client(s) and sponsor(s).

Wageningen Food & Biobased Research
Wageningen, January 2021

Confidential

Report 2123

WAGENINGEN
UNIVERSITY & RESEARCH

WFBR Project number: 6239185800
Version: Final
Reviewer: dr. L.A. van Rooijen
Approved by: dr.ir. H. Wensink
Funded by: National Institute for Public Health and the Environment
Commissioned by: National Institute for Public Health and the Environment
Confidentiality of the report: Confidential until January 2024

Contents

Definitions **4**

Summary **5**

1 Introduction **6**

1.1 Background 6
1.2 Use-cases 7
1.3 Data and knowledge 7
1.4 Prior art 8
1.5 Our approach 9

2 Methods **11**

2.1 Introduction in genetic algorithms 11
2.2 Genetic engine 12
2.2.1 Fitness function 12
2.2.2 Calculating nutritional content 13
2.2.3 Internal representation of recipes by absolute amount (codec 1) 14
2.2.4 Constraint and repair function 15
2.2.5 Internal representation of recipes by relative remainder (codec 2) 15
2.3 Performance evaluation 16
2.3.1 Evaluating the genetic algorithms 16
2.3.1.1 NEVO test set 16
2.3.1.2 LEDA test set 17
2.3.1.3 Evaluating reconstructions without ground truth 17
2.4 Data & knowledge organisation 18
2.4.1 Data Model 18
2.4.2 Data 19
2.5 Pipeline 22
2.5.1 Ingredient declaration parser 22
2.5.2 Nutritional data enricher 24
2.5.3 Criteria for reconstruction 25

3 Results **26**

3.1 NEVO test set 26
3.2 LEDA test set 29

4 Discussion & conclusion **31**

Literature **33**

Annex A: Additional result plots **35**

Annex B: Ingredient parsing in steps **37**

Annex C: Naive recipe estimation **38**

Definitions

EFSA – European Food Safety Authority
EuroFIR – European Food Information Resource
LEDA - De Nederlandse Levensmiddelendatabank (in English: Dutch Branded Food Database)
NEVO – Nederlands Voedingsstoffenbestand (in English: Dutch Food Composition Database)
RDF – Resource Description Framework
RIVM – Rijksinstituut voor Volksgezondheid en Milieu (in English: Dutch National Institute for Public Health and the Environment)
SPARQL - SPARQL Protocol and RDF Query Language

Summary

The Dutch National Institute for Public Health and the Environment (RIVM) monitors various aspects of food consumption and food safety. Product labels provide a lot of relevant information for this purpose, but fall short when it comes to the quantitative contribution of all individual ingredients to a product or a more detailed picture of the nutritional value of a product (only a small number of nutrients is usually declared). Wageningen Food & Biobased Research has investigated if (and which) techniques from artificial intelligence can be used to obtain an approximation (*reconstruction*) of the contribution of individual ingredients to a composite food, using ingredient data, as well as nutritional data from the food label, combined with expert knowledge about its individual ingredients.

This resulted in a genetic algorithm that is capable to find an approximation of the original recipe. Quality of the reconstructions varies with the complexity of the food and richness of its label data. A spread metric was implemented to provide additional insight in the convergence of the algorithm. The algorithm in its current form is best suited for reconstruction of foods that are not subjected to forms of processing that substantially impact the relation between nutritional content and total weight (e.g. heating or drying do this substantially). However, a method is proposed to cater for these forms of processing as well. A software system was developed around this algorithm to allow for large-scale automated application of the genetic algorithm on a database of branded food for the Dutch market. In this system, results of the reconstruction are also used to automatically impute nutrients missing on the food label. Background data and knowledge required to perform the reconstructions were organized in a machine-readable knowledge graph, facilitating automation and allowing for future re-use.

1 Introduction

In order to study various dimensions of food consumption, such as health benefits, food safety risks and environmental sustainability, the Dutch National Institute for Public Health and the Environment (RIVM) requires insight in the contribution of specific ingredients (e.g. tomatoes, olive oil, beef, vegetables) in composite foods. RIVM commissioned Wageningen Research to independently investigate if (and which) techniques from artificial intelligence can be used to obtain an approximation (*reconstruction*) of the contribution of individual ingredients to a composite food, using ingredient and nutritional data from the food label as well as expert knowledge on the food. The term ‘recipe’ is used throughout this document to refer to the contribution (type and amount) of each ingredient in a product, and does not cover any preparation instructions. The main advantage of a computational approach to recipe reconstruction lies with its speed and scale of application: it is potentially capable of reconstructing recipes for a database containing thousands of composite foods in relatively short time, compared to performing the same task manually. The project is closely related to, and builds upon another cooperation between RIVM and Wageningen Research named “Food Matching”. This report describes the scientific process and results of the project, and serves as a reference for the accompanying software system. The approach was developed for foods available on the Dutch market, hence all examples of data in this report will be in Dutch, though in principle it would be relatively simple to adapt the approach for use in other countries and languages.

Wageningen Research and RIVM collaborated to identify relevant data and knowledge for the recipe reconstruction algorithm, and to establish a methodology for evaluating the performance of the algorithm. The established methodology covers convergence of the algorithm under various conditions, as well as its performance on real-world recipes when available (sometimes only partially).

Substantial effort was also made to automatically transform unstructured information in a typical ingredient declaration into a more structured form, such that other relevant information could be more easily connected, and the resulting bundle of information becomes suitable for use with the recipe reconstruction algorithm. Making structured ingredient information and associated knowledge available may also be useful in many other scenarios beside recipe reconstruction.

This project has shown that an approximation of the original recipes could be successfully obtained using genetic algorithms. The accuracy of the reconstruction was shown to vary from food to food, depending on the complexity of the food and the availability and quality of data. Products with very little nutritional information and a large number of ingredients are more difficult to reconstruct accurately. The results of these efforts were combined into a software system capable of large-scale automated recipe reconstruction and imputation of missing nutritional values. While the current system is best suited for reconstruction of foods that have not been subjected to forms of processing that impact nutritional content or weight (e.g. heating or drying), a method is proposed to expand it to cover such foods as well.

1.1 Background

Commissioned by the Dutch Government, the Dutch National Institute for Public Health and the Environment (RIVM) monitors food consumption patterns of Dutch citizens. Consumption is typically reported along several dimensions, related to the intake of both ingredients (e.g. the amount of vegetables) and nutrients (e.g. the amount of zinc). Some of these dimensions can be easily derived from food consumption data, others require more complex data transformations. Products composed of multiple ingredients pose a challenge, as these often contribute partially to one or several dimensions. In order to determine to what degree eating a pizza contributes to the consumption of vegetables, one needs to know which of the pizza’s ingredients are vegetables and how much these contribute to the weight of the total product. Recipes contain this type of information, but are usually not publicly available for branded foods. Moreover, many nutrients in a product are not specified on its label, meaning that these will have to be derived as well. This can be done from nutritional knowledge on individual ingredients combined with a reconstruction of the original recipe. Wageningen Research has investigated if (and which) techniques from artificial intelligence can be applied to reconstruct an approximation of the original recipe, i.e. the individual amounts of the ingredients. To this end, data on nutritional values of ingredients were used from the Dutch Food Composition Database (NEVO).

1.2 Use-cases

During the preparation phase of this project RIVM indicated three use-cases for the results of the recipe reconstruction algorithm. These use-cases influenced the development of the recipe reconstruction algorithm.

1. Imputing missing nutritional values for a food: manufacturers rarely declare all nutritional values on the label. Only a small number is legally required, and a few are of interest for marketing purposes (e.g. to sell a product which is claimed as 'healthy'). The result of a recipe reconstruction algorithm could be used to impute the missing nutritional values. A 'best estimate' for the missing value is preferred. Knowing the lower and upper bounds of uncertainty also helps.
2. Total intake of specific ingredients is relevant in the context of food safety. In case of uncertainty, a worst-case scenario (upper bound) is preferred.
3. Intake of certain ingredient groups is also of interest: the amount of 'red meat', fruit or vegetables consumed. A 'best estimate' is again preferred.

1.3 Data and knowledge

In collaboration with RIVM, the following data and knowledge were identified as potentially relevant to solving the puzzle of recipe reconstruction.

Datasets

- LEDA (Dutch Branded Food Database): containing label information from over 100.000 food products on sale in Dutch supermarkets, such as global trade identification number (GTIN), name, description, declaration of ingredients (free text), and declaration of some[1] nutritional values (structured per nutrient).
- NEVO (Dutch Food Composition Database): containing structured and detailed information on nutritional values for generic foods (often encountered as ingredient as well), as well as recipes for composite generic foods.
- Weight yield factors (WYF) data available from another project at RIVM[2].
- Nutrient retention factors (NRF) data (Vásquez-Caicedo, 2014).

Knowledge

- Ingredients contributing more than 2% to a branded food are declared in decreasing order of weight on the label. Below 2% the order is allowed to be random. See Article 18 and Annex VII of EU regulation 1169/2011 (European Union, 2011).
- Ingredient order is determined by ingredient weight prior to the production process, but there are a few exceptions to be found in Annex VII of EU regulation 1169/2011 that complicate matters. Most notably, a manufacturer using "apple juice concentrate" is sometimes allowed to declare "apple juice" and establish the position in the declaration based on a calculated amount of apple needed to obtain the concentrate. The potential effects of the concentration process are obscured this way.

[1] The typical set of nutrients declared is very small compared to a data source like NEVO, and is usually motivated by regulation requirements and/or by marketing interests (e.g. promoting a healthy product by a detailed declaration of vitamins)
[2] Collected for data collection Dutch National Food Consumption Survey

- Similar products may be assumed to show at least some similarity in their recipes.
- The amount of information available on a food label varies a lot: some product labels are very informative (a lot of nutrients specified, and percentages a-priori known for some ingredients), whereas others are extremely information scarce.

Closer investigation of the problem of recipe reconstruction revealed several specific challenges.

Challenges

- Ingredient information of commercial foods is digitally available, but unstructured (free text). This means it is not straightforward to link individual ingredients to existing data on their nutritional composition. It also means that data quality issues (e.g. a spelling error) and/or alternative spellings and synonyms may be an obstacle in connecting nutritional data.
- Ingredient declarations often contain nested information (a specific ingredient being composed of sub-ingredients individually declared).
- While datasets on the effects of food processing exist, it is not easy to connect this information to a product and its individual ingredients and nutrients.
- The ingredient declaration is sometimes not very well defined, causing difficulty to unambiguously identify an ingredient, its exact processing state or the amount of a nested ingredient.
- Food product data is highly variable in quality and completeness.

1.4 Prior art

Various attempts at (semi-)automated recipe reconstruction have been made in the past, with varying types of data and knowledge involved. The most popular approach assumes that the nutritional values of a food and of its individual ingredients are known a-priori. Given this information, computational methods are used to find a weighted sum of ingredients, such that the nutritional content of the product is best matched. The work of Westrich et al. (1994 & 1998) follows this approach, and applies linear and quadratic programming techniques to find the optimal weight per ingredient. The approach described by Westrich et al. is not fully automatic, as food product data is still entered (and interpreted) manually for each individual product. The most notable aspect not covered by the computational method of Westrich et al. is the influence of food processing (if any) on the nutritional content of a product.

One of the reasons to perform recipe reconstruction is that the resulting recipe can be used in a subsequent step to estimate nutritional values not declared on the product label. With missing value imputation as the main goal, other (more direct) approaches have also been reported. Various (manual) strategies were proposed by Schakel et al (1997), some of which rely on borrowing information from similar products. The work of Kim & Boutin (2015) reported an approach somewhat similar to that used by Westrich et al., but focussed on estimating missing nutrients, rather than the recipe itself. An inspiring aspect of this work is the use of data and knowledge to narrow the range of likely values for individual nutrients, rather than aiming to obtain a single best value (with unknown spread around it).

Benefiting from access to large databases of images and associated consumer recipes, researchers associated to Facebook and the University of Barcelona took a different approach (Salvador et al. 2019), trying to reconstruct a recipe (including preparation instructions) with supervised machine learning techniques (deep neural networks). In similar fashion, Chokr & Elbassuoni (2017) applied deep learning techniques to predict calories from food images more directly. Although entertaining and useful for consumers, the relevance of an image-based approach to the context of RIVM remains doubtful, for two reasons:

1. The available databases used in abovementioned research typically cover consumer prepared food and recipes. No such databases exist for industrial food and recipes.
2. Nutritional information in visual data of products is inherently limited, mostly by object occlusion: the meat in bapao will never be seen, nor will the bapao's shape implicitly and

unambiguously predict the presence of meat inside it. Additional types of data are needed to disambiguate such cases.

In parallel to the attempts to automate recipe reconstruction with computer algorithms, the European Food Information Resource (EuroFIR) has taken the initiative to investigate various methods for estimating missing nutrients through manual recipe reconstruction (Reinivuo et all, 2009), hoping to harmonize one method for food composition databases within the European Union.

1.5 Our approach

The algorithm here developed follows the popular approach of recipe reconstruction as an optimization problem: seeking an optimum for the weighted sum of ingredients, based on nutritional values of both the product and its ingredients. The algorithm largely follows the harmonized recipe calculation method of EuroFIR (see section 1.4), when searching for the optimum recipe. In reality only a limited set of the food's nutritional values is available, which serves as input (see Figure 1). In contrast to Westrich et al. (1994), we rely on genetic algorithms - a form of artificial intelligence that mimics the process of evolution - to solve the optimization problem (Terfăloagă 2015, and Mafteiu-Scai et al., 2018). What is considered optimal is defined and quantified by a so called fitness function. Moreover, we aim to source all data required for recipe reconstruction automatically from databases (LEDA & NEVO). To achieve this, a dedicated ingredient declaration parser was used, and Semantic Web technology (Manola & Miller, 2004 and Brickley & Guha, 2014) was deployed to organize and query relevant data and knowledge.

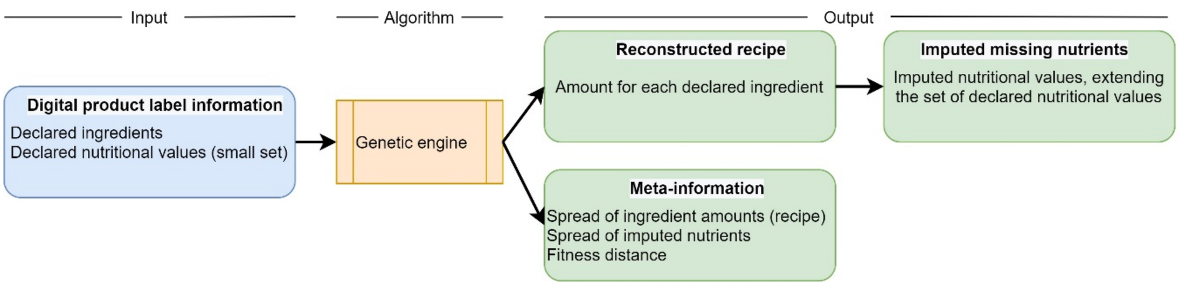

Input
Algorithm
Output
Digital product label information
Declared ingredients
Declared nutritional values (small set)
Genetic engine
Reconstructed recipe
Amount for each declared ingredient
Imputed missing nutrients
Imputed nutritional values, extending the set of declared nutritional values
Meta-information
Spread of ingredient amounts (recipe)
Spread of imputed nutrients
Fitness distance

Figure 1 Overview of the input to and output from the genetic algorithm

Inspired by the work of Kim & Boutin (2015), the algorithm not only aims to find an optimal recipe reconstruction, but also to quantify the spread of near-optimal solutions around it. Spread is quantified both in the domain of the recipe and the product's imputed nutrient content (i.e. those nutrients that were not already declared on the product label). This information facilitates the various use-cases described in section 1.2:

1. Missing data imputation can be achieved with additional insight in the degree of spread between near-optimal solutions. A large spread suggests there are many equally fit solutions, hence higher uncertainty about the imputed missing value.
2. For food safety applications, the worst-case intake of an ingredient can be estimated from the upper bound of the spread.
3. For investigation of the consumption of specific ingredient groups, spread can be ignored, and simply using the 'best' solution suffices.

Future extension of the approach was taken into account from an early stage in the development, already preparing for the use of the following additional knowledge to further refine the algorithm:

- Using data on weight yield factors and nutrient retention factors to take the effects of processing into account
- Using a nutritional value distance metric to similar products to constrain the solutions explored by the algorithm
- Using a recipe distance metric to a 'prototype' product, such that generic expert knowledge (e.g. on how a typical bread is made) can be incorporated as an additional constraint (e.g. avoiding recipes for bread with an unrealistic amount of fluid versus solid matter).

- Using multiple equally likely mappings between commercial food ingredients (from LEDA) and their generic counterpart (from NEVO), and disambiguating based on fitness of the solution found for each ‘hypothesized mapping’ via the algorithm (assuming that the mapping that results in the best fitting recipe is the correct one).

2 Methods

2.1 Introduction in genetic algorithms

Originating from early computational simulations of evolution, genetic algorithms gradually evolved into a more generic tool in the domain of artificial intelligence (see Turing (1950) for the original idea and Mitchell (1996) and Michalewicz (1996) for an overview of modern genetic algorithms). Simply put, genetic algorithms seek a parallel with the mechanisms of evolution to solve an optimization problem. An optimization problem can be characterized as the quest for the best solution from all possible solutions to a problem. What is considered "best" is typically determined by the problem domain, just like the rules and constraints on the "possible solutions".

So how does an evolutionary algorithm arrive at what it believes to be the best solution? It all starts with generating an initial *population* of possible solutions (see Figure 2) based on random amounts for each ingredient within logical constraint. The generation process of the initial population is often largely random, meaning that chances are small to already find a good solution among them. The quality of each individual solution (often called *individual*) is quantified by a *fitness function*. The fitness function is defined specifically for the problem, and results in a measure of fitness for each individual. Based on this information, a selection process takes place, whereby unfit individuals have a higher chance of being removed from the population than fit ones. The result is a population that maintains some of the diversity represented by unfit individuals, but consists largely of more fit individuals. This population can then procreate via a process of cross-over and mutation in order to generate a new population (next generation) of individuals that (hopefully) combine several of the factors that led to the success of their ancestors. The new population is again subjected to the fitness function, and the selection-procreation cycle repeats. A termination criterion determines when this cycle is broken. The simplest termination criterion is a fixed number of cycles: stop after *N* generations. Other criteria can also be used, such as when average fitness of the population no longer improves.

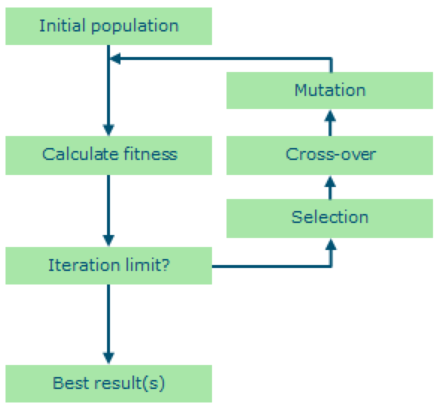


Figure 2 The process of evolution as paralleled by a genetic algorithm

As already discovered by Westrich et al (1994), the problem of recipe reconstruction can be framed as system of linear equations that can be solved with optimization: find the recipe that best matches the nutritional values declared on the product. While there are multiple ways to solve such a problem (see for instance Terfăloagă, 2015), genetic algorithms were chosen for the following reasons:

- Rapid convergence to near-optimal solutions
- Relatively robust against getting stuck in local optima
- Extremely flexible in the definition of constraints (e.g. see Section 2.2.4).

2.2 Genetic engine

The genetic algorithms used in this project were implemented in the Java programming language with the help of the Jenetics library, version 5.2.0 (Wilhelmstötter, 2018). Before being able to solve the problem of recipe reconstruction with a genetic algorithm, two decisions needed to be made.

1) How to calculate fitness?
2) How to represent potential solutions inside the algorithm (internal representation)?

Cross-over and selection mechanisms can sometimes also be developed specifically for a certain problem, but this was not done here as it was not considered a bottle neck in algorithm performance. Instead the Jenetics' default single-point cross-over method (followed by a mutator) was used, combined with the default tournament selector.

In order to allow for a direct comparison between performances of the genetic engine with different internal representations (codecs) of the problem a termination criterion of 10.000 iterations was set.

The following sections focus on answering the abovementioned questions. In order to illustrate the answers, consider the following fictive example of a cucumber salad (see Table 1). The blue cells indicate the typical information that can be found on the food label: the identity and order of the ingredients, and the nutritional value of the total product. The orange fields represent information on the nutritional content of individual ingredients, typically obtained from another source (in this case NEVO), that can be linked to the identity of the ingredients in this product. The green fields represent the recipe: the amounts per ingredient to be reconstructed.

Table 1 Data from a fictive cucumber salad

		Nutrient -- grams (unless specified otherwise) per 100g			
Ingredient	**Amount**	**Carbohydrates**	**Protein**	**Fat**	**Energy (kJ)**
Cucumber	? %	1,3	0,7	0,4	53
Tomato	? %	2,9	0,7	0,4	85
Lettuce	? %	0,3	1,4	0,4	52
Total	100%	1,68	0,77	0,4	62,5

2.2.1 Fitness function

No matter which internal representation is chosen, every potential solution will ultimately be converted to a recipe: an amount for each ingredient in the food. This is shown in Table 2, where the purple column of amounts represents a potential solution to the recipe. As we wish to optimize our recipe for a best match to the original product's nutritional content, we need a way to calculate the nutritional content from the recipe. Section 2.2.2 covers in detail how this was done. The result of such a calculation is visualized in the purple bottom row of Table 2. In order to determine the fitness of a solution, a fitness function was implemented based on the Euclidean distance (Anton, 1994) between the nutritional content as declared on the label (blue bottom row Table 1) and the nutritional content as calculated from the recipe (purple bottom row Table 2). The Euclidean distance was calculated on normalized nutritional values (values between 0 and 1), as described in Section 2.2.3.

Table 2 A potential recipe solution for the fictive cucumber salad

		Nutrient -- grams (unless specified otherwise) per 100g			
Ingredient	**Amount**	**Carbohydrates**	**Protein**	**Fat**	**Energy (kJ)**
Cucumber	50 %	1,3	0,7	0,4	53
Tomato	40 %	2,9	0,7	0,4	85
Lettuce	10 %	0,3	1,4	0,4	52
Calculated	100%	1,84	0,77	0,4	65,7

2.2.2 Calculating nutritional content

Calculating the nutritional content of a food from its recipe (and from the content for individual ingredients) is as simple as a weighted sum:

1. Obtain the nutritional content for each ingredient
2. Multiply this with the amount per ingredient (the recipe)
3. Compute the total content (per nutrient, over all ingredients)

However, in order for this calculation to be valid for processed foods, more steps are needed. Before being able to list these steps, more attention is needed for the meaning of data.

The ingredient declaration is generally assumed to list the ingredients in their state (and weight) prior to the production process. However, this does not imply that these ingredients are not already processed. A manufacturer may use boiled potatoes as an ingredient and declare it as such ("boiled potato")[3], meaning he relied on someone else to do the boiling. As the recipe to be reconstructed is connected to the state of the ingredient as it is declared, it is important to associate the nutritional content of *boiled* potato to this ingredient (rather than its nutritional value in raw form).
To summarize: the first challenge of processed foods is correct identification of the state of the ingredient, and a connection to the nutritional values of the ingredient in this state.

The second challenge of processed foods is the processing the ingredients are subjected to during production (no matter which state the ingredients were already in). For simplicity, it is often assumed that all ingredients are subjected equally and simultaneously to a single production process (e.g. boiling), but in reality that is often not the case: ingredients may be treated with separate processes in parallel and later be added together. As described by Reinivuo et al. (2009) the effects of processing can be incorporated into recipe calculations by using two factors:

- the weight yield factor (WYF) – designates the change in weight, as a consequence of the processing method (e.g. evaporation of water or uptake of cooking medium). A change in weight causes a relative change in the concentration of nutrients. The WYF differs per processing method, and type of food, but may also differ per ingredient. The latter level of detail is not covered by the harmonized approach proposed by EuroFIR as per Reinivuo et al. (2009), nor by the current version of our algorithm.
- The nutrient retention factor (NRF) – designates the change in nutrient content as a more direct consequence of processing (e.g. heating may cause chemical disintegration of certain nutrients). The NRF can differ per nutrient, per ingredient (it is contained in) and per processing method (processing parameters, such as the length of heating are ideally taken into account).

Taking the above into account, calculation of processed foods changes into the following steps:

1. Identify the processing method the ingredients were subjected to
2. Obtain the nutritional content for each ingredient (in its state prior to the food's production process)
3. Multiply this with the amount per ingredient (the recipe)
4. Multiply the result with the NRF applicable to (a) the processing method (b) the ingredient and (c) each specific nutrient therein
5. Compute the total content (per nutrient, over all ingredients)
6. Divide the total by the WYF applicable to the processing method and the type of food[4] (loss of weight causes an increased concentration of nutrients; increase in weight vice versa)

While the abovementioned steps are valid, they are not the preferred way of working. Reinivuo et al. (2009) state that in case primary data on nutritional content (e.g. lab measurements) are available for

[3] In practice it often occurs that manufacturers do not explicitly describe the state of the bought ingredient, making correct identification of the ingredient state even more difficult.
[4] Adding additional detail in the form of a WYF per individual ingredient is redundant in this situation, as the genetic engine is optimizing for nutrient content (which is only proportional to the total weight of the product, not to that of specific ingredients)

processed ingredients, these are preferred over calculating the effects of processing. This translates to the following steps to be taken instead (if such primary data is available):

1. Identify the processing method the ingredients were subjected to
2. Obtain the nutritional content for each ingredient in its state after processing
3. Multiply the amount (recipe) per (unprocessed) ingredient with the WYF applicable to (a) the processing method and preferably also to (b) the ingredient (in its state prior to processing), resulting in the amount per processed ingredient
4. Multiply the amount per processed ingredient with its corresponding nutritional values
5. Compute the total content (per nutrient, over all ingredients)

Both of the above ways to calculate processed foods require rather advanced data management. A critical aspect for both is correct identification of the type of processing applied to the product (if any). This is often very challenging, given the currently available data on commercial food products. The second critical aspect is access to a fairly large and detailed dataset of WYF factors and either (a) NRFs or preferably (b) measured ingredient nutrient content. Linking all of this information to the identity of ingredients in commercial food products is also not trivial.

While the genetic algorithm developed is technically prepared to support both strategies for processed foods, the abovementioned data challenges resulted in the decision to focus first on unprocessed[5] foods.

2.2.3 Internal representation of recipes by absolute amount (codec 1)

The simplest form to represent a potential recipe solution is a series of values, one for each ingredient for which the amount is unknown. The way a problem is encoded into a genetic algorithm may be referred to as a 'codec' or 'encoding'. The specific codec discussed here is named 'absolute amount'. In genetic algorithm jargon, the place for a single ingredient amount is called 'gene' and a collection of such for the recipe is called a 'chromosome'. Keep in mind that in the real world, some ingredient amounts *are* known in advance, which could serve as an additional constraint on potential solutions. Only for unknown amounts a gene will be presented on the chromosome. This, and the fact that products differ in number of ingredients, results in different lengths of the chromosome for different products. Figure 3 illustrates how the purple column with a potential solution to the recipe maps to the genes in a chromosome.

To simplify the multiplications needed for recipe calculation (section 2.2.2) all ingredient amounts were converted to a concentration ratio between 0 to 1 (covering 0 to 100%). Nutritional amounts are not present on the genome, but required for recipe calculation inside the fitness function and usually expressed in g per 100g of product. In order to avoid the need for repeated unit conversions (between gram, milligram, microgram per 100g) they too were converted to concentration ratio between 0 and 1 (the latter meaning 100g/100g). A special case was posed by energy content which, expressed in its original unit (kCal or kJ per 100g), would have a disproportionate impact on Euclidian distance metric and therefore dominate the optimisation process. In order to mitigate this problem, energy is converted to a new measure with scale and bounds similar to those of other nutrients. This is achieved by dividing all energy values (in kJ or kCal per 100g product) by the theoretical maximum (900kCal or 3700kJ per 100g), based on the most energy-rich nutrient (fat) from the conversion table of Annex XIV of the EU regulation 1169/2011 (European Union, 2011).

[5] The term processing in this context refers to only those processes that cause changes in weight of the product (WYF) or damage to nutrients (NRF).

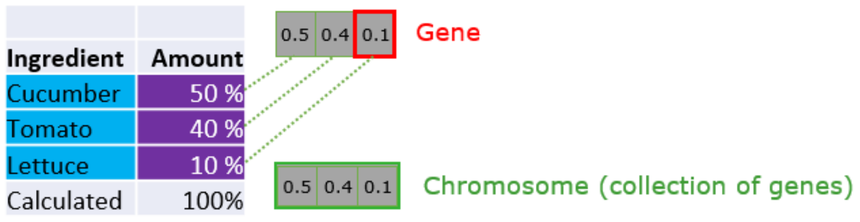


Figure 3 Internal representation of a potential solution

2.2.4 Constraint and repair function

The logical limits for any gene in the abovementioned codec are 0 and 1, and were obeyed at any time during the evolution process inside the genetic algorithm. However, despite this constraint, many solutions can be generated for which we know in advance they will be invalid to the problem. This relates to two additional rules of a-priori knowledge (see section 1.3):

1) Ingredients are in decreasing order of weight, until they cover less than 2%, after which they are allowed to be listed in random order
2) The amounts of all ingredients together should add up to 100% (or 1, using the algorithms internal representation of amount)

In order to bring these rules into the algorithm, a constraint function was implemented. The constraint function is called every time after a new individual is created, either when generating the initial population, or during the process of procreation. The constraint function performs a logical check on both knowledge rules and executes a ‘repair’ action if they are not met, hereby assuring that only valid potential solutions enter the evolution process.

The repair method sequentially works through the genes of a chromosome, checking for each if its current value still allows to meet the above criteria for the whole recipe. Note that the recipe can also include a-priori known ingredient amounts (already declared on the label). These will not be represented on the chromosome (because the solution is already known), but still serve as additional constraint. If a gene value does not allow for the whole recipe to meet the above criteria, the logical upper and lower bounds that *do* allow this are calculated and within these bounds a new random value is generated for the gene.

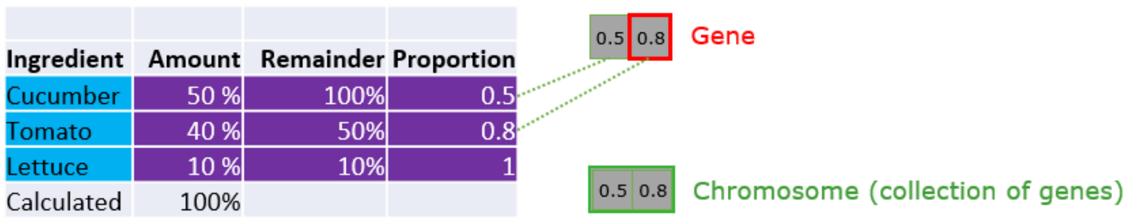
2.2.5 Internal representation of recipes by relative remainder (codec 2)

While representation of a recipe in the absolute amount codec is easy to understand, it is not necessarily efficient for the genetic algorithm. The constraint and repair function keeps invalid solutions away from the evolution process, but it would be better to avoid creating them altogether. While this is not completely possible given the complexity and variation in the problem, an improvement can be achieved with the following codec.

The relative remainder codec represents the solution to the recipe as a series of values that relate to the proportion of the remainder that is covered by the gene (see Figure 4). At the start of the recipe, the remainder is equal to 100%. In order to represent an absolute amount of 50% for the first ingredient, the gene takes a proportion of 0.5 from the current remainder. As the first ingredient now covers 50%, the remainder for the rest of the recipe is down to 50%. In order to represent an absolute amount of 40% for the second ingredient, the gene takes a proportion of 0.8 from this new remainder (50%). In line with the rules presented in section 2.2.4 we want the recipe to add up to 100%. This implies that the last ingredient should always cover the full remainder, translating to a proportion of 1. As this value never changes it does not require to be represented in the gene, as long as it is taken into account in the calculation of nutritional values.

By converting a relative remainder representation to an absolute amount representation, the same constraint and repair mechanism (described in section 2.2.4) could be applied. The resulting logical bounds are translated back into the representation of the relative remainder, after which a new random value is generated within these bounds, in case the constraints aren’t met.

Thanks to this alternative codec, the knowledge rules described in section 2.2.4 are more frequently met from the start for newly generated potential solutions. This means the repair function is working less hard to deal with invalid solutions, making the genetic algorithm more efficient.



Ingredient	Amount	Remainder	Proportion
Cucumber	50 %	100%	0.5
Tomato	40 %	50%	0.8
Lettuce	10 %	10%	1
Calculated	100%		

Figure 4 Internal representation of a potential solution with a relative remainder codec

2.3 Performance evaluation

2.3.1 Evaluating the genetic algorithms

In order to evaluate the performance of the algorithms developed, a dataset with product label information along with the original recipe (ground truth) would be required. However, recipes from food industry are generally considered very sensitive information, and almost never shared. Recipes more commonly available are often targeted at consumers, who do not have access to the same type of processes and ingredients used in food industry. Hence, recipes from cookbooks would not be representative. For that reason, another approach was chosen to evaluate the algorithms developed.

2.3.1.1 NEVO test set

Arguably the most important aspect in any optimization problem is how well the problem is determined. In our case, this varies from product to product. The best scenario is a product with only few ingredients (unknown amounts to be reconstructed) and a highly detailed description of its nutritional values (constraints on the solutions). On the other end of the spectrum we would find a product with hardly any nutritional values specified, while having many ingredients. It becomes even more difficult when some of these ingredients happen to be interchangeable from the perspective of the (few) nutritional values specified for the whole product. The result of such an underdetermined problem is that there will be many different yet equally valid solutions. There is simply not enough information to tell which is the right solution. Our first validation approach is aimed to quantify exactly how problematic underdetermination is.

With the help of RIVM experts a dataset of 44 test cases was compiled from NEVO recipes (these are derived from consumer recipes). The nutritional content of the final product, resulting from these recipes, was always calculated via the same principles described in section 2.2.2. Even though the current version of the algorithm is not yet capable of dealing properly with the effects of processing, recipes requiring this were included in the test set. This needs to be taken into account when interpreting the results. With the selection of test cases, we aimed for maximum variability in the type of products, and number and type of ingredients. While NEVO contains rich information on nutritional values of these products, we deliberately restricted the nutritional information of the total product to the seven most commonly required nutrients on a product label. These are:

- Total (metabolisable) energy (kJ)
- Total fat
- Total fatty acids
- Carbohydrates
- Total sugars
- Total protein
- Sodium (mathematically derived from salt where needed)

In this manner we simulated the underdetermined reality of food product labels. Before handing the information to the reconstruction algorithm, all ingredient amounts were obscured and used afterwards to judge the quality of the reconstruction. An error measure was calculated for each test case, aiming to quantify the deviation between reconstruction and original recipe. This error was calculated as the sum of the absolute errors per ingredient. As ingredient amounts can be expressed in percentages, their error can be expressed in percent point (pp). Subsequently, the average and standard deviation of the errors per test-case were computed to arrive at a performance metric for the whole test set. This procedure was repeated for both codecs, allowing a comparison between them.

2.3.1.2 LEDA test set

While the abovementioned strategy is helpful in gaining insight in the effects of problem determination on reconstruction quality, it cannot be considered a very realistic scenario. In order to come closer to real world scenarios a second validation strategy was developed. For many commercial food products the amount of a few ingredients is known in advance. We refer to this as a partial recipe. This is often a consequence of product label regulation or marketing strategies (e.g. for healthy products). A dataset containing 78 products was compiled with partial recipes of commercial products from LEDA for use as test cases, based on the following selection criteria:

- At least 40% of the number of ingredients has a priori declared amounts (to be used as ground-truth).
- Nutritional values are sufficiently known for individual ingredients, following criteria described in Section 2.5.3.
- Equal representation of all products groups (or as much as possible, given other criteria)
- Products for which processing presumably had little effect on total weight (e.g. WYF close to 1)
- Products with highly similar ingredients (e.g. all vegetables) as well as highly mixed ingredient types
- Products with and without nested ingredients

In contrast to the previous approach, products presumably requiring large processing effects to be taken into account were avoided. As a consequence, some product categories are more strongly represented than others. Before handing the test cases to the reconstruction algorithm, any known ingredient amounts were obscured and used afterwards to judge the quality of the reconstruction. The same error measure, as described in section 2.3.1.1, was computed with one exception. The error for each individual test case was normalized by the sum of a priori known ingredient amounts. The reason for this is that the ground truth per test case is only partially known, and may be more complete for some cases than for others, which would otherwise skew performance measures.

2.3.1.3 Evaluating reconstructions without ground truth

In real world application of a recipe reconstruction algorithm, there is no ground truth. This calls for other ways to gain insight in the quality of the reconstruction, to decide if it was good enough to be used (for example for the use-cases described in section 1.2). In order to judge the quality of a reconstruction, several measures are available. The fitness of the final solution is the first piece of information providing insight. A high distance between the nutritional content of the product and the calculated recipe implies an unreliable result. The reverse however, is not necessarily true in an underdetermined problem. As the genetic engine optimizes for a low distance value, most final solutions have a rather good fitness. However, as the problem is underdetermined, it is possible there are multiple different solutions almost equally fit. In an attempt to quantify how different these solutions can be, a spread metric was implemented as second piece of information for evaluation. Spread is calculated both in the domains of ingredient and nutrient amounts over the 30 best solutions found by the genetic engine. The highest and lowest amounts encountered for individual ingredients and nutrients are stored, providing insight in how different alternative near-optimal solutions can be. A low spread implies a well-determined problem and more reliable result than a high spread. It must be noted that spread for nutrients is only of interest for imputed missing nutrients, as nutrients declared on the product label were the target to optimize for (making spread meaningless).

2.4 Data & knowledge organisation

The genetic engine described in section 2.2 cannot operate without data. Providing it with the required data, requires easy (automated) access and interoperability between existing data sources such as the NEVO table and the LEDA database. In order to meet these requirements for recipe reconstruction, as well as those of the parallel project named 'Food Matching', a set of data models was developed that comply with FAIR (Findable, Accessible, Interoperable, and Reusable) principles (Wilkinson et al. 2016, GOFAIR). To this end a set of ontologies was created that represent the data models. Software was written to translate existing data (e.g. NEVO table, nutrient information, and recipes) into the data model defined by these ontologies. Each data source has an equivalent ontology containing the data. As both the data model and data ontologies are specified in RDF (Manola & Miller, 2004), the data can be accessed using graph data bases (triple stores) or directly using RDF files. The SPARQL query language (Harris and Seaborne, 2014) provides easy access to the data using an SQL-like query language. SPARQL is used by the recipe reconstruction tool to access the data needed for its operation.

Ontology		Description
nevo-syntax.ttl	Data Model	Defines the concepts and relations in which the NEVO table is encoded. It contains concepts such as nevo:FoodProduct, nevo:Packaging, and nevo:PreperationMethod, and relations such as nevo:hasNEVOGroup.
nevo-nutrients.ttl	Data Model	Defines the concepts and relations needed to encode nutrient data into the NEVO table ontology. It defines one concept (nutr:NutrientQuantity) and two properties (nutr:hasComponentCode and nutr: hasNutrientQuantity). It inherits from concepts and relations defined in the Ontology of Units of Measure (Rijgersberg et al., 2011), allowing for unit transformations.
nevo-recipe.ttl	Data Model	Defines the concepts and relations needed for encoding recipes. It contains concepts such as nevorcp:Recipe and nevorcp:Ingredient.
nevo.ttl	Data	The NEVO table encoded in an ontology with the concepts as defined in the nevo-syntax.ttl data model ontology. It contains the food products defined in the NEVO table.
nutrients.ttl	Data	An extension of nevo.ttl, where the food products defined in nevo.ttl are connected to their nutrient data.
recipe.ttl	Data	Contains the recipes for the food products (defined in nevo.ttl and nutrients.ttl) defined by their ingredients including the (relative) amounts of each ingredient as calculated by the recipe reconstruction tool. This is the output and result of the tool.

Table 3. The ontologies for data and data models used in the recipe reconstruction tool

2.4.1 Data Model

The data model (see Figure 5) is defined in three ontologies specifically developed for the Food Matching and Recipe reconstruction projects and generic ontologies such as the Ontology for Units of Measure (OM, Rijgersberg et al., 2011). It also uses concepts and relations defined by the World Wide Web Consortium (W3C) for the Resource Description Framework (RDF, Manola & Miller, 2004 and RDFS, Brickley & Guha, 2014) and the Web Ontology Language (OWL, W3C OWL Working Group, 2012).

The main data model ontology is nevo-syntax.ttl, which defines several concepts such as nevo:FoodProduct, nevo:Diet, or nevo:Packaging. It also defines several NEVO specific properties such as nevo.hasNEVOCode, nevo:hasNEVOName, and nevo:isComposed, used to specify the NEVO code, the NEVO name as used in the NEVO table, and whether a food product is a composed product, respectively. Properties defined in, for instance, the RDF specification (Manola & Miller, 2004) may also be used.
The important concept in this ontology is nevo:FoodProduct. All food products in the NEVO table are defined as an instance of this concept. Categories and subcategories (classes) may be defined as subclasses of this concept. Instances and classes are defined in the data ontologies.

The second data model ontology is nevo-nutrients.ttl, which defines only two concepts: the class nutr:NutrientQuantity and the property nutr:hasComponentCode. The class nutr:NutrientQuantity defines a quantity in which a nutrient measurement is described. It is a subclass of om:Quantity and it thereby inherits the properties and functions of the Ontology of Units of Measure (OM, Rijgersberg et al., 2011). This allows, for instance, for the conversion to different units (e.g. gram to ounce). The property nutr:hasNutrientQuantity is the inverse of the OM property om:hasPhenomenon and has nevo:FoodProduct as domain. Each food product can, of course, have multiple nutrient quantities associated with it.

The third data model ontology is nevo-recipe.ttl, defining the classes nevorcp:Recipe, representing a recipe, as a list of ingredients with amounts, and nevorcp:Ingredient, representing an ingredient of the linked food product. Instances of nevorcp:Ingredient should also be associated with an om:Quantity type, most often om:Weight or om:Volume. These instance of nevorcp:Ingredient should then be linked (with om:hasValue) to an instance of om:Measure with a numerical value and unit. For a graphical representation, see Figure 6.

An ingredient (nevorcp:Ingredient) is also linked to a nevo:FoodProduct using the property nevorcp:ofFoodProduct. This represents the type of ingredient, which is in itself also a food product with associated nutrients and maybe other ingredients. For instance, a NEVO food product representing an apple pie has (among others) an ingredient apple. Both apple pie and apple will be NEVO food products. The ingredient representing apple will be linked to the food product apple with the property nevorcp:ofFoodProduct.

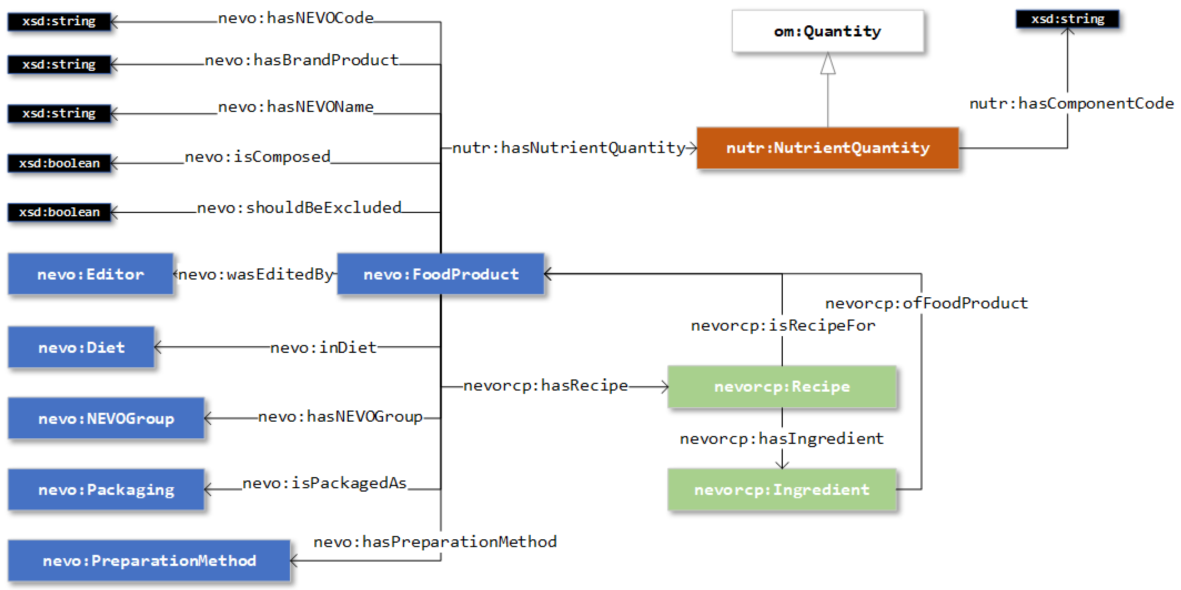


Figure 5. The structure of the data model ontologies used. These ontologies define important concepts and properties that are used to describe the data. Blue is used for concepts in the NEVO syntax ontology, green for the concepts in the recipe ontology, and red for the nutrient ontology.

2.4.2 Data

The data from the NEVO table is automatically generated and written to the ontology nevo.ttl. Nutrient information for each of the entries in the NEVO table is written to nutrients.ttl. The output of the recipe reconstruction tool is written to the recipe.ttl ontology. To define a food product such as fried potatoes (see Figure 6), a new concept is created that has a type-of relation (rdf:type) with the concept nevo:FoodProduct defined in nevo-syntax.ttl. The NEVO code (1457), NEVO name ('Aardappelen gebakken' in Dutch and 'Potatoes Fried' in English), and other properties such as alternative names ('gebakken aardappelen') are connected to the food product. These properties link to information in string format. Other properties, however, link to other instances, such as the preparation method nevo:Gebakken, which is an instance of nevo:PreparationMethod.

The nutrients of a food product are linked to the food product using the properties nutr:hasNutrientQuantity and in reverse om:hasPhenomenon. In the example of Figure 6, only one nutrient quantity is linked to the fried potatoes, energy in kilocalories. Normally, the food product will be connected to a whole host of nutrient quantities. As the nutrient quantity is also an OM quantity, it will also be connected to an instance of om:Measure representing the amount of the nutrient present in the food product and the unit used (usually per hectogram). The nutrient quantities provide the values of the nutrient vector for a food product used in the genetic algorithm.

Each food product can be linked to a recipe that describes the input ingredients referred to in the ingredient declaration. Each ingredient is assumed to also be represented as a NEVO product in the NEVO table. So in the example, fried potatoes (NEVO code 1457) has as one of the ingredients boiled potatoes (NEVO code 982). The example in Figure 6, shows only one ingredient for fried potatoes, but, of course, many more ingredients can be added.
The instance of nevorcp:Ingredient is also of type om:Quantity, or more specifically a subclass of om:Quantity in this case om:Weight. When the recipe reconstruction algorithm is run for this recipe, the algorithm tries to determine the amounts (weights or volumes) of each ingredient that was used to create the food product. These amounts can be added to the ingredient using OM. In the case of this example, the ingredient boiled potatoes (connected with the food product representing boiled potatoes using the nevorcp:ofFoodProduct property) has a value which is an instance of om:Measure (_:node1ea2ive1mx113440, using a blank node identifier). This value defines a numerical value 1.0E3 in the unit of om:gram. The calculated amount added to the recipe for 1kg of fried potatoes. If we look at the output data ontology generated by the algorithm, we will see that another ingredient (unsalted margarine) was used with an amount of 60g, but this is not shown in Figure 6 for simplicity.

The data models defined in ontologies can be used to store the input information and (intermediate) results of the recipe reconstruction algorithm. It uses generic and externally supported ontologies such as OM to define the data, so that the data becomes findable and accessible (because the concepts and properties used such as units are clearly defined), interoperable (with other data sets that used the same generic ontologies), and reusable.

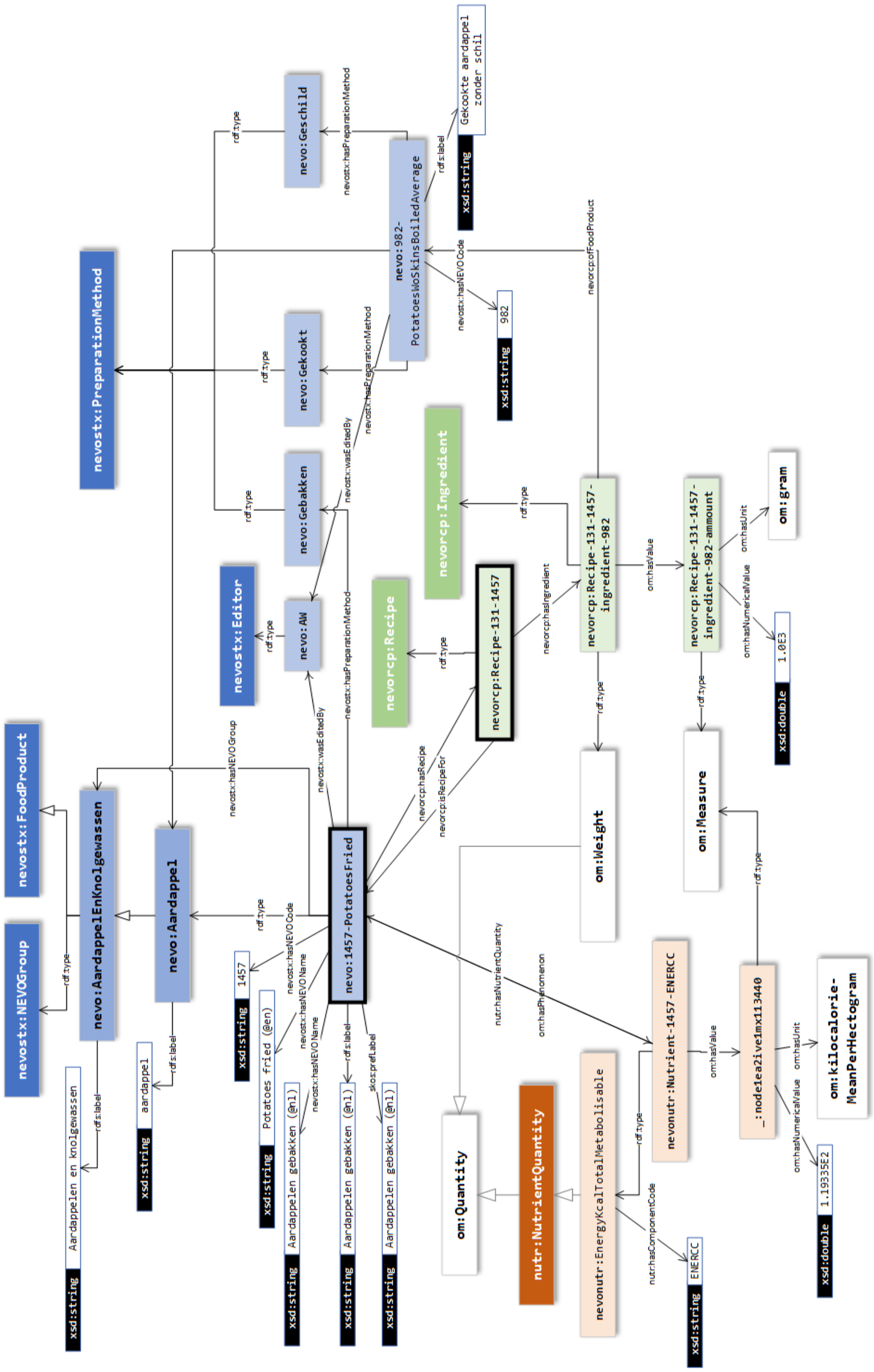


Figure 6. An example showing the different concepts used for defining a food product for fried potatoes. Not all concepts and properties are shown, only one nutrient and only one ingredient are shown, while more are present. The ingredient 'boiled potatoes' will also have its own nutrients and ingredients, which are not shown in this diagram.

2.5 Pipeline

The previous sections have covered the genetic engine and the relevant data sources for it. This section describes how these building blocks were tied together to obtain a prototype of an automated system. The whole pipeline is visualized in Figure 7 and specific elements will be addressed in the following sections.

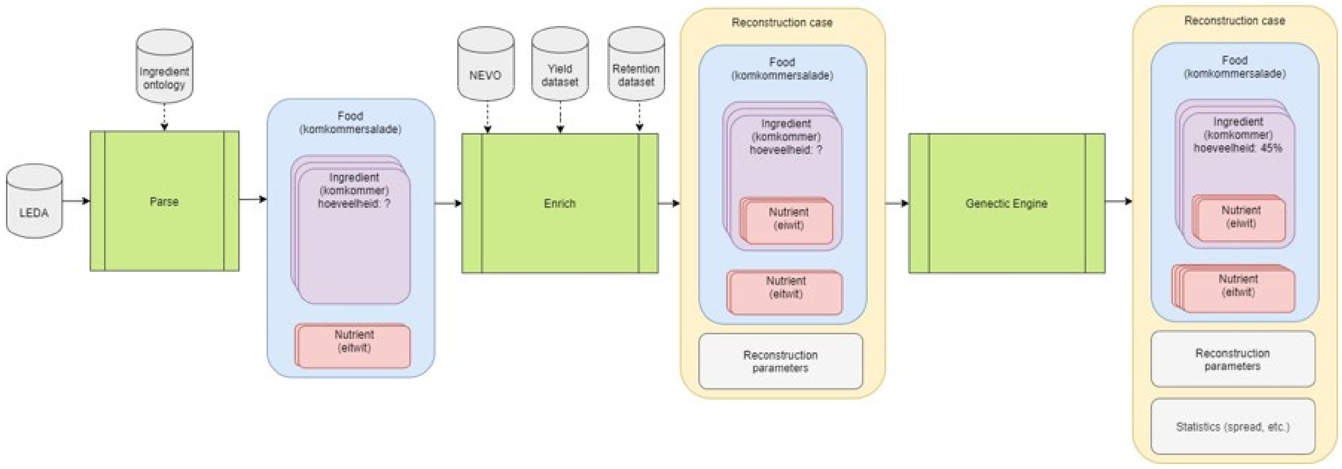


Figure 7 Visualisation of the pipeline built around the genetic reconstruction algorithm

2.5.1 Ingredient declaration parser

The ingredient declaration of branded food products is an important source of information for recipe reconstruction. Unfortunately this source of information is typically unstructured: a free text field that can contain anything (including text unrelated to ingredients, phrases in a different language and spelling and punctuation errors). In order to extract useful information from the ingredient declaration, 'parsing' is needed: turning unstructured data into structured data. The parser used for this was developed in another project (Vlek & Willems, 2019), but repurposed and tuned to this specific application. Key aspect of this tuning was to make a connection between specific NEVO codes and potential ingredient names as they occur in commercial foods. The sections below aim to describe the key mechanisms by which this parser operates. Additional details can be found in Annex B.

The main purpose of the parser for this application is the extraction and identification of individual ingredients in their original order from the ingredient declaration. Identification is crucial for linking each ingredient to nutritional data in other sources. As was discovered in a previous project by a term frequency analysis on a large database of ingredient declarations (Vlek & Willems, 2019), roughly 1500 unique terms are needed to cover 90% of the ingredient declarations of branded food products. When taking into account the fact that some of these terms actually refer to the same substance (e.g. synonyms, plural forms, alternate spellings), the number of unique ingredient entities can be reduced to approximately 1200. A food product ingredient ontology (FPIO) was constructed from this list, following the principles of semantic web (see section 2.4), and additional labels were added to the terms for which synonyms, plural forms or alternate spellings were identified (see Table 4).

The basic FPIO was further enriched with the names of EU approved food additives (E-numbers) obtained from WikiPedia (2019), as well as with a list of common functional ingredient group names (e.g. preservatives, anti-oxidants, etc.).

Table 4 Simplified example of information on a carrot in the Dutch FPIO

Concept name and preferred label	**Alternative label**	**Plural form**	**Alternative plural form**	**NEVO code**
wortel	peen	wortels	wortelen	71

After receiving the ingredient declaration of a product, which is assumed to be in the Dutch language[6], the parser performs the several steps to check and sanitize the data, and chunk the individual ingredients from it (detailed steps described in Annex B). When additional specifications of an ingredient, so called ‘nested ingredients’, are encountered, a tree-like data structure (parse-tree) is created to keep the nested information associated to the ingredient it belongs to. Amounts declared for specific (nested) ingredients are also recognized by the parser and stored separately in the parse tree. They will provide future constraints to the recipe reconstruction process for the other ingredients of which amounts are still unknown.

As a final step the parser aims to establish a (case-insensitive), fuzzy match[7] based on the Jaro-Winkler edit distance (Winkler, 1990), to any of the labels (including alternative names and plural forms) of the ingredient entities in the FPIO[8]. This approach introduces some robustness against spelling variations and errors. If the edit distance of the best match would fall beyond a predefined threshold distance, no match would be established. Determining the best value for this threshold means finding an acceptable balance between two types of errors:

1. False positively matching to the wrong FPIO ingredient. The result of this is that the genetic engine will be supplied with nutritional information about the wrong ingredient, from which the quality of the reconstructed recipe will suffer. How much the quality suffers partially depends on the position of the respective ingredient in the declaration.
2. False negative: failing to recognize an ingredient as an FPIO entity. The result of this is that the genetic engine cannot optimally exploit all available information, in this case the nutrient content of the respective ingredient. In absence of this nutrient content, the algorithm will still try to estimate an amount for the respective ingredient based on its knowledge of surrounding ingredients.

Thanks to the nature of knowledge graphs, knowledge linked to FPIO entities can be easily applied, once these entities are recognized in the ingredient declaration. In this fashion specific NEVO codes were associated to FPIO ingredient entities, allowing for a mapping to the nutritional details in NEVO where possible. These mappings were established by RIVM experts, but this revealed some issues:

- ingredients are sometimes declared in a rather generic way, making multiple NEVO codes equally likely. For instance when the preparation state of an ingredient is unclear (was it already boiled or not).
- many industrial ingredients are not present in NEVO, often they are additives representing a small part of the food’s total weight.

In addition to NEVO relations, several specific sub-classes of ingredients were conceived, allowing to raise a flag in the software when any member of these classes would be encountered. This was done for the ingredient sub-classes of vitamins, minerals, fibres and E-numbers.

While parsing ingredient declaration from the whole LEDA database, statistics were kept of the frequency and relative position (in the declaration) of terms successfully mapped to FPIO. Statistics are also kept for unmapped terms, allowing future improvements to be directed to the most frequent or most important (early in the declaration) terms.

[6] Even though LEDA contains products intended for the Dutch market, this assumption is not always correct. Language recognition could be considered as a future improvement to catch products not in line with the assumption.
[7] A fuzzy match is different from an exact match in that it is robust against small variations in spelling (including spelling errors).
[8] The Jaro-Winkler metric was chosen because it is better suited to comparing shorter strings (e.g. ingredients names) than the commonly used Levenshtein distance (and derivatives).

2.5.2 Nutritional data enricher

The structured ingredient data received from the parser is further interpreted and enriched in a subsequent process (see Figure 8). With NEVO codes already associated to FPIO entities, obtaining NEVO data on each ingredient's nutritional content is straightforward. This data is added to the ingredient data structure. For nested ingredients a special approach was developed. If the 'parent' ingredient can be associated to a NEVO product directly, all of the nested information associated to it becomes redundant and can be ignored. If it cannot however, the nutritional content of the 'parent' needs to be derived from the nutritional content of its sub-ingredients ('children'). In an ideal world, this would call for recursive recipe reconstruction on the nested information, but without nutritional details on the parent ingredient this is impossible. Instead, a more naive method was implemented to estimate the contribution from the nested ingredients (if not declared). This method is described in more detail in Annex C. With the estimated amounts and nutritional content (from NEVO) of each sub-ingredient, the nutritional content of the parent ingredient could be calculated and nested information can be ignored for the rest of the recipe reconstruction process.

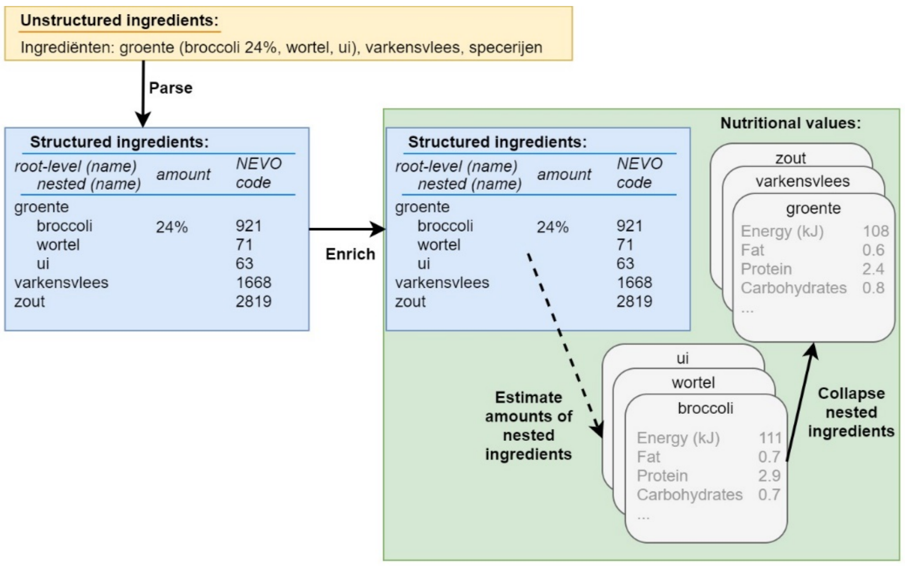


Figure 8 Example of how data is prepared for the genetic algorithm

As mentioned, nested ingredients pose a special challenge for parsing correctly. However, during the project it was discovered that nested ingredients for which amounts are declared, hold an even bigger challenge. Amounts are declared as a percentage, which is a relative metric. When declaring a main ingredient its amount is relative to the total weight of the product (representing 100%). In case of nested ingredients there seem to be two definitions being used in parallel by manufacturers as was observed from LEDA data:

1. The amount of a nested ingredient is declared as a percentage of the total product
2. The amount of a nested ingredient is declared as a percentage of its parent ingredient

This situation leads to confusion when interpreting the amounts of nested ingredients. In some cases it was possible to disambiguate between definition 1 and 2, based on data and logic (including the decreasing order constraint). For instance when the amount of a parent in ingredient is declared, and is exceeded by the sum of its sub-ingredients, the latter amounts are probably referenced to the parent ingredient. From the cases that could be disambiguated in this way it was established that the first definition is the most popularly used. Because of this, it was made the default for all ambiguous situations.

2.5.3 Criteria for reconstruction

As explained in the previous section, establishing a NEVO connection is not always possible for the ingredients in a branded food. Some ingredients may not even be recognized as any of the FPIO entities. Both cases lead to the same situation: no nutritional data for the ingredient while it is required by the genetic engine. This situation was discussed with RIVM and the following criteria were conceived to decide when it is acceptable to ignore the contributions of a specific ingredient to the nutritional content of the product without affecting the quality of the reconstruction too much[9]:

- If it concerns a *nested* ingredient and has an (estimated) amount of less than 50% it may be ignored.
- If it concerns a 'root-level' ingredient (i.e. not nested):
  - If the product contains 4 or fewer ingredients, it may never be ignored.
  - If the product contains more than 4 ingredients, it may only be ignored when its position is beyond the first 80% of the total number of ingredients, and the number of ingredients ignored so far is not exceeding 10% of the total number of ingredients.

When the above criteria can be met, the missing nutritional values of the ingredients that were allowed to be ignored are set to zero. This way, the genetic engine still tries to find an amount for it, and imposes its constraints on the possible solutions (i.e. maintaining a decreasing order and normalized recipe), but the nutritional value of the respective ingredient is no longer playing an active part in this: nutritional values of zero imply that all possible solutions are equally valid for this ingredient.

When the above criteria *cannot* be met, the nutritional information is considered too incomplete to attempt recipe reconstruction. Additionally, several other criteria also justify not performing recipe reconstruction on a product:

1. If the product contains fewer than 2 ingredients with unknown amounts. In this case the recipe can be resolved by a simple calculation (which is done).
2. If the product contains an invalid partial recipe (e.g. "Ingredients: 40% apple, 50% pear" can never meet the required decreasing weight ingredient order).

[9] This was a subjective judgement, difficult to objectify, based on nutritional knowledge from RIVM experts as much as possible.

3 Results

As described in section 2.3 performance of the genetic algorithm was evaluated with two complementary approaches. Please bear in mind that a genetic algorithm is by nature non-deterministic, meaning that running it twice on the same data will not yield identical results. Because of this nature, the average performance as well as standard deviation on the datasets are specified.

3.1 NEVO test set

The first approach, using test cases from NEVO, assumes that nutritional values are known for all individual ingredients of a product, and that the total nutritional value of the product can be calculated via the method described in section 2.2.2. The set of nutritional values of the total product is deliberately restricted to a limited number. Table 5 shows the average absolute recipe error in percent point on this test set for the absolute amount codec (3.3 percent point) and relative remainder codec (3.7 percent point). As can be seen, these average errors are well within each other's standard deviations, meaning that there is no significant performance difference between the two codecs. However, there seems to be a slight advantage to the relative remainder codec in terms of computational efficiency (lower execution time), as was expected. It depends on the exact application whether this benefit outweighs the advantage of the absolute amount codec in being more intuitive to understand and easier to explain.

Table 5 Performance of the two different codecs on the NEVO test set

	Codec 1: absolute amount	**Codec 2: relative remainder**
Mean absolute error (percent point)	3.3pp (SD=5.0pp)	3.7pp (SD=6.1pp)
Execution time[10] (s)	253	211

The following graphs depict a single reconstruction case taken from the NEVO test set, where the recipe was reconstructed using the absolute amount codec. This example concerns Tjap Tjoi without rice. While this product commonly requires a form of heating to prepare, the effects of heating are already taken into account by defining ingredients (predominantly) in prepared form (e.g. 'gekookte wortel'). The best recipe solution returned by the genetic algorithm is shown in Figure 9, which has an absolute error of 11.25 percent point with respect to the original recipe (an error above average on this test set). The calculated nutrients of this solution have a fitness distance of 0.00054 with respect to the original nutrients supplied to the genetic algorithm for this product[11] (see Figure 10). Together these figures tell an interesting story: that it is possible to achieve a low fitness distance with a recipe solution that is not exactly identical to the original recipe. This reflects the ill-posed nature of the recipe reconstruction problem, where multiple solutions may be equally valid, given current heuristics, constraints and information about the product.

Figure 11 depicts the nutrients that were never presented to the genetic algorithm, but that were imputed with the reconstructed recipe. Figure 11 and Figure 12 depict the absolute and relative error for each individual nutrient. Graphs of a second example product can be found in Annex A.

[10] Measured on a 64-bit Windows 10 laptop loaded with an Intel i7-6600U CPU @ 2.60GHz and 32GB of RAM
[11] Please bear in mind that fitness distance is calculated on the normalized form of nutrients as internally used by the algorithm. The fitness distance cannot directly be translated to an error in g/100g.

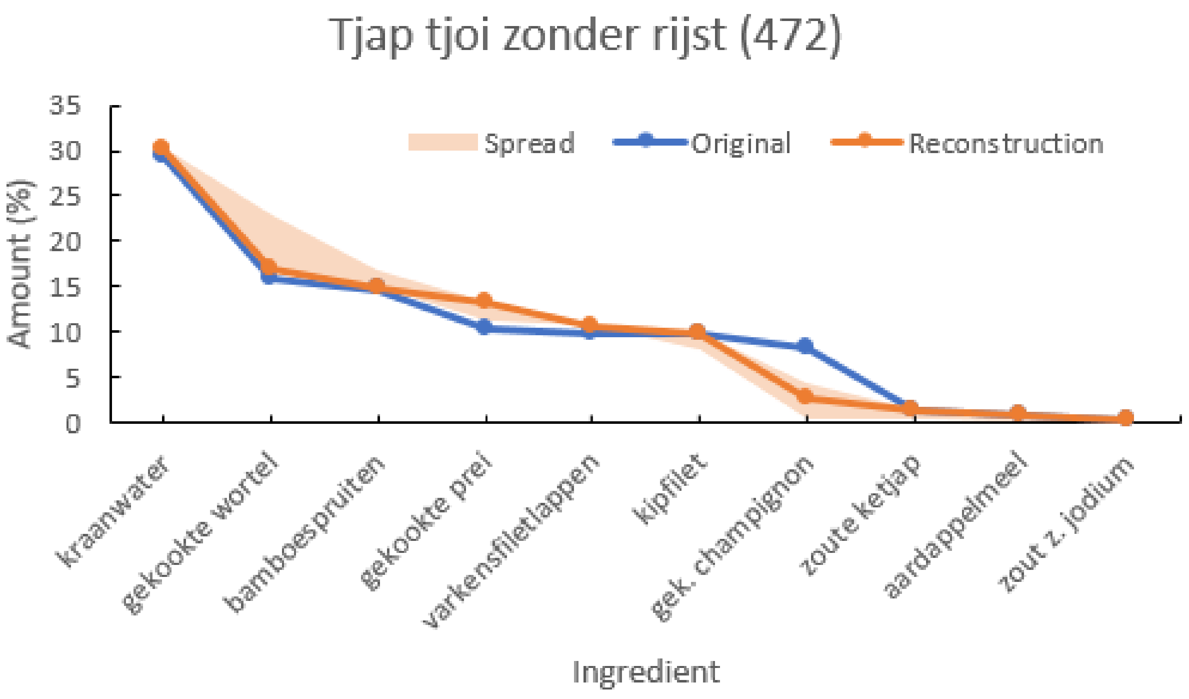


Figure 9 Reconstruction example from the NEVO test set

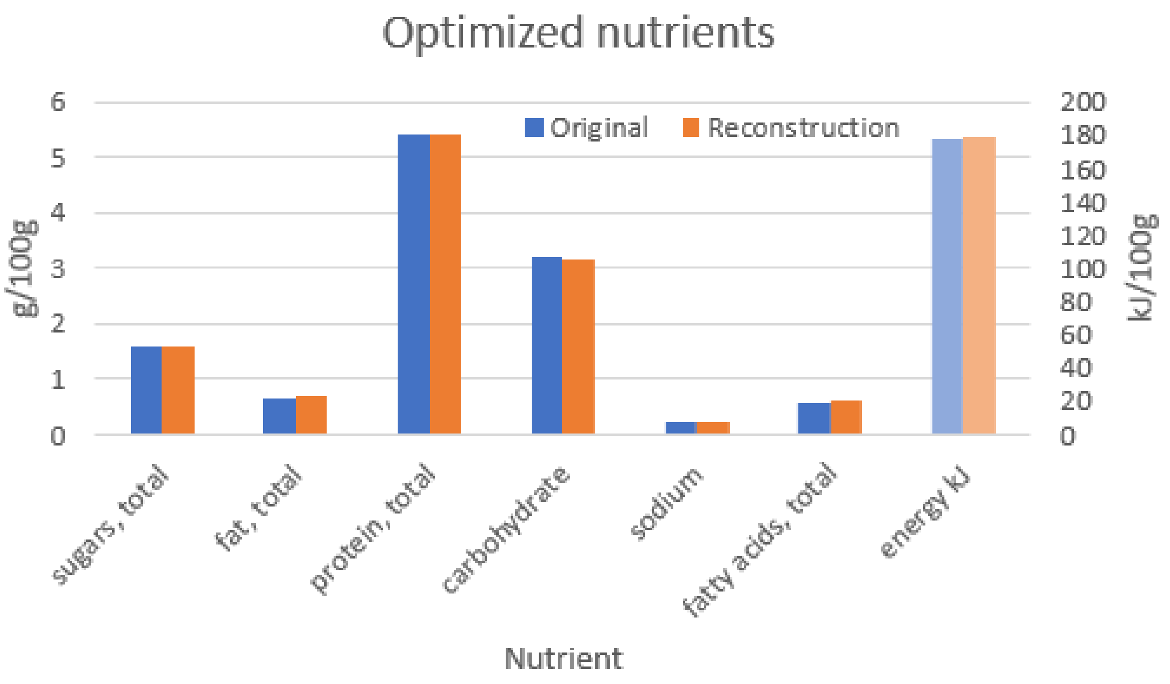


Figure 10 The nutrients optimized for by the genetic algorithm in the above example case. Energy (kJ) is plotted against the secondary vertical axis because of its scale.

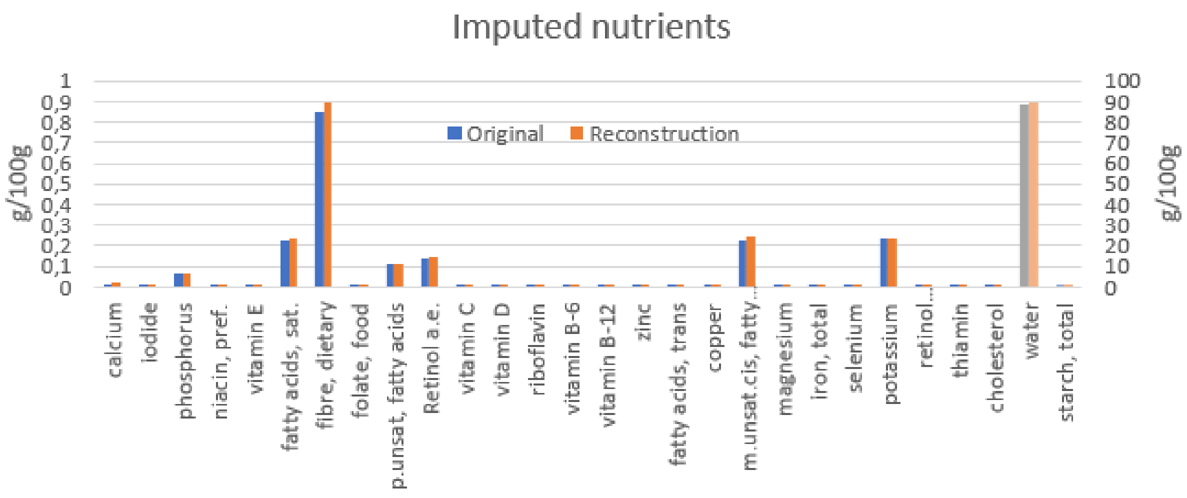


Figure 11 Imputed nutrients for the example case. Water and starch are plotted against the secondary vertical axis because of their magnitude.

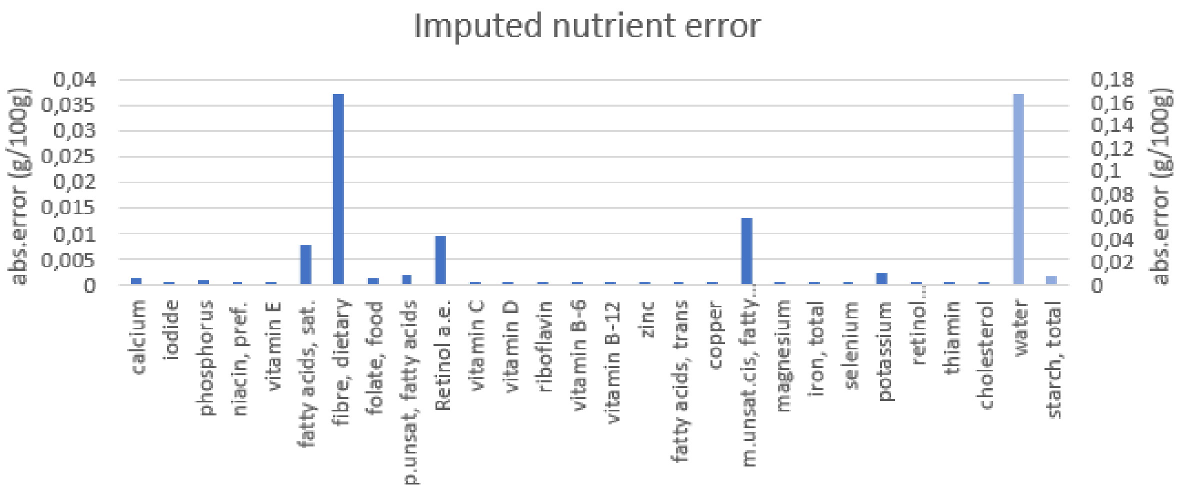


Figure 12 Absolute error on imputed nutrients for the example case. Water and starch are plotted against the secondary vertical axis because of their magnitude.

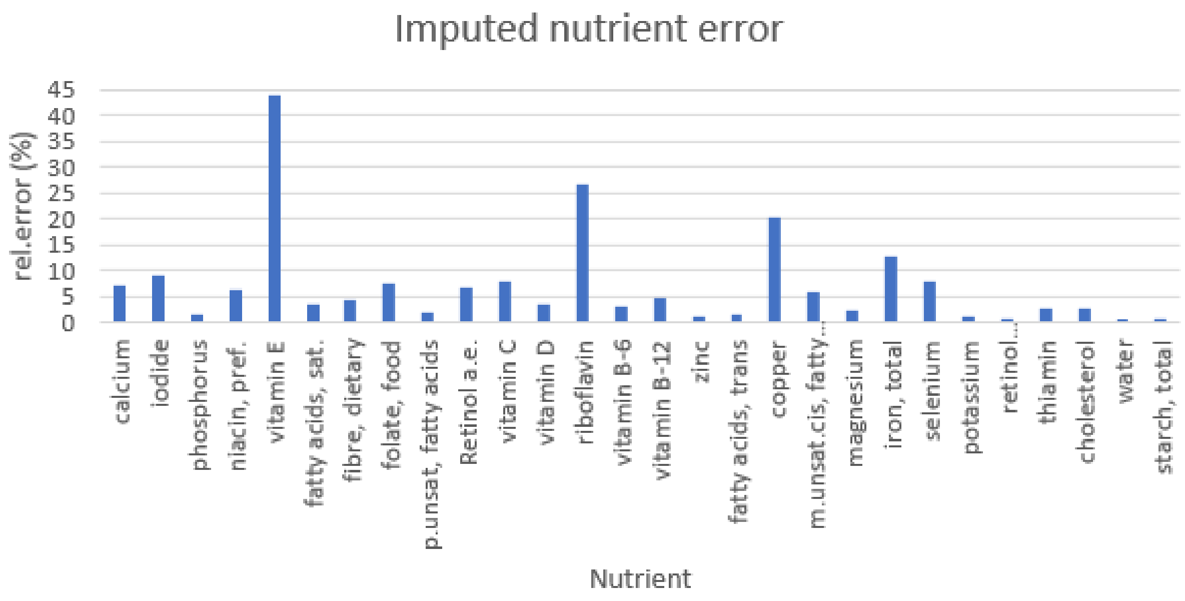


Figure 13 Relative error on imputed nutrients for the example case. Please note: when a nutrient is hardly present a large relative error is more easily made.

3.2 LEDA test set

Table 6 shows the performance of both codecs on the more realistic situation of the LEDA test set. Keep in mind that it is no longer the genetic algorithm in isolation being evaluated, but rather its performance in tandem with the parser and enricher. Also keep in mind that most test cases were only able to supply a partial ground truth of the recipe, meaning that the recipe error needed normalization before it could be averaged over the entire set of test cases. As with the NEVO test set, codec 1 and 2 perform comparably. Interestingly, the absolute error for both codecs has raised almost by an order of magnitude, meaning that real world conditions pose a more severe challenge to the process of recipe reconstruction. Figure 14 shows reconstruction results for two examples from the LEDA test set. For the product ‘Couscous met groente 8 maanden” only a partial ground truth was available, making the blue line discontinuous. This product’s normalized absolute error was 26.87 percent point, and fitness distance of the optimized nutrients was 0.00354. The second example product, called ‘Gourmetschotel 300g’, yields a normalized absolute error of 22.11 percent point and a fitness distance of 0.00211. What is particularly challenging about the second example product is that it contains many ingredients that are largely interchangeable (e.g. paprika in 3 different colours) from the perspective of the set of nutrients available to the algorithm for optimization.

Table 6 Performance of the absolute amount codec on the LEDA test set

	Codec 1: absolute amount	**Codec 2: relative remainder**
Mean absolute error (percent point)	33.5pp (SD=26.9pp)	33.9pp (SD=26.4pp)

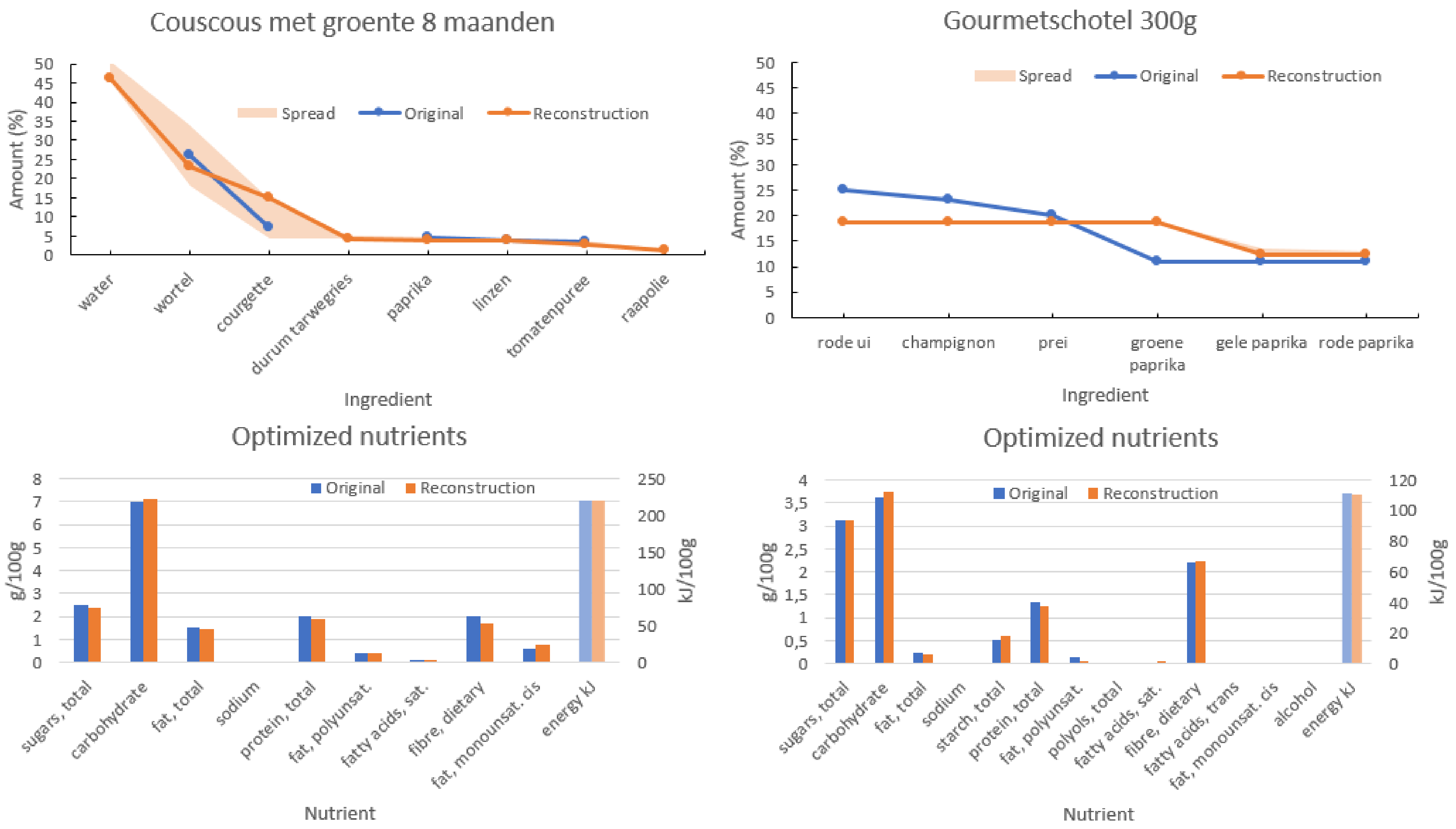


Figure 14 Two reconstruction examples from the LEDA test set (left and right columns), their recipes (top row) and nutrients as optimized by the genetic engine (bottom row). Energy (kJ) is plotted against a secondary vertical axis because of its scale. Spread is sometimes extremely close around the reconstruction curve, making it difficult to see (especially for curve of the Gourmetschotel 300g).

4 Discussion & conclusion

Close inspection of the results on the NEVO and LEDA test sets have revealed several insights. First and foremost, the test cases have demonstrated that genetic algorithms are in principle capable to perform recipe reconstruction when framed as an optimization problem. The flexible nature of genetic algorithms easily facilitates all relevant constraints, such as maintaining a generally decreasing order of weight in the recipe. As expected, reconstruction quality varies with product complexity and with the richness and quality of product label data. Also it needs to be noted that the performance measures obtained on the LEDA test set relate not only to the performance of the genetic algorithm, but to the reconstruction system as a whole, including the parser and the nutritional value enricher. The performance difference observed between the NEVO and LEDA test sets emphasizes an important point: the primary challenge of recipe reconstruction is in posing the optimisation problem correctly, not in solving it. This comprises:

- Associating correct and representative nutritional data to declared ingredients (e.g. the quality of the food label data, parser and nutritional composition database).
- Identifying processing methods the ingredients were subjected to (prior to, or during production).
- The model and data used for calculating the nutritional content of a product, given the amount and nutritional content of its ingredients, and including the possible effects of processing.
- Finding additional data (or a metric) that can help disambiguate nutritionally interchangeable ingredients.

Whether a reconstruction is accurate enough is highly dependent on the intended use of the results. We aimed to make this judgement easier by providing both the fitness distance and spread metrics.

As can be observed from the reconstruction plots in section 3, the spread metric can be informative in quantifying the variations between equally valid recipe solutions. However, it became clear that the current spread metric does not exhaustively capture all equally valid recipe solutions. In the reconstruction plot of the *Gourmetschotel 300g* (Figure 14) this situation is clearly visible: spread is hardly visible, because of being close to the proposed solution, yet the original recipe (ground truth) falls well outside of the spread band. When this happens, the spread metric fails to be informative for judging the reliability of the recipe solution (and of imputed nutrients derived from it). Two factors presumable cause this behaviour of the spread metric. Firstly, genetic algorithms are known for rapid convergence to a near-optimal solution, which is an advantage for finding a single solution quickly, but a disadvantage for exploring solution spread within the search space. Secondly, the repair function partially counters the degree of randomness introduced by the cross-over and mutation mechanisms (and counters this increasingly towards the end of the recipe). This randomness is important for a sufficiently wide exploration of the search space. The spread metric may perhaps improve by customizing the cross-over and mutation mechanism and integrate the repair function inside it. Additionally, the spread metric could benefit from capturing statistics of all solutions within a certain percentage of the best solution’s fitness (rather than of the static best N solutions).

The most predominant reasons for inaccurate reconstructions can be summarized as follows:

- The genetic algorithm cannot meet the total declared nutritional value, still showing a relatively high fitness distance for its best (yet inaccurate) solution. This happens when:
  - Ingredients are not recognized by the parser (often due to lacking data quality), such that their impact on the nutritional value of the product cannot be taken into account.
  - Ingredients are recognized by the parser, but could not be linked to nutritional values from NEVO. Again, their impact on nutritional value cannot be taken into account.
  - Ingredients are recognized and linked to nutritional values, but the effects of processing on nutritional value and weight are not taken into account.
  - The ingredient declaration is ill-defined, for instance when it is unclear if percentages of nested ingredients are relative to the total product or to the parent ingredient.
- The genetic algorithm is able to meet the total declared nutritional value with a relatively low fitness distance, but its best solution is not accurate. A high spread is usually observed on the

recipe amounts in this situation. This happens when the nutrients known for the total product are not well differentiated over its ingredients. Ingredients may be interchangeable from the nutritional perspective (e.g. red and green paprika). This leads to multiple equally valid, yet incorrect, solutions, and there is simply not enough information for the algorithm to tell which solution is correct. This happens sooner with products where only few nutrients are declared. While spread on the recipe may be high, it is not always as bad for the imputed missing nutrients: e.g. the imputed sum of nutrients from both paprikas is not heavily influenced by the balance between the red and green variety.

Several lessons were also learnt during the process of designing a large-scale automated software tool around the recipe reconstruction algorithm. First and foremost, it made explicitly the degree of implicit knowledge and background information involved in manual recipe reconstruction processes. It also emphasized the challenges of unstructured data and the importance of strict definitions for food product data (e.g. how percentages of nested ingredients should be represented) such that they can be unambiguously interpreted. Data and knowledge organisation was a substantial element in this project, and this process has led to interesting discussions as well as the identification of current knowledge gaps.

A method was proposed for expanding the reconstruction algorithm to cover processed foods, but several challenges remain to be solved before it can be implemented. A major challenge of processed foods is the identification of the processing method with a sufficient level of detail to allow for modelling of the impact on both weight and nutritional content. For some food categories processing may be relatively uniform (e.g. bread) or explicitly declared and well standardized (e.g. 'pasteurized'), while for others it may be highly variable and complex. Moreover, it remains a challenge to identify the processing state of the ingredients as they are declared. Early in the project, attempts have been made to include the effects of processing (NRFs and WYFs) as additional variables (genes) to be optimized by the genetic engine. However, the shear additional degrees of freedom this (especially the NRFs) posed to the genetic algorithm immediately showed their effect on performance and led to the conclusion to not further pursue this route. This means identification of processing methods is needed, as well associated data on their effects (WYFs and NRFs).

The large-scale automated recipe reconstruction software developed within this project is a substantial step forward from the original, largely manual, way of working. It is predominantly a step in efficiency, and arguably also in systematicity: manual processes may be performed slightly differently by different persons. At the time of writing it is difficult to judge if the accuracy of the system will always meet the requirements of the various use cases of RIVM, but it certainly aims to meet the use-cases defined in section 1.2.

Suggestions for further improvements can be formulated based on the predominant reasons for inaccurate reconstructions listed above:

- by making use of the term-frequency statistics kept by the parser, frequently occurring (or new) ingredients can be added to the FPIO, such that they are recognized by the system.
- investigation of terms recognized by the parser yet not connected to nutritional contents
- industrial ingredients and additives are not usually covered by NEVO, meaning they are often ignored by the system. In some cases they may contribute substantially to very specific nutrients (e.g. E300 connects directly to the nutrient Vitamin C). When these specific nutrients are also declared for the total product, they become an active part of the optimization process and ignoring the ingredient has a major impact on the reconstruction. For such ingredients a sparse extension of the NEVO ontology could be considered.
- the challenges surrounding processed foods are perhaps best address in collaborative efforts (within EU?), aiming to come to a standardized way to identify and model the effects of processing.
- although RIVM has no direct control over the definitions, regulations and formats associated to food product data it may be valuable to remain in close contact with policy makers, aiming to move manufacturers towards more structured, well-defined, quality controlled food product data.

Literature

Anton, H. (1994), Elementary Linear Algebra (7th ed.), John Wiley & Sons, pp. 170–171, ISBN 978-0-471-58742-2

Brickley, Dan and Guha, R.V. (2014). RDF Schema 1.1. W3C Recommendation, URL: https://www.w3.org/TR/rdf-schema/, Retrieved on August 13, 2020.

Chokr, M., & Elbassuoni, S. (2017). Calories Prediction from Food Images. AAAI.

European Union (2011), Regulation (EU) No 1169/2011 of the European Parliament and of the Council of 25 October 2011 on the provision of food information to consumers.

GOFAIR, "FAIR Principles", URL: https://www.go-fair.org/fair-principles/, Retrieved on August 13, 2020.

Harris, Steve and Seaborne, Andy (2014). SPARQL 1.1 Query Language. W3C Recommendation, URL: https://www.w3.org/TR/sparql11-query/, Retrieved on August 13, 2020.

Kim, J., & Boutin, M. (2015). Estimating the Nutrient Content of Commercial Foods from their Label Using Numerical Optimization. ICIAP Workshops.

Mafteiu-Scai, L., Mafteiu, E., & Mafteiu-Scai, R. (2018). Solving Equations Systems Using Artificial Intelligence – a Survey. International Journal of New Computer Architectures and their Applications 8(3):102-118.

Manola, Frank and Miller, Eric (2004). RDF Primer. W3C Recommendation, URL: https://www.w3.org/TR/rdf-primer/, Retrieved on August 13, 2020.

Michalewicz, Z. (1996). Genetic Algorithms + Data Structures = Evolution Programs. Springer Berlin Heidelberg.

Mitchell, M. (1996). An Introduction to Genetic Algorithms, MIT Press.

Reinivuo, H., Bell, S., & Ovaskainen, M. (2009). Harmonisation of recipe calculation procedures in European food composition databases. Journal of Food Composition and Analysis, 22(5):410-413.

Rijgersberg, H., Wigham, M., & Top, J.L. (2011). How semantics can improve engineering processes: A case of units of measure and quantities. Adv. Eng. Informatics, 25, 276-287.

Salvador, A., Drozdzal, M., Giró-i-Nieto, X., & Romero, A. (2019). Inverse Cooking: Recipe Generation From Food Images. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 10445-10454.

Schakel, S.F., Buzzard, I.M., & Gebhardt, S.E. (1997). Procedures for Estimating Nutrient Values for Food Composition Databases. Journal of Food Composition and Analysis, 10, 102-114.

Terfăloagă, I.M. (2015). Solving Systems of Equations with Techniques from Artificial Intelligence. Analele Universitatii 'Eftimie Murgu';2015, Vol. 22 Issue 1, p397.

Turing, Alan M. (1950). Computing machinery and intelligence. Mind. LIX (238): 433–460.

Vásquez-Caicedo, A.L., Bell, S., Hartmann, B. (2014). Report on collection of rules on use of recipe calculation procedures including the use of yield and retention factors for imputing nutrient values for composite foods. EuroFIR TECHNICAL REPORT.

Vlek, R.J.; Willems, D.J.M. (2019), Naar consistente voedselproductdata: hoe kunstmatige intelligentie datakwaliteit kan bewaken en bevorderen, Wageningen Food & Biobased Research, Research Report 1977 - ISBN 9789463951319

W3C OWL Working Group (2012). OWL 2 Web Ontology Language Document Overview (Second Edition). W3C Recommendation, URL: https://www.w3.org/TR/owl2-overview/, Retrieved on August 13, 2020.

Westrich, B., Altmann, M., & Potthoff, S. (1998). Minnesota's Nutrition Coordinating Center Uses Mathematical Optimization to Estimate Food Nutrient Values. Interfaces, 28(5), 86-99.

Westrich, B.J., Buzzard, I.M., Gatewood, L.C., & McGovern, P.G. (1994). Accuracy and Efficiency of Estimating Nutrient Values in Commercial Food Products Using Mathematical Optimization. Journal of Food Composition and Analysis, 7(4): 223-239.

WikiPedia (2019). URL: https://nl.wikipedia.org/wiki/Lijst_van_E-nummers, Retrieved on March 15, 2019.

Wilhelmstötter, F. (2020). Jenetics. Retrieved February 11, 2020, from http://jenetics.io/.

Wilkinson, M., Dumontier, M., Aalbersberg, I.J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J., Santos, L.O., Bourne, P.E., Bouwman, J., Brookes, A., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C., Finkers, R., González-Beltrán, A.N., Gray, A., Groth, P., Goble, C., Grethe, J., Heringa, J., Hoen, P.A., Hooft, R., Kuhn, T., Kok, R., Kok, J., Lusher, S., Martone, M.E., Mons, A., Packer, A., Persson, B., Rocca-Serra, P., Roos, M., Schaik, R.V., Sansone, S., Schultes, E., Sengstag, T., Slater, T., Strawn, G., Swertz, M., Thompson, M., Lei, J.V., Mulligen, E.V., Velterop, J., Waagmeester, A., Wittenburg, P., Wolstencroft, K., Zhao, J., & Mons, B. (2016). The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data, 3, Art#: 160018

Winkler, W. E. (1990). String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. Proceedings of the Section on Survey Research Methods. American Statistical Association: 354–359.

Annex A: Additional result plots

NEVO test case: Rijstevlaai (489)
Absolute error on recipe: 10,64 (percent point).
Fitness distance: 0,00083 (on the nutrients shown in Figure 10).

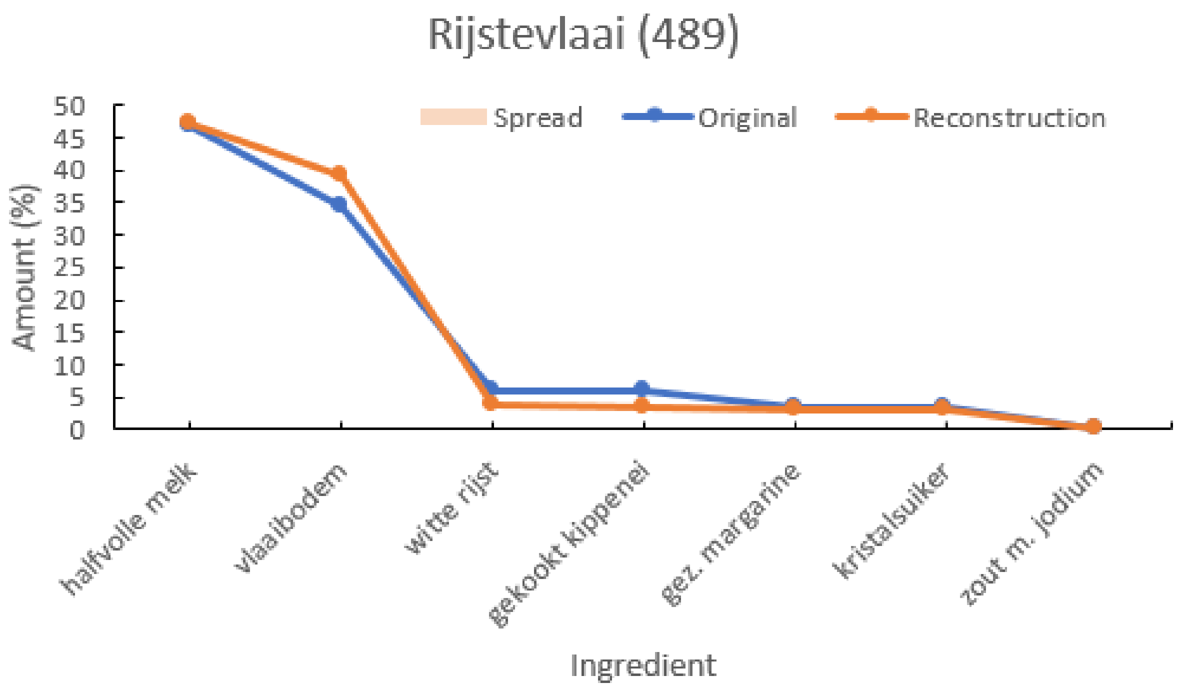


Figure 15 Reconstruction example from NEVO test set. Spread sits tightly around the reconstruction, making it hard to see.

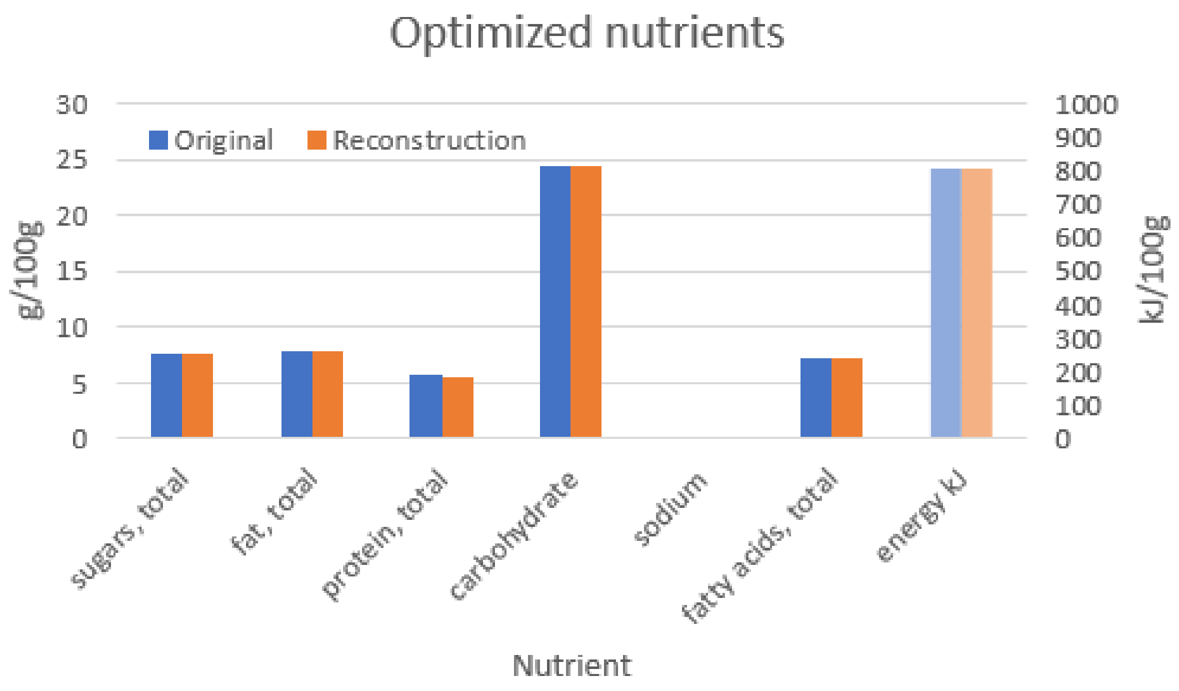


Figure 16 Nutrients optimized by the genetic engine for the above example. Energy (kJ) is plotted against the secondary vertical axis because of its scale.

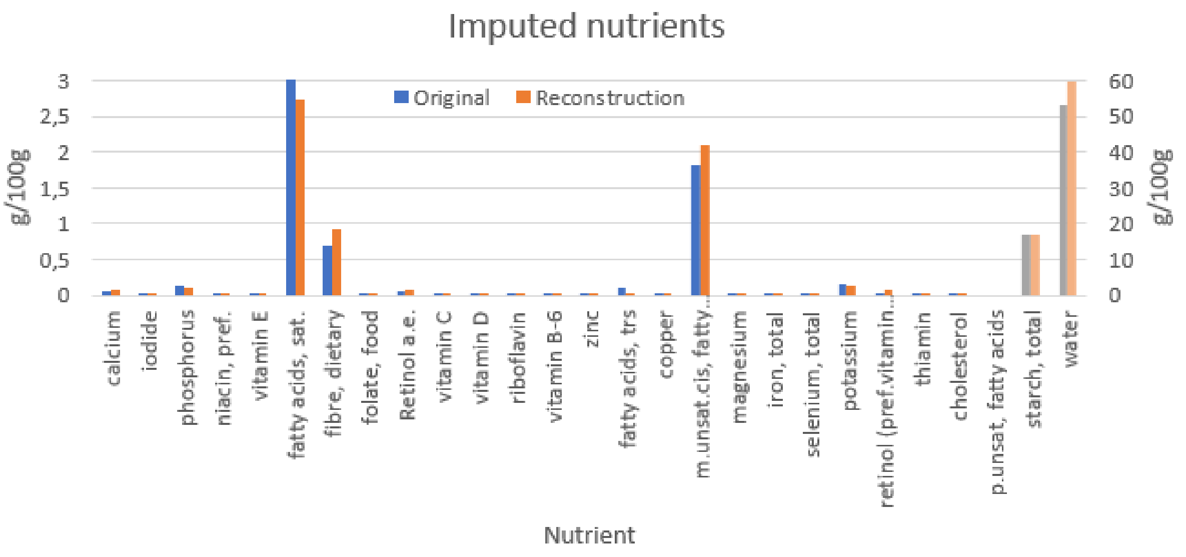


Figure 17 Nutrients imputed with the reconstructed recipe from the above example. Starch and water are plotted against the secondary vertical axis because of their magnitude.

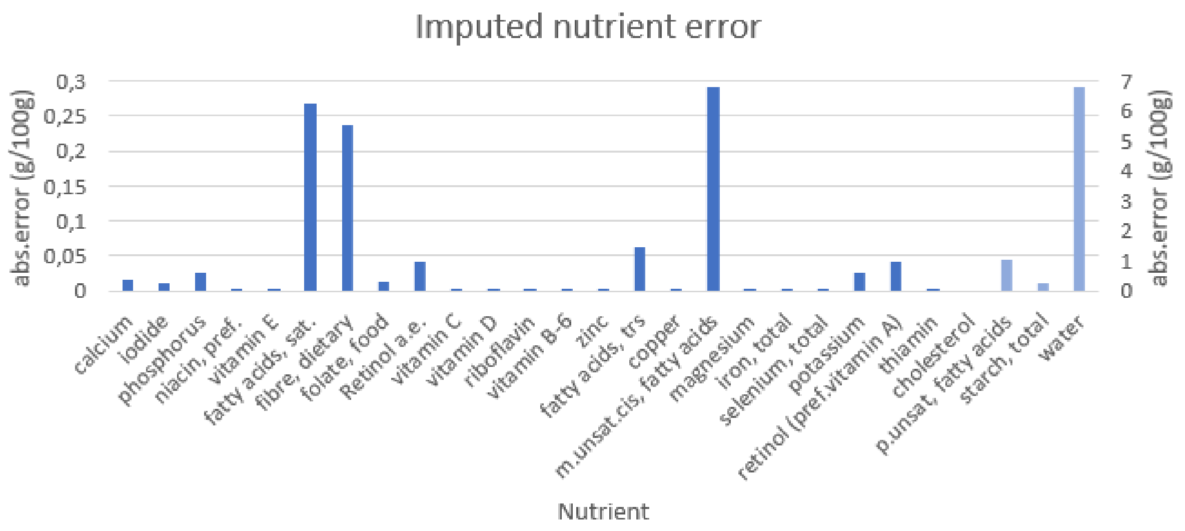


Figure 18 The absolute error between original and imputed nutrients for the above example. Polyunsaturated fatty acids, starch and water are plotted against the secondary vertical axis because of their magnitude.

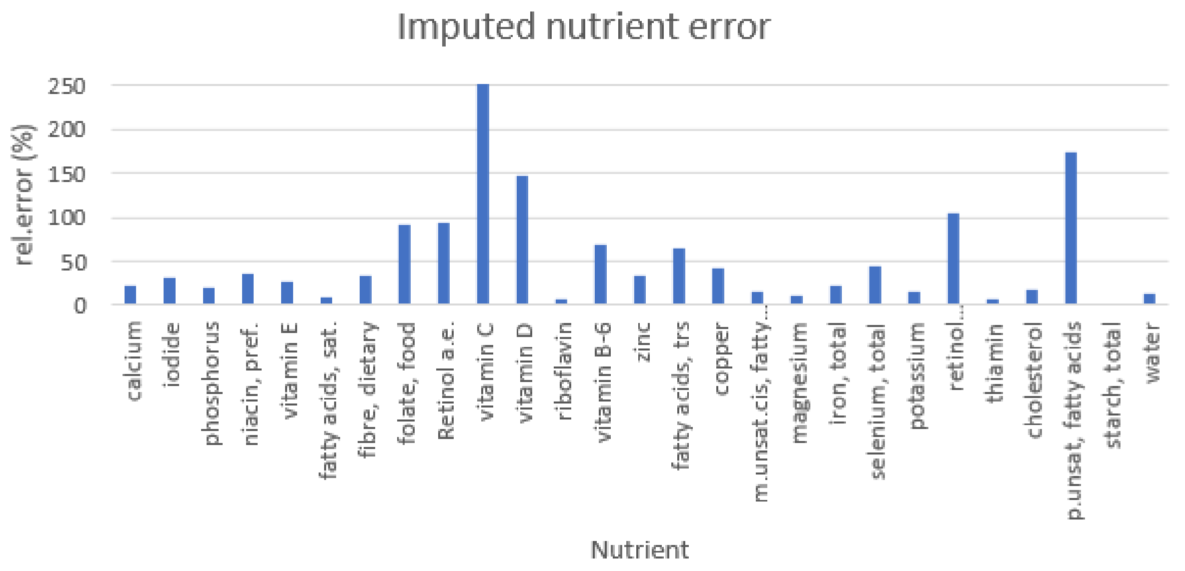


Figure 19 The error of the reconstructed nutrients as a percentage of the original nutrient amount. While vitamin C is hardly present in both original and reconstruction of this product, its relative error is quite large.

Annex B: Ingredient parsing in steps

The following steps are performed by the parser to chunk individual ingredients from a declaration:

1) Perform basis syntactic checks and clean-ups:
    a. Remove duplicate characters which are known to be incorrect (e.g. ,, and ..)
    b. Add missing spaces back in, where they unambiguously justified (e.g. after , between words)
    c. Remove parentheses around percentages (e.g. apple (12%), parentheses fulfil a different role later in the parser)
    d. Force the symbol ‘,’ as a decimal sign in numbers
2) Remove commonly occurring pieces of text, identified by regular expressions, known to be not directly related to ingredients (e.g. certification, health claims, allergen statements, statements about origin, warnings).
3) Remove trailing and leading white space
4) Any parenthesis () or {} or [] are assumed to contain so called ‘nested’ information: more detailed specifications of a specific ingredient. In order to be able to parse these, we first need to check if they are balanced (more opening than closing parenthesis of any time pose a serious problem).
5) Extract the first sentence of the ingredient declaration, identified by anything up to the first ‘.’ symbol (remember, step 1 takes care it never occurs as a decimal sign), as any additional sentences typically contain other information (not directly related to ingredients).
6) The final step is to split individual (root-level, not nested) ingredients at each occurrence of the symbol ‘,’ as long as it does not occur between numbers (as decimal sign) or within parenthesis (indicating nesting). If no ingredient terms can be chunked this way, it is possible it concerns a product for which declaring ingredients is not required (see Article 19 of EU regulation 1169/2011). In that case the description of the product itself may be interpreted as a single ingredient term in an attempt to match to known ingredients.

In the case of nested ingredients, step 6 is performed recursively after removing the parenthesis indicating a new level of nesting. The result is a so called parse-tree where sub-ingredients from the nested information are attached to a specific root-level ingredient as their parent.

After chunking individual ingredient terms from the ingredient declaration of a branded food product, the parser attempts to extract from each term any predefined amounts (percentages) if they exist. Remaining ingredient terms are subjected to another clean-up procedure (removing tabs, line breaks, special characters, punctuation, and trailing and leading spaces).

Annex C: Naive recipe estimation

An algorithm was developed for (coarse) estimation of recipes in absence of any other information than the ordered list of ingredients (e.g. no nutrients of ingredients and food), and is therefore referred to as 'naive'. It was applied for dealing with nested ingredients and for foods that did not meet the criteria to be reconstructed with the genetic engine. Just like the genetic engine's repair function (Section 2.2.4), it implements the following two knowledge rules as constraints to the recipe solution:

1) Ingredients are in decreasing order of weight, until they cover less than 2%, after which they are allowed to be listed in random order
2) The amounts of all ingredients together should add up to 100% (or 1, using the algorithms internal representation of amount)

Additionally, the algorithm utilizes a-priori knowledge of ingredient amounts (sometimes declared on the label) as additional constraint.

The native recipe estimation algorithm sequentially works through the recipe, starting with the first ingredient, and determines the logical upper and lower bounds that still allow to meet all abovementioned constraints for the recipe. After an upper and lower bound is established, the average between the two is used as the estimated amount for the respective ingredient and the algorithm proceeds to the next ingredient. Estimated values of previous ingredients will always be used in determining logical bounds of the current ingredient, as are a-priori known amounts ahead.

For foods without a-priori knowledge on certain ingredient amounts, the recipe amounts resulting from this algorithm will be solely determined by the number of ingredients and the relative position of each ingredient in the declaration. Resulting recipe estimations for such foods with up to 7 ingredients are shown in Figure 20.

As soon as a-priori amounts *are* available for some ingredients, the resulting recipe accommodates this knowledge (affecting surrounding ingredients) and becomes more food specific.

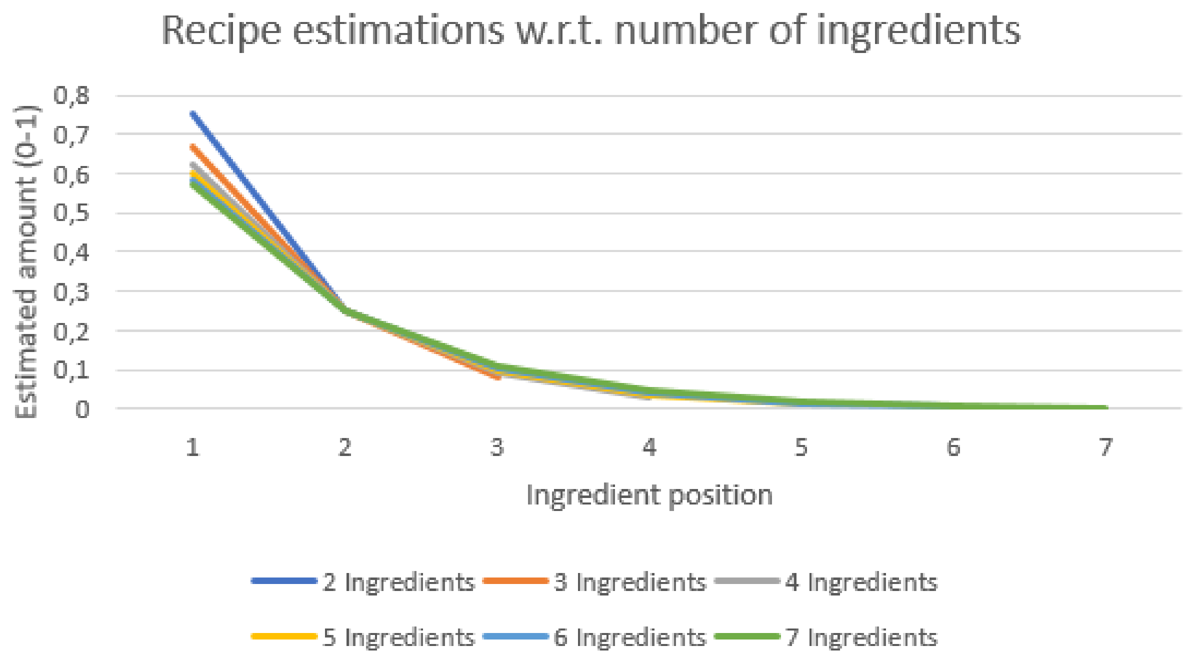


Figure 20. Estimated recipes for foods with up to 7 ingredients

To explore the potential of nature to improve the quality of life

Wageningen Food & Biobased Research
Bornse Weilanden 9
6708 WG Wageningen
The Netherlands
www.wur.eu/wfbr
E info.wfbr@wur.nl

Report 2123

Confidential