

Received June 14, 2020, accepted July 10, 2020, date of publication July 31, 2020, date of current version August 20, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3013436

Creative Culinary Recipe Generation Based on Statistical Language Models

WILLIAN ANTÔNIO DOS SANTOS¹, (Member, IEEE), JOÃO RIBEIRO BEZERRA², (Member, IEEE), LUÍS FABRÍCIO WANDERLEY GÓES^{1,2}, (Member, IEEE), AND FLÁVIA MAGALHÃES FREITAS FERREIRA¹

¹Electrical Engineering Department, Pontical Catholic University of Minas Gerais (PUC Minas), Belo Horizonte 30535-060, Brazil

²Computer Science Department, Pontical Catholic University of Minas Gerais (PUC Minas), Belo Horizonte 30535-060, Brazil

Corresponding authors: Willian Antônio dos Santos (willian.antonio.bh@gmail.com), João Ribeiro Bezerra (joaorb64@gmail.com), Luís Fabrício Wanderley Góes (lfgoes@pucminas.br), and Flávia Magalhães Freitas Ferreira (flaviomagfreitas@pucminas.br)

This work was supported in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) under Grant 001, in part by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), and in part by Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG).

ABSTRACT Many works have been done in an effort to create systems for automatic generation of creative culinary recipes. Although most of them are related to the recipe ingredient lists, few works have been done to evaluate and generate the preparation steps of culinary recipes. This work proposes the use of statistical Language Models, as well as the perplexity metric, for the generation of culinary recipes. In this work, we also developed a system for automatic generation of creative culinary recipes using two approaches: one based on a genetic programming algorithm guided by the proposed language model; and the other based on a decomposition of existing recipes and recombination of new recipes through a genetic algorithm guided by the proposed language model. This second approach achieved the best results. For this approach, a total of 6 recipes were generated to evaluate, through an online survey, the influence of the Language Model in the generation of recipes with better use of secondary ingredients, oils and seasonings, throughout the preparation steps. In the comparison between these two groups of recipes, the respondents considered the recipes generated using the language model as having the best quality, presenting an average evaluation of 63.6% of the scale (i.e. between medium and good use of oils and seasonings compared to recipes from the other group). In addition, a recipe from this approach was cooked and tasted for taste assessment, obtaining an average evaluation of 93% of the scale.

INDEX TERMS Language models, culinary recipe, computational creativity.


I. INTRODUCTION

Computational Creativity is a relatively new field of Artificial Intelligence (AI) [1], consisting of the creation of ideas or artifacts that are considered novel and useful within a given context for a group of people [2], [3]. There has been interest in applying Computational Creativity to the generation of culinary recipes. The process of creating a culinary recipe is two-fold: i) to define the steps that ingredients undergo in order to prepare a dish; and ii) the list of ingredients. However, due to its complexity, particularly in identifying the preparation steps, much of the work involving Computational Creativity explored mostly the creation of ingredient combinations [1], [4], [5].

Although the preparation steps are as important as the selection of the ingredients, there is a certain assumption that

the flavor derives mainly from the ingredients that compose a dish. In fact, the taste sensation is multi-sensory and depends on the chemical composition of the selected ingredients, besides smell and taste [6], [7]. However, some actions in the preparation instructions have physical (e.g. to smash, beat and freeze) or chemical (e.g. to ferment, clarify and smoke) transformations that may affect the taste experience of these selected ingredients. In addition, the taste sensation can be affected by effects such as: visual, tactile, thermal and sonorous (e.g. potato chips, crisp and pretzel) [8]. In turn, these effects can be also altered by the preparation actions, for example, the dish's visual with decoration actions or changes in the dish's temperature with actions like cooking, boiling or freezing.

In essence, the selection of ingredients even by humans is not solely based on taste. In addition to circumstantial criteria, such as a regional abundance of particular ingredients, they may also be chosen for the purpose of mechanical stability of

The associate editor coordinating the review of this manuscript and approving it for publication was Sabah Mohammed .

the dish (e.g. eggs) [6], which is only achieved due to actions of physical transformations (e.g. to beat). Thus the value (flavour) of a dish consists not only of the food chemistry but also of how the actions of the preparation steps modify the ingredients.

Due to the great competitiveness of the market, innovation is a necessity. And creativity is considered a factor of disruptive innovation. With regard to the importance and impact of creativity in the culinary domain, we can cite the example of *The Fat Duck*¹ restaurant, 3 Michelin-starred by the renowned English chef Heston Blumenthal, which features innovative dishes based on scientific research on the influence of sound in taste perception [8]. The interest for creative and innovative dishes in high gastronomy has existed for at least a century. In this sense, the system of this work can contribute by supporting the creativity of chefs and restaurants (helping and working together) to generate creative recipes that go beyond just the combination of ingredients.

This work proposes the use of language models in the evaluation of structural patterns in the steps of the preparation of culinary recipes, in order to help generate preparation steps with a structural consistency similar to the recipes used in the training. The main objective of the work was to develop a system for generating complete culinary recipes (with the actions of the preparation instructions) from a list of creative ingredients generated through the system provided in [5]. To achieve this main objective, two approaches were taken: (i) generation of the recipes using genetic programming and language models, named *ReProg* (Recipe Programming); (ii) generation of the recipes using a process of recipe decomposition of a training database, clustering of these recipes parts, and recombination of new recipes through genetic algorithm and language models, named *ReComp* (Recipe Composition). The *ReProg* approach is less restricted in relation to the number of combinations of possible recipes, but the *ReComp* approach is more robust in relation to the consistency of the processing of each ingredient.

For the implementation of the system it was also necessary to create a database of structured human recipes; and a NLP parser (Natural Language Processing) was also proposed to carry out the structuring of these recipes (i.e. responsible for converting the natural language text of the recipe preparation steps into a data structure that the system can consume).

This work is organized as follows: Section 2 presents the background on Computational Creativity and statistical language modeling. Section 3 presents the related work. Section 4 presents the architecture of the proposed system. Section 5 presents the details of the implementation of both the developed model and system (in two proposed approaches). Section 6 details the performed experiments. Section 7 presents the experiments' results. Section 8 presents the conclusions. Section 9 proposes future work.

¹*The Fat Duck* is considered one of the best restaurants in the world. Its website can be accessed at: <http://www.thefatduck.co.uk/>

II. BACKGROUND

Computational Creativity (CC) is a field of Artificial Intelligence (AI) that consists of replicating creative behavior in computational systems [1]. There are two approaches [9]: one focused on the creative process, in which one tries to recreate the human process that leads to creative behavior; while the other approach is focused on the creative artifact, in which one tries to generate artifacts that are considered creative by humans. The focus of this work is on the artifact. Two factors lead a group of individuals from a specific application context to judge an artifact as creative [2], [10]:

- a) Novelty, how an artifact differs from artifacts known to the individual.
- b) Value, the utility of an artifact in the application context.

Highly creative artifacts are also associated with a third factor [9]:

- c) Surprise, (which presupposes novelty) the distance between the actual artifact and the expectation from individuals of a group in the application context.

Surprise is interesting for the evaluation of artifacts because, in addition to presupposing novelty, it is used as a guide in the exploration of unknown environments in the field of Autonomous Agents in AI [11]. This makes it a great meta-heuristic to explore the space of artifacts in the search for creative artifacts, which are unknown.

A commonly used metric for surprise is the Bayesian surprise, based on the difference in the level of the *a posteriori* information (after observing the event) in relation to the *a priori* information (before the observation of the event) [12].

A. COMPUTATIONAL CREATIVITY IN CULINARY

A culinary recipe consists of a list of components (the ingredients) and a sequence of steps to be applied on these components to generate a dish. The goal of a computational creativity system in the culinary domain is to generate a recipe so that the resulting dish is considered creative by a group of individuals.

The value of the ingredients list is usually calculated based on a hypothesis called food pairing [6]. However, there is no well-defined value metric to evaluate the preparation steps.

The API used for the generation of ingredients list of the present work was based on the food pairing hypothesis. It states that ingredients that share more chemical compounds have more synergy, which translates into tastier and more aromatic combinations [6]. In [6] this hypothesis has been demonstrated as a very strong criterion in the West, mainly in Northern European and North American cuisines, although the taste sensation involves more senses than the aroma and taste themselves. Texture, consistency, antimicrobial properties [13], nutritional value, regional abundance and cultural that can also affect the choice of ingredients along with flavor [6]. However, the hypothesis of food pairing is treated as a criterion of taste preference due to the chemistry of the ingredients, with compounds related to the aroma and taste. Some authors distinguish between taste as a human sense

and flavor experience, which is multi-sensory. Ingredients do not contain the flavors in their composition; they contain chemical compounds that are associated by the brain as the taste sensation [7].

B. LANGUAGE MODEL

Language Models (LM) are models of the probability distribution of a sequence of words in a given language. Among the various types of models, the Statistical Language Model is the most generic [14]. A Statistical Language Model consists of calculating the probability of a sequence of words based on the estimate of their occurrence. The probability $P(w)$ of a sequence of T words $w = w_1, w_2, \dots, w_T$ is given by (1).

$$P(w) = \prod_{t=1}^T P(w_t | w_1, w_2, \dots, w_{t-1}) \quad (1)$$

The traditional approach to the creation of such models is the n-gram statistical model [14]. Such model consists of the truncation of the historical context of a word in n (i.e. the probability of a word occurring in a sequence is calculated only with respect to $n - 1$ predecessor words). The probability of a word occurring in a given context is calculated on the basis of the relation between the frequency of the sequences containing the n words divided by the frequency of the sequence containing the $n - 1$ words.

Others approaches use *Artificial Neural Networks* (ANN) to predict conditional probabilities of the model (i.e. prediction of the word n based on $n - 1$ predecessors) [15]–[17]. The simplest approach to an implementation of such model is through a *Multilayer Perceptron* (MLP) feedforward network, where a vector representing the sequence of the $n - 1$ predecessor words is presented as the input of the network and another vector representing the word to be predicted (nth word of the sequence) is shown as the network output. For the calculation of the nth word probability given to the $n - 1$ predecessors (presented as the input of the network), a *Softmax* is performed on the network’s output. Each word is represented by *1-to-K* vector, in which only the position representing the word in the vocabulary is set to 1, where K is the total number of words of the vocabulary. In order to represent sequences of words, other approaches can be used: the concatenation of vectors *1-to-K*; or a *Bag-of-Words* (BOW) [17], consisting of the weighted sum of the *1-to-K* vectors, where the weights are inversely proportional to the distance in time (i.e. index t) of the word from the sequence to the nth word to be predicted. Fig. 1 exemplifies a LM based on feedforward Neural Network with BOW input for n equal to 3 (i.e. predicting the third word based on the previous two).

In Fig. 1, $x(t)$ is the state of the input layer (consisting of BOW); $s(t)$ is the state of the hidden layer; and $y(t)$ is the result of the application of Softmax on the state of the output layer, so that it generates the conditional probability of the model. The vectors w_{t-1}, w_t correspond to the *1-to-K* representation of the words of the text at positions $t - 1$ and t respectively; while the vector $P(w_{t+1} | w_{t-1}, w_t)$

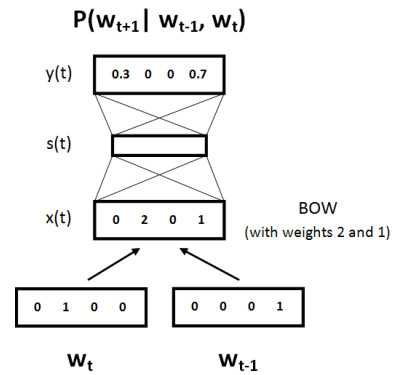


FIGURE 1. Example of LM (equivalent to a trigram) based on Neural Network Feedforward through a BOW.

corresponds to the probability vector for the word at position $t + 1$. The probability of w_{t+1} , given the previous words (i.e. w_{t-1} and w_t), is obtained by the scalar product of the *1-to-K* representation of w_{t+1} within the vector $P(w_{t+1} | w_{t-1}, w_t)$. It should be noted that the probability values on the layer $y(t)$ are only illustrative (as with other language models throughout the remainder of the text) and the smoothing promoted by softmax precludes the null values presented in this layer (this smoothing is necessary to add robustness to unseen data in training).

Perplexity is an information theory metric that measures how much a model is able to reduce the uncertainty of the symbols of an information source [18]. In the context of Language Models, *perplexity* consists of how much the LM is able to reduce the uncertainty in the prediction of the words of a text in the language for which it was trained. Roughly, *perplexity* measures the average number of viable symbols in the prediction of each word in the text, as if it reduced the problem of prediction from the original vocabulary to an equiprobable vocabulary of the size of the *perplexity* value. Thus, the smaller the *perplexity*, the better the predictive model predicts such text (i.e. the model is able to model the language of the evaluated text with high accuracy). The formula for calculating *perplexity* is given in (2) and (3) [18].

$$H_p(w) = \frac{1}{|w|} \log_2 \left(\frac{1}{P(w)} \right) \quad (2)$$

$$PP = 2^{H_p(w)} \quad (3)$$

where w is a test text in the language in which the model was trained; $|w|$, the number of words in such text; $P(w)$, the probability of the text belonging to the language modeled by the LM, which is calculated by (1) through the conditional probabilities provided by the model; $H_p(w)$, the *cross-entropy* of the text in relation to the model; and PP is the *perplexity* of the model.

III. RELATED WORK

In [1], a system was proposed for the creative generation of soups, concentrating on the list of ingredients. The generation and evaluation of recipes are based on an inspiring set

(i.e. artifacts that are known and evaluated by humans as containing quality) and on a metric of novelty based on a n-gram statistical model for evaluating rare combinations of ingredients. The recipes are generated by a genetic algorithm and evaluated through two MLP Neural Networks, which were trained for different levels of abstraction of the ingredients in the inspiring set, and a novelty metric calculated based on the inspiring set.

In [19], a system for creative generation of salads was proposed (also concentrating only on the ingredient list). However, in this case, the evaluation consisted of a discriminative classifier, which indicated the best recipe, based on the star rating of the *allrecipes.com* site, for each pair of evaluated recipes. Through this discriminative classifier, the search algorithm could exploit the space of salads and generate creative recipes.

In [5], a system was proposed to generate lists of ingredients for creative recipes. This system uses a genetic algorithm to create the lists of ingredients and evaluates the creativity of these artifacts through the RDC metric (a metric for evaluating creativity for different domains, which integrates value and novelty). The calculation of the value was made from a synergy graph based on the hypothesis of *Food Pairing* and the *Bayesian Surprise* was used to calculate novelty.

The authors in [3] proposed a system for generating complete creative recipes (i.e. including the steps of the preparation instructions). This system uses the following metrics in the evaluation of creativity: the calculation of value is based on the hypothesis of *Food Pairing* and the perception of taste according to neurogastronomy; the novelty evaluation metric is the *Bayesian Surprise*. The approach used to create the preparation instructions was based on a clustering algorithm of the total path of actions that an ingredient suffers, considering the recipes where it occurs, in which are selected the paths of the clusters with the greatest number of occurrences for each ingredient and the joining of these paths in a single execution tree.

The present work uses the system available in [5] to create the lists of ingredients. However, unlike [1], [5], [19], the steps in the preparation instructions are also generated. The *ReProg* approach, first approach of this work to the creation of recipe preparation, differs greatly from [3]. Instead of using an algorithm to decompose and recombine the recipes into the recipe database into a new execution plan, this approach uses a Genetic Programming algorithm guided by the perplexity metric on a Language Model trained with the internal structures of recipes. Secondly, in *ReComp*, other approach proposed in this work, we used the strategy of decomposing and recomposing recipes similar to [3]. However, this strategy was implemented in a genetic algorithm and uses the perplexity metric on a Language Model to evaluate the mixtures of the ingredients along the recipe actions, in order to guide the recomposition of the recipes for better mixtures (e.g. with the proper use of seasoning, and oils in frying).

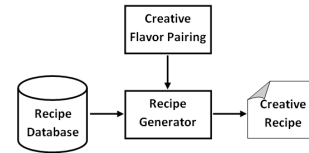


FIGURE 2. Simplified diagram of the system architecture.

IV. SYSTEM ARCHITECTURE

The generic architecture of the system for generating creative culinary recipes proposed in this work is presented in Fig. 2.

The *Recipe Database* is a repository of recipes in a structured format for training the system models. The *Creative Flavor Pairing* component is the API for generating creative ingredient lists available in [5], in which lists of ingredients are generated based on the hypothesis of *Food Pairing* and the RDC metric (*Regent-Dependent Creativity*) [10]. The RDC is a domain independent metric that assesses the creativity of artifacts. This metric requires that artifacts are described within the Regent-Dependent Model, in which artifacts features are represented as a set of pairs between its features and their modifiers. This dependency relationship is defined by a pair $P(\text{regent}; \text{dependent})$ associated with a numeric value v . A regent is a feature that contributes to describe an artifact, and may be an action or attribute, while a dependent can change the state of an attribute or connect an action to a target.

The *Recipe Generator* is the component responsible for generating the sequence of actions of the preparation from the list of ingredients and also the composition of these parts into a complete recipe. The *Recipe Generator* presents two approaches that are described in detail in Section V.

A. RECIPE DATABASE

A recipe is composed of a list of ingredients and a sequence of actions to be applied to the ingredients. In recipes created by humans, the list of ingredients is usually well-structured, but the steps of the actions is arranged in a natural-language text without structuring (i.e. the actions are arranged throughout the text along with other words). Thus, for a computational system, it is easier to process the list of ingredients than the actions of the preparation instructions.

Processing the preparation instructions requires a pre-processing step, which consists of structuring the preparation instruction actions that are arranged in the natural language text of the recipe. This requires a Natural Language Processing (NLP) effort, culinary knowledge (through ontologies of ingredients, tools, and actions) and inference rules (to evaluate the disposition of actions temporarily and to use ontologies to remove ambiguities, treat implicit references, etc.). Through this preprocessing, the system can find the dependency relationships between actions (i.e. sequences of actions, where one action is performed on the result of the other) and also between actions and ingredients to finally be able to process the instructions of the preparation.

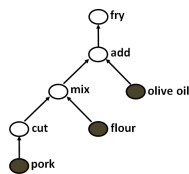


FIGURE 3. Example of recipe represented as an acyclic graph.

These dependencies of the preparation instructions form an acyclic graph [3] as that of Fig. 3.

In Fig. 3, the edges symbolize the relation of dependence (the direction of the edge indicates the sequence of actions - from past actions to future actions); and the nodes represent actions and ingredients. In Fig. 3, the nodes of ingredients were highlighted in darker color to demonstrate that the ingredients always occur at the ends of the graph.

The next subsections present the recipe databases used in the context of this work.

1) WIKITAAABLE

Wikitaable² is a database of recipes created on the semantic wiki platform to enable reasoning about culinary knowledge [20]. The recipes in this database have three main components: list of ingredients; text of the actions of the preparation instructions; and a formal description of the actions of the preparation instructions. An example of such a formal description is shown in Fig. 4. This formal description resembles a graph list representation that shows the state name and information on which other states it has an edge. Thus, this base does not require the preprocessing of the text of the preparation instructions, since it already has a representation of the preparation instructions as an acyclic graph (in the form of a list of edges). All the recipes available in the Wikitaable dump, containing 1439 recipes, was used.

```

cook : squash_0(squash) => squash_1(squash)
mash : squash_1(squash) => squash_2(squash)
add : milk_0(milk) squash_2(squash) => mixture_0(milk,squash)
beat : egg_0(egg) => egg_1(egg)
place : egg_1(egg) => egg_2(egg)
bake : egg_2(egg) marshmallow_0(marshmallow) =>
      mixture_1(egg,marshmallow)
    
```

FIGURE 4. Formal preparation of the "Squash fluff" recipe.

The database of Wikitaable also has an ontology of actions and ingredients. The ontologies of actions and ingredients used in the present work were based on these two ontologies of Wikitaable. The following modifications were made: in the ontology of actions two levels of more generic classes were used; on the ingredient ontology was used the most specific level (e.g. "Bacon" was assigned to the class "Pork" instead of just "Meat"). The complete ontology of

²Wikitaable is a semantic Wiki in the field of cooking. A cached version can be accessed at: https://web.archive.org/web/20120615184112/http://wikitaable.loria.fr/index.php/Main_Page

class	subclass	action	class	subclass	action	
to change physically	to cream	cream, reduce	to garnish	-	garnish, trim	
	to soften	soften		to cover for	brush, decorate, glaze, powder	
	to strain	strain		to garnish by	dizzle, top	
	to tender	tenderize	to move	to bring	bring	
	to change temperature	to decrease temperature		chill, cool, freeze, frost, refreeze	to place	place, store, transfer, unsmold
to increase temperature		bake, boil, broil, brown, cook, fry, grill, heat, melt, microwave, reheat, roast, saute, simmer, steam, thaw, toast, warm, wilt		to shake	shake	
to form	-	reconstitute		to turn	flip, turn	
	-	form	to prepare	-	work	
	to dice	dice, layer, shred, slice		to add	add, deglaze, grease, salt, season, spray	
	to flatten	flatten		to cover	coat, cover, dust, wrap	
	to fold	fold		to dissolve	dissolve	
	to prick	prick		to pour	arrange, fill, ladle, lay, pour, punch, put, seal, shape, spoon, sprinkle	
	to roll	roll		to process	process	
	to spread	spread		to shake	shake	
	to thicken	thicken		to mix	beat, blend, combine, mix, scramble, stir, whip, whisk	
	to split	-		split	to soak	dip, drop, marinate, marinate, soak
		to break		break, crumble	to pat	pat
		to clean	prune, rinse, wash	to reserve	-	reserve
		to cut	chop, cut, mince, separate		to smoke	smoke
to grate		grate	to gather	to process	process	
to halve		divide, halve		to shake	shake	
to press		crush, grind, ground, juice, knead, mash, press		to mix	beat, blend, combine, mix, scramble, stir, whip, whisk	
to refine		drain, sift		to soak	dip, drop, marinate, marinate, soak	
to remove		debone, discard, dry, peel, rub, scrape, skim, skin, uncover		to pat	pat	
to reserve		reserve		to reserve	reserve	
to smoke	smoke	to smoke		smoke		

FIGURE 5. Actions ontology.

actions used in this work is presented in Fig. 5, where the merged cells have the name of the classes, each with their set of subclasses and the actions (which actually appear in the recipe preparation) on the right.

B. WIKIA

As an alternative source of recipes, the online recipe database Wikia³ has also been used (500 recipes of the category of American recipes). Due to the fact that this database does not contain the formal description of the preparation instructions, it was necessary to create an NLP parser, as mentioned in Section IV-A. Such a parser is explained in detail in the next subsection.

1) NLP PARSER

The architecture of the NLP parser is shown in Fig. 6.

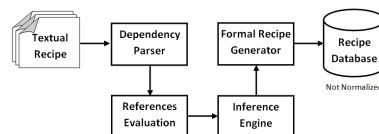


FIGURE 6. Parser NLP architecture.

The *Dependency Parser* consists of a parser that generates dependency tags (e.g. direct object, subject, etc.) and Part of Speech tags (PoS tags - some examples of these tags are nouns, verbs, and so on) of words in natural language phrases. In this work, we used the parser dependencies called Parsey McParseface created from the framework of Google SyntaxNet. An example output of the dependency parser is shown in the text below:

³Wikia is an online repository of textual culinary recipes. It can be accessed at: https://recipes.fandom.com/wiki/Category:American_Recipes

```
Input: cook onion in oil in large skillet.
Parse:
cook VB~ROOT
  +-- onion NN dobj
  +-- in IN prep
  |   +-- oil NN pobj
  +-- in IN prep
  |   +-- skillet NN pobj
  |       +-- large JJ amod
  +-- . . punct
```

In the output of this dependency parser we have: the sentence evaluated in the first line (preceded by “Input:”); the result of the dependency parser after the second line (below “Parse:”). At the end of each result line, there are 3 components: one word; the PoS Tag; and the dependency tag. The indentation of the result represents the hierarchy between the words (the more left and higher, the higher in the hierarchy).

TABLE 1. Tags for action detection.

Dependency Tag	PoS Tag
“ROOT” (root of the sentence)	-
“partmod” (participial modifier)	-
“xcomp” (open clausal complement)	-
“dep” (generic dependent)	-
“parataxis”	“VB”, “VBP”, “VBZ”, “VBD”, “VBG”, “VBN” e “MD” (verb conjugation and tense)

TABLE 2. Tags for ingredients and tools detection.

Dependency Tag	PoS Tag
“dobj” (direct object)	-
“iobj” (indirect object)	-
“pobj” (object of preposition)	-
“conj” (conjunction, in the branch of some ingredient/tool)	-

The *References Evaluation* step consists of analyzing some of the tags generated by the *Dependency Parser* in order to detect references to actions (ontology actions), ingredients (from the list of recipe ingredients), and tools. Table 1 shows the list of tags for action detection. The ingredients and tools are detected, on the branch of each action, from the tags in Table 2 (the distinction between ingredients or tools is made by comparison of the detected words). In both tables, the dependency tag is the criteria for detection, while the PoS tag serves only as a filter (only words with those PoS tags will be detected). The hyphen symbol (minus) denotes an absence of a filter (i.e. any PoS tag is accepted in this case). An example of such detections is shown below:

```
Input: layer lettuce , bacon , turkey and
      tomato on each tortilla.
Parse:
layer NN ROOT
  +-- lettuce NN dobj
  |   +-- , , punct
  |   +-- bacon NN conj
  |   +-- , , punct
  |   +-- turkey NN conj
  |   +-- and CC cc
  |   +-- tomato NN conj
  +-- on IN prep
```

```
|   +-- tortilla NN pobj
|       +-- each DT det
+-- . . punct
Detections:
  Action      = ``layer``
  Ingredients = [ ``lettuce``, ``bacon``,
                  ``turkey``, ``tomato``,
                  ``tortilla`` ]
```

The Inference Engine step performs inferences based on culinary knowledge for the construction of a timeline containing the sequences of actions and their ingredients. This timeline contains initially (denoted as -1 time) only ingredients in the ingredient list (including the appropriate units and quantities available). For each action detected by the Reference Evaluation step, a time record (starting from 0) is created with the action and ingredients (again with their units and quantities available) that undergo such action. These ingredients along the timeline represent the pathway of each ingredient in mixtures (i.e. not always represent the isolated ingredient). Each action generates a new time in the timeline. And for each ingredient name (word detected in the Reference Evaluation step) a filtering is performed on all ingredient records present in the timeline (throughout all previous times) to select the ingredient record for the action of the current time. Such filtering is performed with the following criteria:

- a) Similarity in the name.
- b) If there are verbs in the tree branch of the ingredient, the similarity of the name of these verbs with the names of the actions of the timeline.
- c) If there are quantities in the tree branch of the ingredient, the compatibility of these values with the unit and quantity of the ingredients in the timeline (removal of insufficient quantity records).
- d) If there is more than one record of the ingredient in the timeline, select only the one associated with the longest time.

After selecting the ingredient from the timeline, a copy of this record is made for the current action time by keeping a reference to the original record and transferring the original record amount to the copy (partial, if there is a quantity or total reference in otherwise). If the original record belongs to a mixture (i.e. the action of that record is being run on more ingredients), all other ingredient records of that time are also copied (in the same way as the selected one). If no ingredient in the timeline is selected, it is evaluated whether the ingredient name is listed as an ingredient class name (eg “all ingredients”, “vegetables”, etc.), therefore all ingredients belonging to this class are added (according to the ontology of ingredients) and containing quantity available.

After an action is finished processing (i.e. selecting and creating the ingredients in the timeline), the following analysis should be performed:

- a) If the action is an “add”, for a time greater than 0 and contain all the ingredients like “dobj” and “iobj” or all the ingredients as “pobj” on a single preposition, then all records of the ingredients from the previous time are ingredients of this action.

- b) If the action is different from an “add” and does not contain any ingredient record in the timeline, then all ingredient records from the previous time are treated as an ingredient of this action.

TABLE 3. Example of timeline built up to time 2 (third action).

Time	-1	0	1	2
Action	-	“cut”	“fry”	“slice”
Record	(0, “potato”, ∅) (1.0 kg, “pork”, ∅) (0, “orange”, ∅)	(0, “potato”, -1)	(2, “potato”, 0)	- - (1, “orange”, -1)

In Table 3, it is presented a small example of a timeline created by the stage of the inferences engine for the recipe below, where each timeline record corresponds to an ingredient tuple containing: ingredient quantity (and unit) available; ingredient name; time of origin of the ingredient in the timeline (i.e. from where that ingredient was copied) or the symbol ∅ (indicating the ingredient list).

Ingredients:

- 2 potato
- 1 kg pork
- 1 orange

Directions:

- Cut the potatoes.
- Fry.
- Slice the orange.
- ...

Finally, the Formal Recipe Generation step generates recipe from the timeline (i.e. the ingredient list and the preparation instructions graph as a list of edges). Each time from 0 on the timeline will generate a node with the name of the action related to that time. In turn, each ingredient record in the timeline will form an edge between the action node of that time and the time action of the source record (from where that record was copied). If the ingredient record has been copied from the ingredient list (time -1), then an ingredient node is created with the ingredient name as the input of that edge.

C. NORMALIZATION OF THE RECIPES

In order to remove redundancies in the recipes, the addition, substitution or removal of states from its graph in the preparation instructions was performed. These transformations were based on [3], aimed at promoting more efficient processing of recipes by the rest of the system. They were applied on all recipes, regardless of the database (Wikia or Wikitaable). The following rules were established for the structure of the graph of preparation:

- a) It should contain only actions present in the ontology of actions (i.e. remove missing actions in the ontology).
- b) Actions should represent relevant transformations of the ingredients (i.e. remove actions on tools or move ingredients).
- c) There should be no synonymous actions (i.e. to replace synonymous actions).

- d) Every recipe must contain a “serve_recipe” action by joining all the related components of the graph by the last action (in time) of each component (i.e. a single output action).

The removals required to satisfy rules a and b involve passing all input nodes (from the node to be removed) to each of their output nodes (those having the node removed as an input). The actions that were removed because of rule b were the following: actions of the classes “to_reconstitute”, “to_move” (except the “to_shake” subclass), “to_pour”, “to_clean” and “to_prepare”; and also the actions “reserve”, “fold”, “form”, “drop”, “discard” and “preheat” (does not belong to ontology of actions, but also occurs frequently in recipes).

In order for the recipes to comply with rule c, the substitutions shown in Table 4 were made.

TABLE 4. Replacement of synonyms in the normalization of recipes.

Substitute Action	Original Action
“add”	“attach” “enclose”
“cut”	“chop”
“divide”	“halve”
“freeze”	“frost” “refreeze”
“fry”	“refry”
“garnish”	“decorate”
“heat”	“reheat” “warm”
“mash”	“crush”
“mix”	“blend” “combine” “stir”
“toss”	“shake”

In the case of the ReComp approach to generate the preparation instructions (i.e. the approach based on decomposition and recomposition of recipes), the following rule was also observed: all actions should have arity to one (i.e. on a single input), with the exception of “add” and “mix”, which can have more than one input (i.e. thus, insert an “add” on the inputs of any action that does not respect this rule). This additional rule makes it possible to control the arity of the actions in a more rigid way (necessary in the ReComp approach), but it promotes a lot of modification in the internal structure of the recipe, by the addition of nodes, and for this reason, it was only used in this approach.

Fig. 7 shows how the recipe “Squash fluff”, presented in Fig. 4, would look after its normalization.

After the normalization stage, the recipe database is ready to be attached to the system.

V. RECIPE GENERATION

Once the database of recipes is created and a list of ingredients is requested from the Creative Flavor Pairing API, we can generate the graph of the actions of the preparation instructions. Two approaches (i.e. two systems) have been developed for this task.

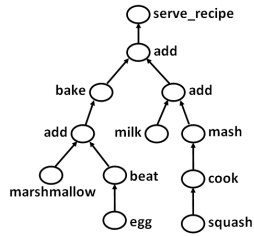


FIGURE 7. Normalized "Squash fluff" recipe.

- a) *ReProg* Approach, consisted of the generation of the recipes using genetic programming and language models.
- b) *ReComp* Approach, uses a process of decomposition of the database recipes, clustering of these recipe parts and recombination of new recipes through genetic algorithm and language models (on mixtures).

The following sections describe the two approaches.

A. ReProg APPROACH

The system architecture for this approach is shown in Fig. 8.

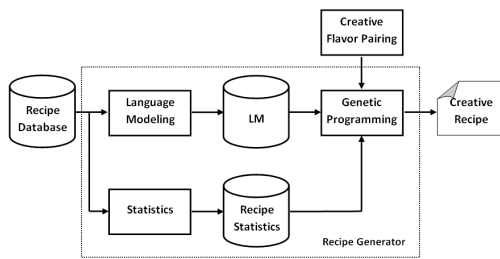


FIGURE 8. System architecture for ReProg Approach.

The *Language Modeling* is the component responsible for creating language models (LM repository) from the *Recipe Database*, while the *Statistics* component is responsible for extracting recipe statistics and storing it in the *Recipe Statistics* repository. These statistics are related to the repetition of ingredients and actions of the preparation instructions; as well as the arity statistics (i.e. the number of ingredients participating in each type of preparation action). The *Genetic Programming* component consists of the genetic programming algorithm, responsible for generating the recipe preparation from the list of ingredients, language models and recipe statistics.

1) LANGUAGE MODELING FOR RECIPE EVALUATION: ACTIONS ON ITS INPUTS

Just like the order of words in a text, the order and sequence of steps are important in a recipe. The steps of the preparation can promote physical or chemical transformations, such that some of these actions only make sense to be performed on ingredients in a given state. However, unlike texts, a culinary recipe does not contain information arranged only sequentially. The steps of a preparation form a tree so that certain

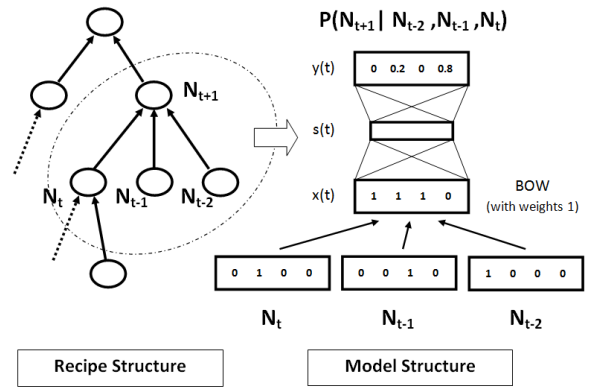


FIGURE 9. Architecture of the proposed Language Model being applied to the structure of a culinary recipe.

branches can occur in parallel. In order to evaluate this parallel information, a Language Model applied to culinary recipes must evaluate information between consecutive steps and parallel steps of the same branch. The architecture proposed in this work is shown in Fig. 9, in which for each action of the preparation instructions to be predicted (denoted by N_{t+1} , which could be predicted, for example, as "saute" with probability 0.2 or "fry" with probability 0.8), the predecessor actions are evaluated at the level below the branch (i.e. all the "children" ingredients and actions of the action to be predicted - denoted in Fig. 9 by N_t to N_{t-2} , which could be, for example, "cut", "onion" and "oil"), as if the n th word of the Language Model was the action to be predicted and all the "children" of that branch were the predecessor sequence.

Once this model is trained on the database recipes, the model will absorb and generalize certain syntactic patterns present in the structure of culinary recipes (i.e. a kind of language of the recipe execution process). Then it is possible to use the *perplexity* metric as a selection criterion for recipes that present these patterns. The more the recipes present the syntactic patterns modeled, the lower the *perplexity* of the model, compared to the perplexity of the recipes.

It should be noted that Recurrent Neural Networks (RNN) are the state of the art for language models. The RNNs have the greatest capacity to evaluate long sequences of temporally arranged words (in relation to other networks), but in our adaptation of language models for evaluating the recipe preparation graph, there is no evaluation of very long temporal sequences (basically just one time step - from a node to its input). Thus, the use of RNN-based language models in this context is not justified.

2) GENETIC PROGRAMMING FOR RECIPE GENERATION

The ingredients previously defined for the ingredient list of the new recipe are used as a set of variables in the GP algorithm (i.e. it defines the ingredients available for the evolutionary process restricted to those generated by the *Creative Flavor Pairing* API). The function set consists of all the actions that occur on a recipe database (including the root node of the recipe, which unifies the final parts of the recipe).

The GP algorithm used was the DEAP (*Distributed Evolutionary Algorithms in Python*) framework.⁴ Such algorithm was implemented for multiobjective optimization: to minimize the *perplexity* of the model in relation to the individuals (i.e. to obtain better *perplexity* values); and maximize the depth of a recipe (otherwise, minimizing *perplexity* would always lead to very small recipes). On the objective of *perplexity* two constraints were imposed. In the event of such restrictions, *perplexity* is penalized: (i) if the individual presents repetitions of some node above the maximum or below the minimum repetitions found for such node in the database; and (ii) if any of the ingredients in the initial list are not used in the individual. The penalty function is showed in (4).

$$\begin{aligned} igd &= \frac{(|I| + 1)}{(|L| + 1)} \\ PP^* &= PP \left(\frac{(rpt + 1)}{igd} \right)^2 \end{aligned} \quad (4)$$

where PP is the original *perplexity*; PP^* , the penalized *perplexity*; rpt , an indicator function of out-of-range repetitions (value equals to 1 if there is a repetition above the maximum or below the minimum repetition recorded in the recipe database and it equals to 0 if otherwise); I , the list of ingredients used in the recipe; and L , the list of ingredients in the list provided by the *Creative Flavor Pairing* (i.e. available ingredients).

The fitness function of the individuals is presented in (5). The objective space used for the fitness calculation consisted of the Cartesian plane formed from the penalized *perplexity*, PP^* , and the depth of the preparation graph of the individuals.

$$fitness = \frac{1}{v} + \frac{1}{f} \quad (5)$$

where v is the number of neighbors of the individual in the objective space (*perplexity* and depth), which is calculated through a grid (where neighbors are individuals belonging to the same cell, including the current individual); and f is the number of the Pareto front to which the individual belongs (the lowest possible value is 1).

should be noted that the variable v is one of the factors in the objective search space (i.e. it is not an objective in itself). While the variable f is the composition of the two objectives (i.e. the penalized *perplexity* and the depth of the recipe), but a composition that respects the pareto optimality. That is, f always presents the same value for each set of solutions, which forms a frontier (called a pareto frontier), from which a solution cannot be chosen as an optimal solution of the problem in relation to the others of this same set without prioritize a specific objective. Pareto frontiers are based on the concept of dominance. Given two distinct points (solutions) in the objective space, it is said that one point dominates the other if for all objectives of that point

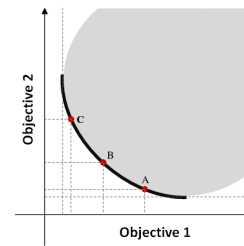


FIGURE 10. Example of objective space to minimize two conflicting objectives.

have equal or better values optimized in relation to the other point (i.e. the point that dominates can be said better than the another point, even without prioritizing any objective). The first pareto frontier of a population of possible solutions (individuals of the evolutionary algorithm) is the set of solutions that are not dominated by any other in this entire population. That is, for at least some objective, each of these frontier solutions presents a more optimized value compared to the other solutions (including in relation to other solutions on the frontier itself). For example, in Fig. 10, points A , B and C belong to the first pareto frontier that minimizes the 2 objectives illustrated in the image. Point B is better than point A in the first objective and better than point C in the second objective; while A is better than B and C in the second objective; and C is better than A and B in the first objective. Thus, none of the three points can be considered a better solution than the other two without prioritizing any specific objective. Removing all points from the first pareto frontier, it is possible to create, by the same principle, a second pareto frontier with the remaining points that are not dominated by any other of this new population. Therefore, although the fitness function is just a single equation, due to this multi-objective nature of the variable f , this single function will be able to optimize two objective functions (in this case, penalized *perplexity* and depth of the recipe).

The selection operation used for the crossover was the tournament of size 3. And the crossover used in the genetic programming algorithm of the present work is based on the following steps: after the two parent individuals have been chosen, a copy of both is made to represent the genetic material available for the generation of the children. Based on this copy of the parents, a selection node is randomly selected for each of these two trees. The only restriction for choosing these two cutting nodes is that the root of the tree must be disregarded (i.e. it cannot be chosen). From these two chosen cutting nodes, the entire branch of the trees is cut. Then the exchange of these cut branches is made between the two copies of the parents' trees (i.e. the branch removed from the first tree is connected in the place where the chosen cutting node of the second tree was and vice versa). If the resulting trees have a depth above the pre-established maximum limit for recipes, then the children are discarded and the process is repeated in a similar way. However, at each unsuccessful iteration, the new cut point of the largest branch (in depth)

⁴DEAP is a framework for implementing evolutionary algorithms. It can be accessed at: <https://github.com/DEAP/deap>

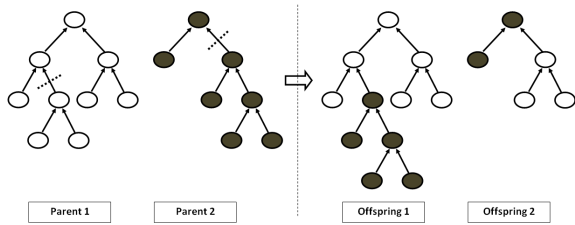


FIGURE 11. Example of recombination operation for genetic programming.

of the previous iteration is exchanged for a cut node closest to the leaf on that same branch of the previous iteration (i.e. the branch is exchanged for a sub-branch). The process is repeated until cutting nodes are chosen that are capable of generating two children with valid depth. An example of this crossover operation is shown in Fig. 11.

The mutation does not occur individually. It was only applied to the whole set of genes of the population of the generated offspring. And the mutation operation consisted of changing the content of the nodes at random (i.e. a portion of the nodes in the entire population undergone on an exchange of the preparation action or ingredient with which it was associated). The only three restrictions for this operation were the following ones: i) the “serve_recipe” action can only appear at the root of the tree and cannot be changed; ii) preparation action nodes cannot be exchanged for ingredient nodes and vice versa; iii) and the space of possible ingredients for mutation is restricted to the ingredients previously chosen by the step of generating the list of ingredients.

At each iteration of the algorithm, crossovers are performed in a quantity necessary to double the size of the population and then half of the total population that appear in the smaller nondominated fronts is chosen, eliminating the rest. The first nondominated front of the final population is always compared to the *Hall-of-Fame* (containing individuals from previous iterations that have never been dominated by any other), in order to update such a set. And the final answer of the algorithm is the *Hall-of-Fame* of the last iteration.

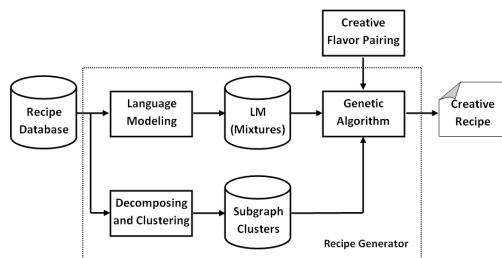


FIGURE 12. System architecture for ReComp Approach.

B. ReComp APPROACH

The system architecture for this approach is shown in Fig. 12. *Language Modeling* is the component responsible for creating the language models for mixtures (LM repository)

from the recipe database. The *Decomposing and Clustering* is the component responsible for the decomposition of the recipe preparation (subgraph decomposition) and clustering of its parts (forming a repository of subgraph clusters). The *Genetic Algorithm* component consists of the genetic algorithm responsible for recomposing the graph of preparation of the new recipe from the subgraph clusters (restricted to the list of ingredients provided by the *Creative Flavor Pairing* and using language models to evaluate the ingredient mixtures).

1) LANGUAGE MODELING FOR RECIPE EVALUATION: ACTIONS ON MIXTURES

A second approach that we explore in this work is to evaluate in the recipes the mixtures of ingredients that occur along the steps. Several actions occur on a single ingredient or a single mixture (e.g. “peel an apple”). Such actions do not promote the formation of a new mixture (only the processing of an existing mixture or ingredient). On the other hand, there are actions that promote the creation of new ingredient mixtures (e.g. “fry an egg in oil”, “cook a chicken with garlic”, etc). As recipes are normalized according to Section IV-C, these actions that promote new mixtures always contain an “add” or “mix” in their inputs. The proposed model for this approach is similar to that of *ReProg* Approach (Section V-A1). However, this model presents the following differences: actions to be predicted by the model (in the layer $y(t)$) are only those that promote new mixtures (ignoring all other actions); and the BOW of the layer $x(t)$ should be composed only of the ingredients of the mixture undergoing the predicted action (i.e. any ingredients occurring within the branch of action to be predicted - ignoring any intermediate processing that the ingredient undergoes). Fig. 13 shows an example of applying this model to a recipe, where N_{t+1} is the action that promotes a mixture (action to be predicted by the model - for example, “fry” with probability of 0.3 and “cook” with probability 0.7); I_0, I_1, I_2 are ingredients achieved in the branch and form the mixture promoted by the action N_{t+1} (e.g. “oil”, “garlic” and “pork”).

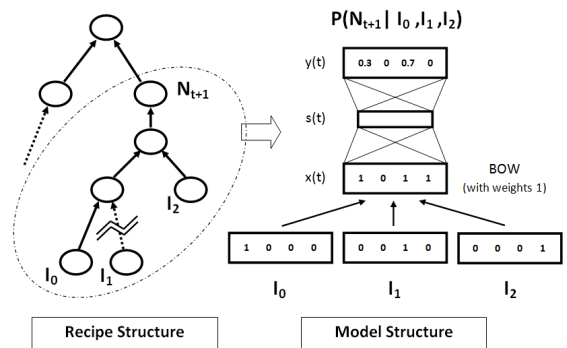


FIGURE 13. Architecture of the proposed Language Model for mixtures being applied to the structure of a culinary recipe.

This model was also trained on the recipe database, but the goal, in this case, was to absorb the ingredients mixing patterns throughout the recipe. In this case, the perplexity

metric indicates how well adjusted the structure is in terms of the formation of ingredient mixtures compared to these mixture patterns in the recipe database. The more the evaluated recipe contains these mixture formation patterns, the lower the perplexity value of the model compared to the evaluated recipe.

2) DECOMPOSITION AND RECOMPOSITION OF RECIPES

This recipe decomposition and recomposition approach is based on [3]. The main difference is that the algorithm of recomposition of the recipes was implemented on a genetic algorithm and also counted on the use of language model.

In the process of recipes decomposition, they should be broken down into pieces of recipes, which are basically paths on the graph of the preparation that begin in the last action of the graph (the root - which in this case is always "serve_recipe") and ends in an ingredient (a leaf in the graph). In this way, there will be a path to the sequence of actions each ingredient undergoes throughout the recipe. Successive repetitions of the same action along that path can be removed. An example of this decomposition for the recipe of Fig. 7 is shown below:

```
Decomposition of the "Squash fluff" recipe:
[['serve_recipe', 'add', 'mash', 'cook', 'squash']]
[['serve_recipe', 'add', 'milk']]
[['serve_recipe', 'add', 'bake', 'add', 'beat', 'egg']]
[['serve_recipe', 'add', 'bake', 'add', 'marshmallow']]
```

Once all the database recipes have been decomposed into vectors, these vectors are separated into sets according to the ingredients at the end of each vector. Thus, there will be for each ingredient a set of all the sequences of actions that occur on it in the recipe database. In addition, sets are also created for classes of ingredients, for which a copy of each ingredient vector belonging to the class is made by replacing the ingredient name with the class name. These class sets present the sequence of steps that ingredients of the same class usually suffer (e.g. action sequences for "meats").

On each of these sets, a clustering is performed through the K-Medoids algorithm [21]. The K-Medoids algorithm works in a way equivalent to K-Means, but instead of calculating a centroid for each cluster, this algorithm uses the most centralized data point (with less distance to the other points) of the cluster to represent it, since we can not calculate a mean coordinate of a vector of symbols [21]. The distance metric used to cluster these sets of recipe parts is the Levenshtein distance metric, which consists of the least amount of addition, subtraction, or substitution required to make two sequences equal [22].

Each cluster in a set represents a sequence of actions on an ingredient (or class of ingredients) that is repeated in several recipes with small variations. The larger the cluster size the more frequent the sequence of actions. In this way, the importance (i.e. the weight) of each cluster, and consequently its sequence of actions, is measured by its size.

After clustering, each sequence vector present in the set is associated with a tuple containing its numerical identifier (the position of this vector in the set); the weight of its cluster

TABLE 5. Example of cluster and triples for ingredient and its class.

Vector	Cluster	Triple (id, weight, class)
['serve_recipe', 'fry', 'potato']	0	(0, 3, False)
['serve_recipe', 'mix', 'fry', 'potato']	0	(1, 3, False)
['serve_recipe', 'add', 'fry', 'add', 'potato']	0	(2, 3, False)
['serve_recipe', 'mash', 'bake', 'potato']	1	(3, 2, False)
['serve_recipe', 'mash', 'slice', 'bake', 'potato']	1	(4, 2, False)
['serve_recipe', 'fry', 'tuber']	0	(0, 3, True)
['serve_recipe', 'mix', 'fry', 'tuber']	0	(1, 3, True)
['serve_recipe', 'add', 'fry', 'add', 'tuber']	0	(2, 3, True)
['serve_recipe', 'mash', 'bake', 'tuber']	1	(3, 4, True)
['serve_recipe', 'cool', 'slice', 'bake', 'tuber']	1	(4, 4, True)
['serve_recipe', 'mash', 'slice', 'bake', 'tuber']	1	(5, 4, True)
['serve_recipe', 'mash', 'cut', 'bake', 'tuber']	1	(6, 4, True)

(the size of the cluster); and a boolean class indicator of the set type, with false for ingredient sets and true for class sets. With this triple assignment (id, weight and class indicator), the database recipe decomposition step is finished. Table 5 shows an example of two clusters by set (i.e. in the table, 4 clusters are presented, since they are two sets) and their triples for the "potato" ingredient and its class (the set for "tuber" class).

The recomposition step involves the creation of new recipes from the vectors of the clusters and the language model (LM for mixtures). Such recomposition was performed through a genetic algorithm (GA) with the individual modeled as follows: each gene is associated with an ingredient in the ingredient list of the new recipe (generated by the Creative Flavor Pairing API); and the value that each gene can assume is the triple value (id, weight, and class indicator) of the set of recipes decomposed for that ingredient and its class. Thus, each gene will contain a reference to a sequence of actions of one of the ingredients. The GA of this approach presents evaluation, selection, crossover and mutation operations. This GA is similar to the PG described in Section V-A2 in the following aspects: it was implemented using the DEAP framework; used the fitness function presented in (5); used the tournament of size 3 for selection operation; presented a crossover based on population doubling and choosing the smaller nondominated front; and the *Hall-of-Fame* always contains the first nondominated front between the population and the *Hall-of-Fame* of the previous iteration. However, the crossover used in this genetic algorithm consisted of a simpler operation compared to the GP approach. First, a copy of the vector was made containing the genetic material of the two chosen parents. After that, a random cut point was chosen for these two vectors, which can be any position except the last position (so that you can use the genetic material of both parents). Then, to compose the child, all genes up to the cutoff point were chosen from one parent and after the cutoff point to the end of the vector were chosen from the other parent. An example of this crossover operation is shown in Fig. 14. The integer values of the content of the genes in the image is merely illustrative to simplify visualization.

Table 6 shows an example of an individual of this genetic algorithm (with the true content of the genes).

Note that by adding the weight of the triples of an individual, the total weight indicates how often (and suitable)

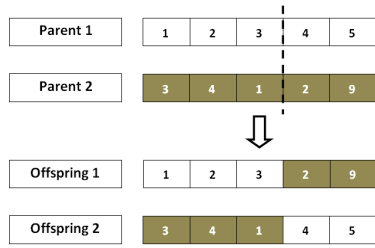


FIGURE 14. Example of recombination operation for a genetic algorithm.

TABLE 6. Example of an individual of GA.

Ingredient	Genotype	Associated Vector
“potato”	(2, 3, False)	[“serve_recipe”, “add”, “fry”, “add”, “potato”]
“olive oil”	(0, 6, True)	[“serve_recipe”, “add”, “fry”, “add”, “oil”]
“salt”	(5, 6, False)	[“serve_recipe”, “add”, “salt”]

the preparations of the ingredients of this individual are (i.e. the total weight can be treated as a quality criterion). On the other hand, triples with the active class indication should be penalized because the most suitable preparation for a class of ingredients is not always the best preparation for a particular ingredient of this class. Another detail that we can realize is that not necessarily the resultant preparation graph of an individual will respect the fifth rule of normalization, about arity, defined in Section IV-C (e.g. if there was no “add” after “fry” in the first two ingredients of the Table 6, then the “fry” action presents arity 2 - and only the “add” and “mix” actions can have arity greater than 1). In order to quantify the quality of the preparation according to the criterion of arity, it is performed a count of the actions that do not follow this fifth rule and it is divided by the quantity of actions in the recipe preparation (i.e. the lower this average arity error, the better the quality of the recipe). Thus, the evaluation of individuals in the genetic algorithm uses a multi-objective criterion:

- The first objective was to maximize the sum of the total weight of the individuals (if any triple of a gene is of a class of ingredients, the weight of triple falls to 10% of its value - for example, the total weight of the individual in Table 6 is 9.6).
- The second objective was to minimize the error of arity and the perplexity of the language model (LM for mixtures).

This second objective can only be evaluated through the composition of the recipe using the gene vectors, from which we obtain the graph of the recipe preparation. The algorithm used for this composition was based on the algorithm of determinization of the finite automaton (with an adaptation in the sense and content of the edges). Such an algorithm has the following steps:

- Creation of a graph containing the sequence of actions joined only in the last action, the root action “serve_recipe” (depth 0).

- Unification of nodes with equal content that has an edge with a common node (edge destination node), eliminating the redundant edges.
- Evaluation of the union of the nodes in breadth (i.e. the need to join all nodes of the same depth before leaping to achieve greater depth).

Fig. 15 shows the steps of this composition algorithm to generate the method of preparation of the recipe encoded in the genes of the individual described in Table 6. The sequence of steps of the composition algorithm goes from the figure (a) through to (d), one figure per depth.

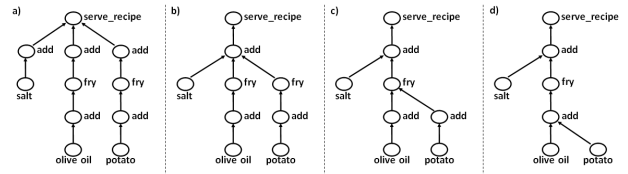


FIGURE 15. Example of recipe composition.

The final result of this approach, which is the recipe with preparation, is obtained from this algorithm of composition applied to each individual of the final *Hall-of-Fame* of the genetic algorithm.

C. EVALUATION OF NOVELTY FOR CULINARY RECIPES

In order to evaluate the novelty of the preparation instructions of the culinary recipes, it was used the graph dissimilarity metric based on the Kernel of simple paths. A simple path is any sequence, without repetition, of consecutive nodes in the graph structure. It is worth to mention that all the paths p must have one endpoint in the last action of the recipe (i.e. “serve_recipe”) and the other endpoint in any other state (at any depth, including the ingredients). Example of some simple paths of the graph of Fig. 15 (only those beginning with the “serve_recipe” node) are: [“serve_recipe”, “add”]; [“serve_recipe”, “add”, “salt”]; [“serve_recipe”, “add”, “fry”, “add”]. Note that this simple path shows repetition of “add” content, however are from different nodes (i.e. the structure node should not repeat); etc. The idea behind the Kernel of simple paths technique is to create a transformation function ϕ_p that can indicate the existence (returning value 1) or non-existence (returning value 0) of a simple path p in a given graph [23]. Thus, in this new domain it is possible to calculate the similarity between the graphs by (6) [23], and the dissimilarity by (7):

$$sim(G, H) = \frac{\sum_{\forall p} \min(\phi_p(G), \phi_p(H))}{\sum_{\forall p} \max(\phi_p(G), \phi_p(H))} \quad (6)$$

$$dis(G, H) = 1.0 - sim(G, H) \quad (7)$$

where G and H are two graphs; $sim(G, H)$ is the similarity value between G and H and $dis(G, H)$ is their dissimilarity.

This distance metric (the dissimilarity) can then be used to evaluate the novelty of a recipe in relation to the database.

VI. EXPERIMENTAL SETUP

In this section the experimental setup is presented, with the criteria, objectives, expected results and configuration of the evaluations. The details of parameterization and training of LMs and Evolutionary Algorithms involved in such experiments are also shown. In Section VII the results of these experiments are presented.

A. EVALUATION OBJECTIVES AND CRITERIA

The experiments carried out in the present work have the following objectives: (i) to evaluate if the two architectures of LMs proposed for the ReProg and ReComp approaches were able to absorb knowledge of the human recipe database that would allow perplexity to be a good metric of quality for evaluation of recipes; and (ii) assessing whether the system as a whole (i.e. not just the LM, as in the previous item), for the ReProg and ReComp approaches, was able to generate quality recipes by human assessment. Performing experiments for objective (i) is necessary because there are no work in literature that uses LMs applied to recipes, whose structure differs greatly from that of the one in texts addressed by them. The experiment (ii) is necessary to demonstrate, through human evaluation, that the LM quality metric reflects in quality recipes. The recipe quality criteria that LMs are expected to learn varied between the two approaches, ReProg and ReComp, and will be presented below.

The ReProg approach is less restricted in terms of possible combinations of the recipe preparation graph, but this greater space of combinations allows the creation of nonviable recipes. Thus, one of the recipe quality criteria evaluated in the experiments is the consistency of the preparation, which is understood as the absence of sequences of incompatible actions. That is, an incompatibility of an action with respect to some other action that precedes it or with some ingredient that undergoes this action. These incompatibilities may be due to physical impossibilities (e.g. cutting liquid is an action impossible to perform) or different conditions from which the action is defined, so that the action can not be performed or has no effect on the preparation (e.g. frying is an action defined only in the presence of oils).

The ReComp approach presents greater consistency in recipe preparation at the cost of a reduction in the space of combinations. Thus, it makes no sense to assess whether a recipe is viable. This approach is similar to [3], however an LM was added with the function of promoting better mixtures of ingredients throughout the steps (e.g. the proper use of seasonings throughout the preparation). That is, the recipe quality criterion proposed in this model relates to the mixtures of ingredients throughout the steps, which are understood as the points where the step sequence of two ingredients are combined, which may occur in some action in the middle of the recipe or only in its last action (i.e. “serve_recipe”). Quality mixtures for this LM are those that resemble the ones which occur in the database of human recipes, among which we chose to evaluate: the mixture of seasonings to the

ingredients that need seasoning; and the mixture of oil to the ingredients in actions requiring it (e.g. cooking processes by immersing ingredients in oil). It is possible that there are other mixtures, as well as these two, in which one ingredient has a secondary role (i.e. one ingredient has the function of mixing with another to perform an auxiliary function).

Two other quality criteria for recipes are evaluated in the experiments: taste perception, which consists of human evaluation of the ingredients and preparation in the recipe text regarding the past experiences of flavor (i.e. based on recipes that people know, evaluate the ingredient and preparation mode of a new recipe); and taste itself, which involved human evaluation by tasting the recipe performed (i.e. taste assessment of a dish).

Having presented the objectives and defined the evaluation criteria, the details of the evaluations are presented below.

B. PARAMETRIZATION AND TRAINING OF THE APPROACHES

1) RECIPE DATABASE USED IN EXPERIMENTS

The database of human recipes used in the experiments were described in Section IV-A. Table 7 shows the number of recipes used from each recipe source.

TABLE 7. Database size for each recipe source.

Recipe source	Size
Wikitaable	1439 recipes (all recipes)
Wikia	500 recipes (category of American recipes)

All recipes (i.e. the 1939 recipes contained in the resulting database) were normalized as explained in Section IV-C.

2) PARAMETERIZATION OF RECIPE DECOMPOSITION

In the ReComp approach, the number of clusters used to divide the sets of ingredients, as explained in Section V-B2, was 3 (i.e. the K in the K-Medoids algorithm equals 3). In this way, each ingredient has a set with 3 divisions, each division being a typical preparation, raised by the K-Medoids algorithm, that this ingredient suffers in database human recipes. The same was done for the class of ingredients, each class of ingredient (e.g. “vegetables” for the lettuce, cabbage and etc. ingredients) was divided into 3 clusters.

3) LANGUAGE MODELS TRAINING

The language models of the two approaches, ReProg and ReComp, were trained through K-fold cross-validation, which consists in dividing the data into K equal parts (one for testing and the remaining $K - 1$ for training) and generating K models (where the test portion is chosen, among these K , different for each model). Table 8 presents the parameters of this training for the LM of both approaches.

Thus, 8 LMs were created for each approach. However, in the experiments that involved the evaluation of the system, only the LM with the lowest perplexity value of the test

TABLE 8. Training parameters of LMs.

Parameter	Value
K (number of partitions in K-fold)	8 (parts)
MLP input layer	-
MLP hidden layer	200 neurons
MLP output layer	91 neurons
MLP training iterations	1500

portion in the cross-validation was chosen to compose the system.

4) PARAMETRIZATION OF EVOLUTIONARY ALGORITHMS

For both the PG of the ReProg approach and the AG of the ReComp approach, the parameterization of the evolutionary algorithm was presented in Table 9.

TABLE 9. Parameters of evolutionary algorithms.

Parameter	Value
Population	200
Maximum iterations	200
Mutation rate	10%
Density of the neighborhood calculation grid	0.2 x 0.2

Where the density value of the grid of the neighborhood calculation indicates the dimensions of each grid cell in the objective space, where each objective is normalized between 0 and 1.

In addition to these parameters, the AG of the ReComp approach should penalize the weight of the triples derived from the sets of classes of ingredients during the process of recomposing recipes. This penalty should reduce to 10% the value of the weight in these triples for the calculation of the first objective of the GA.

C. EXPERIMENTAL DESIGN

In this section we present the types of analyses performed in the experiments and a brief description of such experiments.

The experiments were divided into two types of analysis: objective analysis (denoted by the prefix “OA”), which seeks to evaluate the performance of a binary classifier based on the perplexity metric (i.e. low perplexity indicating good quality and high indicating poor quality) in assessing recipes from an artificial base annotated according to the quality criteria defined in Section VI-A; and subjective analysis experiments (denoted by the prefix “SA”), which consisted of human evaluations, in the quality criteria defined in Section VI-A, of recipes generated by the system. Table 10 shows the experiments chosen to perform these two types of analyses. Where, in the “Name” column, names were assigned to the evaluation experiments; “to evaluate” indicates which part of the system, and from which approach, the evaluation was performed, according to the evaluation objectives reported in Section VI-A; “LM used” indicates which language models, those reported in Section VI-B3, were used in the evaluation

TABLE 10. Summary description of the experiments.

Name	To Evaluate	LM Used	Description
<i>OA_cut</i>	LM of ReProg	All LM (in K-fold)	Classifier for recipes with cutting actions on solids or liquids.
<i>OA_reduce</i>	LM of ReProg	All LM (in K-fold)	Classifier for recipes with reduction action on solids or liquids.
<i>OA_cool</i>	LM of ReProg	All LM (in K-fold)	Classifier for recipes with chilling actions on hot or cold ingredients.
<i>OA_fry</i>	LM of ReProg	All LM (in K-fold)	Classifier for recipes with frying actions in the presence or absence of oils.
<i>OA_oil</i>	LM of ReComp	All LM (in K-fold)	Classifier for mixtures of ingredients with or without oils in the frying actions.
<i>OA_seasoning</i>	LM of ReComp	All LM (in K-fold)	Classifier for mixtures of ingredients with or without seasonings in frying actions.
<i>SA_feasible</i>	ReProg	The best LM (in K-fold)	Survey to evaluate system recipes in the feasible and taste perception criteria.
<i>SA_mixture</i>	ReComp	The best LM (in K-fold)	Survey to evaluate system recipes in the mixture quality and taste perception criteria.
<i>SA_dish</i>	ReComp	The best LM (in K-fold)	Tasting of a recipe generated by the system.

(if more than one, the experiment involves an evaluation for each LM); “Description” explains briefly the evaluation.

The objective evaluation experiments (“OA”), implemented using classifiers, aimed to evaluate only the LMs alone (i.e. without the rest of the system). The subjective evaluation experiments (“SA”), through surveys or tasting, had as purpose the evaluation of the system as a whole. Figures 16 and 17 present the flowcharts of these two types of evaluation performed.

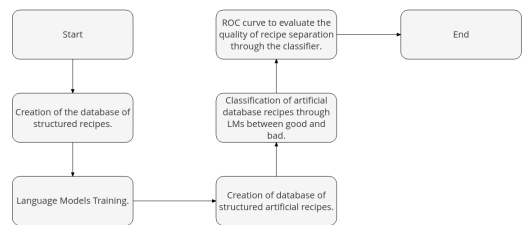


FIGURE 16. Flowchart for objective evaluations (both approaches).

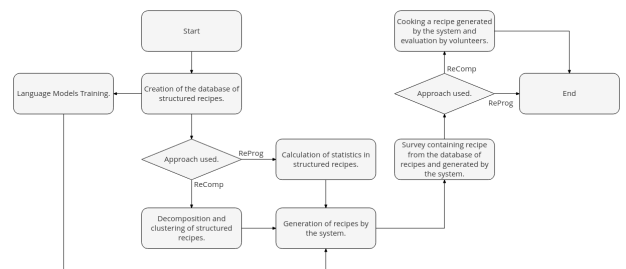


FIGURE 17. Flowchart for subjective evaluations (both approaches).

In the following sections, these experiments are presented in more detail.

1) EXPERIMENTS FOR THE ReProg APPROACH

For the LMs architecture of the ReProg approach, explained in Section V-A1, 4 objective evaluations, named as *OA_cut*, *OA_reduce*, *OA_cool* and *OA_fry*, were performed.

These 4 evaluations involved using the 8 LM cross-validation, explained in Section VI-B3, as binary recipe classifiers. These classifiers are based on the perplexity metric, so that high perplexities indicate low quality and low perplexity indicates high quality of the recipes. The 4 evaluations consisted of the creation of 4 artificial recipe databases, with recipes selected based on the consistency of the preparation defined in Section VI-A, and evaluation of the classifiers' ability to separate these recipes correctly (i.e. that the perplexity of the 8 LMs presented high values for nonviable recipes and low values for viable recipes). The classifiers are evaluated by using the Receiver Operating Characteristic (ROC) curve, which consists of asserting the relationship between false positives and true positives when sliding the class separation threshold. The expected result for the ROC curve is high values of true positives and low values of false positives along the whole curve, generating an Area Under the Curve (AUC) above 0.5 and close to 1. It is important to note that the final result of the experiment is the average ROC curve from all the LMs used (i.e. each ROC curve presented in the results section is the composition of the ROC curves of the various models).

Due to the fact that there were no works in the sense of evaluating inconsistencies in the preparation of recipes, even in the culinary literature, four intuitive inconsistencies were chosen, which derive from physical impossibilities or from the culinary definition of actions. The four inconsistencies, and their respective artificial databases, are detailed as follows:

- a) *OA_cut* - The artificial database of this experiment is based on the physical impossibility of cutting action on liquid ingredients. This artificial database consisted of 50 small recipes composed of 6 cutting actions (cut, chop, dice, mince, slice and shred) on 25 liquid ingredients, noted as low-quality recipes, and those same cutting actions over 25 solid ingredients, noted as high quality recipes. Example of sequence of actions of these small recipes: ["serve_recipe", "cut", "milk"], ["serve_recipe", "chop", "onion"] and etc. Not all the recipes created for the artificial database *OA_cut* are used for the evaluation of the classifiers via the ROC curve. In order to make the analysis independent of cutting actions specificities (e.g. cut shape), only the least perplexing (better quality) recipes were used for the evaluation. For example, the recipe of the sequences of steps ["serve_recipe", "chop", "onion"] was used in the evaluation of the classifiers, while the remaining recipes for "onion" (e.g. ["serve_recipe", "cut", "onion"], ["serve_recipe", "dice", "onion"], and so on) were not used.
- b) *OA_reduces* - the artificial database of this experiment is based on the definition of the reduction action, which is the heating of liquid ingredients with the intent of increasing its density (and consequently the concentration of some ingredients). The reduce action is most useful on liquid ingredients. This artificial database

consisted of 50 small recipes composed of the action reducing the same 50 ingredients of the *OA_cut* database, but the annotations were reversed (i.e. the recipes with liquid ingredients were noted as higher quality, while the recipes of solid ingredients were noted as of lower quality).

- c) *OA_cool* - The artificial database of this experiment is based on the definition of cool action, which is the cooling of hot ingredients or mixtures. Thus, the cool action is useless on frozen ingredients. This artificial database included all combinations (about 600) of recipes with 2 cooling actions, "cool" and "chill", applied to the same 50 ingredients of the *OA_cut* database, but having these ingredients suffered before 2 freezing actions ("frost" and "freeze" - recipes low quality) or 4 heating actions ("cook", "heat", "bake" and "boil" - recipes high quality). Example of sequence of actions of these recipes: ["serve_recipe", "cool", "bake", "salmon"], ["serve_recipe", "chill", "frost", "orange_juice"]. As in the *OA_cut* database, not all the recipes from this artificial database were used in the classifier evaluation. In order for the evaluation to be independent of the specificities of the actions (cooling, freezing and heating), two recipes were chosen for each ingredient, one annotated as viable and the other noted as nonviable, with the lowest perplexity values found. For example, recipes for the sequences of actions ["serve_recipe", "cool", "cook", "chicken_broth"], annotated as viable, and ["serve_recipe", "chill", "freeze", "chicken_broth"], annotated as nonviable, were used in the evaluation of the classifiers; while the remaining recipes for "chicken_broth" (e.g. ["serve_recipe", "cool", "heat", "chicken_broth"], ["serve_recipe", "chill", "cook", "chicken_broth"], ["serve_recipe", "cool", "freeze", "chicken_broth"] and etc.) were not used.
- d) *OA_fry* - The artificial database of this experiment is based on the definition of the fry action, which consists of the process of cooking ingredients on some oil. The database of artificial recipes included 5 low quality recipes consisting of the fry action on 5 oils (i.e. only the oils as an ingredient); and all combinations of fry action recipes on 12 ingredients alone (recipes also noted as low quality) and fry action on these 12 ingredients with 5 oils (recipes noted as high quality). Example of recipes: "fry" on "pork" and "olive oil". As in the *OA_cut* database, not all the recipes from this artificial database were used in the classifier evaluation. For the evaluation to be independent of the specific oil used in the fry process, only for the recipes containing ingredients with oils, recipes were chosen the with the lowest values of perplexity found (better quality) for each ingredient. For example, the "fry" recipe for "bacon" and "oil" was used to evaluate the classifiers, while the remaining recipes for "bacon" and some other oil (e.g. "fry" on "bacon" and "canola oil") were not used.

The recipes generated by the (complete) system were also evaluated. However, this evaluation was subjective (i.e. by a human assessment) conducted through a survey, named *SA_feasible*.⁵ The LM used in the system that generated the recipes of this survey was the model with the best perplexity of the test in the cross-validation presented in Section VI-B3.

TABLE 11. Size of the recipes generated for the *SA_feasible* experiment.

Recipe Size	Number of Ingredients	Maximum Depth
Small	4	5
Medium	6	6
Large	8 to 9	8

The choice of the recipes to compose the *SA_feasible* evaluation survey was as follows: 18 recipes were chosen, 9 of the database of human recipes and the other 9 generated by the system. Each of these groups of 9 recipes was divided into 3 parts, containing 3 small, medium and large recipes. These recipe sizes are described in Table 11. For each of these recipe sizes, the system was run 3 times for 5 different ingredient lists and the 3 recipes that appeared to be the most viable among all generated were chosen. It should be noted that the number of recipes generated by the system for each execution is not very large (i.e. Hall-of-Fame is small compared to the PG population), being of the order of the maximum depth chosen for the recipes (in this case, as seen in Table 11, was 5, 6 and 8 - that is, at most 8 recipes). Thus, although it involved 3 runs of the system, the number of recipes generated was small compared to the space of possible preparation combinations (which even in the smallest case is exponential - for example, soup space with 4 ingredients suffering a single action per ingredient, out of 90 possible actions, is in the order of 90^4 combinations).

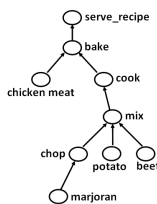


FIGURE 18. A recipe created by the ReProg approach (in graph form).

After choosing 9 human recipes and 9 recipes generated by the system, a textual description was made for the preparation graph in order to write the 18 recipes in natural language. Figures 18 and 19 exemplify these two versions, graph and textual, of a these recipes. Some questions in this survey were based on [24], but the questions related to *SA_feasible* assessment were as follows:

- a) Is the recipe presented feasible?
- b) With respect to the flavor, is the recipe tasty?

⁵This survey is in Portuguese and can be accessed at: <https://goo.gl/forms/nSI9HzVqo8meymWv2>

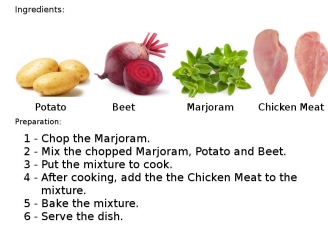


FIGURE 19. A recipe created by the ReProg approach (in textual form).

2) EXPERIMENTS FOR THE ReComp APPROACH

For the LMs architecture of the ReComp approach, explained in Section V-B1, two objective evaluations, named as *OA_oil* and *OA_seasoning*, were performed. As in Section VI-C1, these 2 evaluations involved using the 8 LM of cross-validation as binary classifiers of recipes based on perplexity metrics. The ROC curve was calculated to evaluate the performance of these classifiers. The criterion used to note the recipes from the artificial databases of these 2 evaluations was the criterion of mixture quality of ingredients throughout the preparation defined in Section VI-A (i.e. recipes noted as having high mixture quality or low mixture quality).

The two artificial databases created to evaluate the classifiers were detailed as follows:

- a) *OA_oil* - The artificial database of this experiment is based on the secondary role of the oil ingredient in frying of other ingredients (i.e. mixtures of oil and other ingredients promoted by fry action). Thus, the oil mixtures with other ingredients in frying consists of mixtures of good quality, while the mixtures of ingredients that contains no oil in frying consists of mixtures of poor quality. The artificial database has 192 small recipes consisting of the fry action on 2 ingredients, the first one being one of 12 ingredients and the second one may be one of the other 11 remaining ingredients, for bad mixture quality recipes, or one of 5 oils, for good mixture quality recipes. Example recipes: “fry” on “onion” and “pepper” (poor mixture quality); and “fry” on “onion” and “vegetable_oil” (good mixture quality). Not all the recipes from this artificial database were used in the classifier evaluation. For the evaluation to be independent of the second ingredient of the mixture, it added some other ingredient or specific oil, for each ingredient in the first position of the mixture two recipes were chosen, one noted as good mixture quality and the other noted as poor mixture quality, with the lowest perplexity values found. For example, the “fry” on the mixture of “pork” and “oil” (good mixture quality) and the “fry” on the mixture of “pork” and “potato” (poor mixture quality) were selected, while all recipes with “fry” on the “pork” mixture with any other ingredient were not used.
- b) *OA_seasoning* - The artificial database of this experiment is based on the secondary role of seasoning ingredients in frying of other ingredients (i.e. mixtures of

seasonings and other ingredients, which bring seasoning, promoted by fry action). Like the *OA_oil* experiment, mixtures with these secondary ingredients, in this case seasonings, frying with other ingredients consist of good quality mixtures, whereas the absence of these secondary ingredients consists of poor quality mixtures. The artificial database has 117 small recipes composed by the fry action on 2 ingredients, the first one being one of the 9 ingredients and the second can be one of the other 8 remaining ingredients (for bad mixture quality recipes) or one of 5 seasonings (for good mixture quality recipes). Example of recipes: “fry” on “beef” and “salmon” (poor mixture quality); and “fry” on “beef” and “garlic” (good mixture quality).

As with *OA_oil*, not all artificial database recipes were used in classifier evaluation. In order to avoid specificities of the second ingredient of the mixture, two recipes were chosen for each ingredient in the first position of the mixture, one noted as good mixture quality and the other one noted as poor mixture quality, with the lowest perplexity values found. For example, the “fry” on the “egg” and “salt” (good mixture quality) and the “fry” on the “egg” and “bacon” (poor mixture quality) were selected, while all recipes with “fry” on the “egg” with any other ingredient were not used.

In this approach we also evaluated the recipes generated by the system. Two evaluations were done by humans, one was performed through one survey and the other through tasting of a recipe executed, these experiments were named as *SA_mixture*⁶ and *SA_dish* respectively. The model used in the system that generated the recipes for these evaluations was the model with the best perplexity of the test in the cross-validation presented in Section VI-B3.

The choice of recipe to compose the *SA_mixture* evaluation survey was as follows: there were 6 recipes generated by the system, divided into 3 pairs. Each pair contains recipes, denoted by *A* and *B* in the survey, which have the same list of ingredients, but with different preparation steps. Recipe *A* was generated by the system proposed in this approach, including LM, while the recipe *B* was generated by the system proposed in this approach, but without the use of an LM. That is, in this second recipe we used a completely similar system to the one proposed in [3], without the aid of a LM to guide the GA toward recipes with better mixtures of ingredients.

After choosing the 6 recipes generated by the system, a textual description was made for the preparation graph in order to write them in natural language. The questions present in the survey were as follows:

- Which recipe (*A* or *B*) uses seasoning and oils that better match the ingredients?
- With respect to the flavor, is the recipe tasty?

A recipe for this approach, using LM, was chosen to be executed in the *SA_dish* experiment because in the

⁶This survey is in Portuguese and can be accessed at: <https://goo.gl/forms/51srK3pib6Qxgu612>

comparison of the results of *SA_feasible* and *SA_mixture* surveys, which will be shown in Section VII, this configuration of the system was the one that presented the most promising results of taste perception. The choice of only this approach is due to the higher cost of this experiment. The choice of the recipe to be executed was as follows: 4 lists of creative ingredients were generated with 6 to 7 common ingredients found in Brazil. The system run for each of these 4 lists of ingredients and the recipe with the lowest arity error was selected by ingredient list. Among the 4 selected recipes, we chose the recipe with the preparation graph apparently more consistent and with better mixture quality of the ingredients. The textual description of this recipe is shown in Fig. 20.



FIGURE 20. Recipe generated by the system that was cooked (in textual form).

The *SA_dish* evaluation consisted of executing the recipe of Fig. 20 and presenting it for a small group of volunteers to taste. After eating the dish, each volunteer indicated whether the dish is tasty from a note on the following numerical scale: 1 means very bad, could not finish eating; 2, bad; 3, more or less; 4, good; and 5 means very good, would even repeat.

VII. RESULTS

A. RESULTS FOR THE ReProg APPROACH

The results for the objective analysis through classifiers explained in Section VI-C, are presented in Fig. 21, where the graphs from (a) to (d) are relative to the *OA_cut*, *OA_reduce*, *OA_cool* and *OA_fry* experiments, respectively; AUC (*Area Under Curve*) summarizes the efficiency of the classifier (1.00 is the ideal value); and “std. dev.” means the region which comprises up to a standard deviation of the mean curve (i.e. the value of the standard deviation along the mean curve).

For all 4 experiments in Fig. 21, an AUC greater than 0.83 was obtained, which is considered a good value (well above 0.50). The model (and the *perplexity* metric) were able to separate the viable recipes from the nonviable ones, so that there were several thresholds capable of bringing most of the true positives (cases where the threshold and

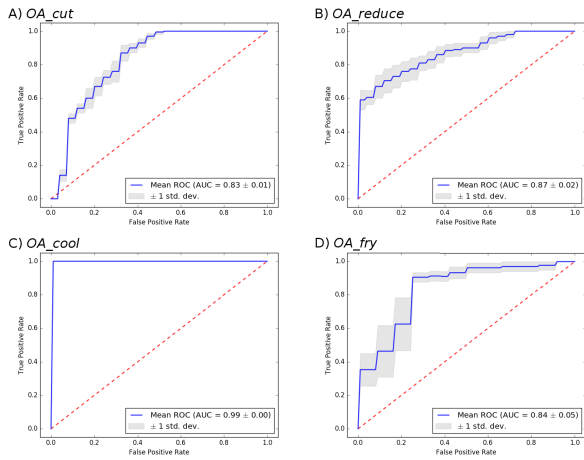


FIGURE 21. ROC curves for the classifiers of the ReProg approach.

label hit) without bringing many false positives. Particularly, the classifier of experiment *OA_cool* was practically perfect, obtaining an AUC of 0.99. One of the factors that may have led to this inconsistency to be better classified is the fact that it happens between actions, which occurs more frequently than ingredients in the database (in addition to the vocabulary of actions being smaller than that of ingredients, each recipe presents, in general, more actions than ingredients), and thus the amount of training data is more robust. Another detail that we can see in the experiments of Fig. 21 is that the mean deviation in the *OA_fry* experiment was the highest of these 4 experiments, with a value of 0.05. This was mainly due to two reasons: i) some recipes presuppose the use of oil for the “fry” action, leaving this ingredient often implicit; and ii) some recipes treat the action of preheating the oil for frying as a different “fry” action, such as “heat” action, which makes the language model does not treat oil as a direct ingredient of frying in these recipes. Since these two problems depend on the origin of the recipes, they may vary throughout the folds of the cross-validation, generating models with adverse quality between the folds that explain this greater variation.

Regarding the second type of analysis, the experiment *SA_feasible*, a total of 36 respondents completed the survey. The relative amount of positive evaluations regarding feasibility for each group of recipe size, which were defined in Table 11, is shown in Fig. 22. We can see that both groups of recipes generated by the system and those generated by humans had satisfactory values, being considered feasible by at least 78% of the evaluations. Overall, human evaluations had slightly better results (5% to 6%), with the exception of the medium-sized (6-ingredient) recipes group. The average evaluation for taste perception, where 0% means bad taste and 100% means good taste is presented in Fig. 23 for each group of recipe sizes. The human-generated recipes had satisfactory values, presenting an evaluation of at least 57% (between medium and good taste). Larger-sized human recipes achieved the best result, with an average of 82% (close to good taste), at least 22% above the other sizes. For the small

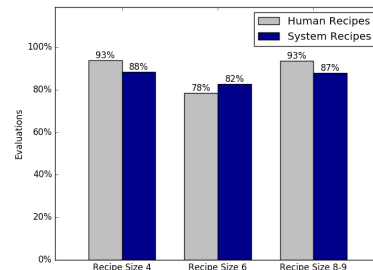


FIGURE 22. Result of evaluating the recipes as feasible (ReProg approach).

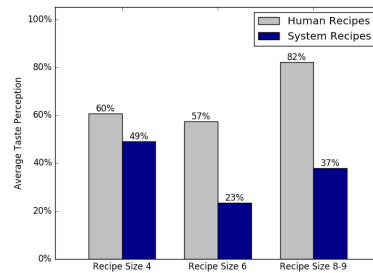


FIGURE 23. Result of evaluating the recipes as tasty (*SA_feasible* - ReProg approach).

and medium-sized human recipes, the result were very close, only 3% difference. Thus, there seems to be a tendency to better evaluate the taste perception of larger recipes, possibly because people consider these recipes more complete. The same behavior seems to occur, to a lesser extent, between large and medium-sized recipes generated by the system. Larger-sized recipes had an average evaluation score of 37% (medium to bad taste), while medium-sized recipes had a 23% (also medium to bad taste) result, 14% below the larger size. On the other hand, the smallest recipes got a better result than both, with an average evaluation of 49% (with a medium taste perception). This is due to the fact that the search space for the generation of preparation steps is smaller for the recipes with fewer ingredients, making the algorithm reach better recipes (with respect to preparation steps) faster than in the recipes with more ingredients. This second factor conflicts with the predilection factor for larger recipes sizes.

Another important detail is that these 9 system-generated recipes (the recipes of experiment *SA_feasible*) presented a dissimilarity more than 75% in relation to any recipe in the database, which means that the recipes are also novel.

B. RESULTS FOR THE ReComp APPROACH

The results for the first type of analysis in this approach are presented in Fig. 24 (the ROC curves of the model as a classifier). For the classifier of *OA_oil* an AUC of 0.97 was obtained, that is, the classifier was almost perfect. As for *OA_seasoning*, an AUC 0.77 was obtained, which is a good performance. For this second classifier there was a deviation from the high average of 0.07, probably due to the fact that the number of seasonings (salt, herbs, spices and even

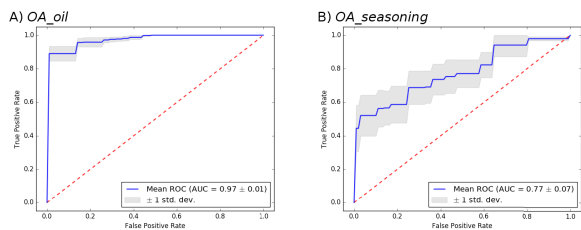


FIGURE 24. ROC curves for the classifiers of the ReComp approach.

some oil infusions) is higher than that of frying oils. In this way, the model for each fold of the cross-validate would have greater difficulty in generalizing the seasonings than the oils, leading to adverse (specialized) performances for each section of the training database.

Comparing the *OA_oil* with the *OA_fry*, presented in the previous section, we can see an improvement in the performance of the classifier and reduction of the mean deviation. Between these two classifiers, the recipes were almost all the same (with the exception of nonviable recipes with only one oil as an ingredient) the Language Models were of different architecture. The language model of *OA_oil* is able to treat oil as an ingredient of frying even though it is not a direct ingredient of the frying action, which reinforces the previously reported problem for the largest deviation of the mean in *OA_fry*. Thus, the ReComp approach model is more robust than the ReProg approach in these cases. However, the AUC of the ReProg approach presented better overall results, with a minimum value found of 0.83 compared to 0.77 of ReComp. However, both approaches still obtained quite satisfactory results, which demonstrates that Language Models were able to learn, based on human recipes, how to separate good quality recipes from those of poor quality (in the criteria adopted for the experiments).

As can be seen in Section VI-C, the second type of analysis performed for this approach relied on two experiments: *SA_mixture* and *SA_dish*. The *SA_mixture* experiment, which consisted of a survey, was completed by a total of 31 respondents. The average evaluation for the quality of the use of oils and seasonings between the two versions (generated by the system with and without language models) of each recipe is presented in Fig. 25, where 0% means that the quality of the recipe is bad and 100% means that the quality is good on this criterion. In general, the versions of the recipes generated by the system with language model had the best results, having an evaluation of at least 52% and obtaining for the three recipes an average of 63.7%. The recipes generated by the version of the system without language models presented an average quality score of 36.3%. As can be seen, the quality assessment between the two groups of recipes is complementary (i.e. the sum gives 100%), this being because respondents were asked to benchmark the quality of one group over the other. These results demonstrate that the language model can assist the system in generating recipes with better mixtures of the ingredients along the preparation steps (in this case,

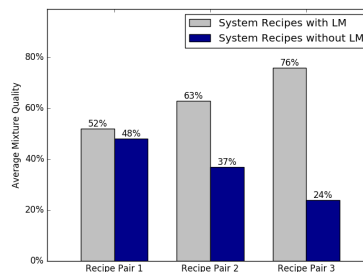


FIGURE 25. Result of evaluating the recipes as mixture quality (ReComp approach).

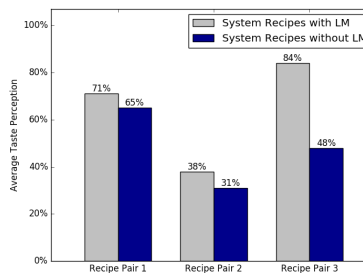


FIGURE 26. Result of evaluating the recipes as tasty (*SA_mixture* - ReComp approach).



FIGURE 27. Dish for taste evaluation by volunteers.

mixtures containing oils and seasonings). The average taste perception for these same recipes are shown in Fig. 26. In recipes pairs 1 and 3, for both systems, the result obtained was between medium and good, while for the recipe pair 2 it was between medium and bad. But, in general, the recipes of the system with language model obtained a better average taste perception, with an average value for the three recipes of 64.3%, between medium and good.

The *SA_dish*, which consists of tasting a recipe generated by the system with language model, had the participation of 18 volunteers. The amounts selected for the ingredients were: 3 egg whites; salt to taste; 1 leaf of mint per serving; 3 chicken tenderloin; 150 grams of cream cheese; and a pinch of saffron (just to color the cream cheese mixture). The time to bake the cream cheese mix was 50 minutes at 200 degrees celsius. The chicken was fried until golden. The dish was like that of Fig. 27.

The average score obtained in the evaluation of the volunteers was 4.72 out of 5, between good and very good (93% of the taste rating scale used in this experiment). This result demonstrates that the proposed system not only generates recipes with a good taste perception (results from the experiment *SA_mixture*), but also was able to generate a tasty creative recipe.

The 7 recipes generated by the system with this approach (the recipes of experiment *SA_mixture* and *SA_dish*) had a dissimilarity greater than 70% in relation to any recipe of the database. This demonstrates that this approach was also able to generate the preparation instructions with novelty.

VIII. CONCLUSION

The objective evaluations, using classifiers based on language models, have demonstrated the quality of such models in evaluating nonviable recipes and bad mixtures of ingredients in the preparation steps (e.g. bad use of seasonings), with AUC values above 0.83 and AUC of at least 0.77, respectively.

In the evaluation by online survey, an average evaluation of the recipes as feasible of 85.6% and an evaluation of the good use of oils and seasonings with an average of 63.6% (between medium and good) were obtained. These results demonstrate that language models and perplexity metrics can guide the recipe generation algorithm toward recipes with more consistent preparation steps, in the case of a less restricted generation approach (*ReProg* Approach), Language Models also guide the algorithms for yielding recipes with better mixtures of ingredients throughout the preparation steps, in the case of the approach with the consistency of the preparation more guaranteed (*ReComp* Approach). Regarding the taste perception of the generated recipes, only an intermediate result for small recipes was obtained in the *ReProg* Approach, with an average taste perception around 49% of the scale, while for the *ReComp* Approach the results were better. In the *ReComp* Approach, the average evaluation of taste perception was around 64.3% of the scale, between medium and good, and the average taste of a cooked recipe was 93% of the scale, close to good taste.

In addition, recipes generated from both approaches presented a dissimilarity for any database recipe by at least 70%, even for the *ReComp* Approach, which were based on a direct decomposition of the database recipes.

In general, the *ReComp* approach presented the best results and it was possible to achieve the goal of creating a system capable of generating complete creative recipes (i.e. with a creative ingredient list, a sequence of consistent preparation steps and a good use of oils and seasoning in the preparation).

IX. FUTURE WORK

As a future work, we propose to improve the database of recipes by adding more recipes than the 1939 used in the present work and improving the quality of the NLP Parser (e.g. by adding specialized ingredient inference strategies for every possible action in the ontology of actions). For the architecture of the language model, one could try to overcome

the problem of exponential growth of the information considered in the prediction of the actions of the preparation graph for a number of time steps greater than 1, so as to make it possible to use an architecture based on recurrent neural networks (e.g. LSTM). A solution in this sense could be the creation of a criterion of importance for the actions of the preparation in order to reduce the size of the branch considered in the prediction of the model. Another possibility would be to evaluate other types of models (e.g. Bayesian models) to evaluate the preparation steps of the recipes and mixtures of the ingredients throughout these steps. In the API used to generate the list of ingredients, it will be useful to make changes so that the user can choose ingredients according to their category (e.g. meats, vegetables, seasoning and etc) and kitchen templates (e.g. Italian, French and etc).

ACKNOWLEDGMENT

The authors would like to thank CAPES, PUC Minas, CNPq, and FAPEMIG for the financial support.

REFERENCES

- [1] R. G. Morris, S. H. Burton, P. Bodily, and D. Ventura, "Soup over bean of pure joy: Culinary ruminations of an artificial chef," in *Proc. ICCV*, 2012, pp. 119–125.
- [2] R. K. Sawyer, *Explaining Creativity: The Science of Human Innovation*. London, U.K.: Oxford Univ. Press, 2011.
- [3] F. Pinel, L. R. Varshney, and D. Bhattacharjya, "A culinary computational creativity system," in *Computational Creativity Research: Towards Creative Machines*. Paris, France: Springer, 2015, pp. 327–346.
- [4] K. Grace and M. L. Maher, "Surprise-triggered reformulation of design goals," in *Proc. AAAI*, 2016, pp. 3726–3732.
- [5] A. Amorim, L. F. W. Góes, A. R. D. Silva, and C. França, "Creative flavor pairing: Using RDC metric to generate and assess ingredients combinations," in *Proc. Lighth Int. Conf. Comput. Creativity, ICCV*, Atlanta, Georgia, 2017, pp. 1–8.
- [6] Y.-Y. Ahn, S. E. Ahnert, J. P. Bagrow, and A.-L. Barabási, "Flavor network and the principles of food pairing," *Sci. Rep.*, vol. 1, no. 1, p. 196, Dec. 2011.
- [7] G. M. Shepherd, *Neurogastronomy: How Brain Creates Flavor Why it Matters*. New York, NY, USA: Columbia Univ. Press, 2011.
- [8] M. Zampini and C. Spence, "The role of auditory cues in modulating the perceived crispness and staleness of potato chips," *J. Sensory Stud.*, vol. 19, no. 5, pp. 347–363, Oct. 2004.
- [9] K. Grace, M. L. Maher, D. Fisher, and K. Brady, "Data-intensive evaluation of design creativity using novelty, value, and surprise," *Int. J. Design Creativity Innov.*, vol. 3, nos. 3–4, pp. 125–147, Oct. 2015.
- [10] C. França, L. F. W. Góes, A. Amorim, R. Rocha, and A. R. D. Silva, "Regent-dependent creativity: A domain independent metric for the assessment of creative artifacts," in *Proc. 7th Int. Conf. Comput. Creativity*, 2016, pp. 68–75.
- [11] L. Macedo and A. Cardoso, "The exploration of unknown environments populated with entities by a surprise–curiosity-based agent," *Cognit. Syst. Res.*, vol. 19, pp. 62–87, Sep. 2012.
- [12] P. Baldi and L. Itti, "Of bits and wows: A Bayesian theory of surprise with applications to attention," *Neural Netw.*, vol. 23, no. 5, pp. 649–666, Jun. 2010.
- [13] J. Billing and P. W. Sherman, "Antimicrobial functions of spices: Why some like it hot," *Quart. Rev. Biol.*, vol. 73, no. 1, pp. 3–49, Mar. 1998.
- [14] G. A. Fink, *Markov Models for Pattern Recognition: From Theory to Applications*. London, U.K.: Springer, 2014.
- [15] E. Arisoy and M. Saraçlar, "Multi-stream long short-term memory neural network language model," in *Proc. 16th Annu. Conf. Int. Speech Commun. Assoc.*, 2015, pp. 1413–1417.
- [16] R. Masumura, T. Asami, T. Oba, H. Masataki, S. Sakauchi, and A. Ito, "Latent words recurrent neural network language models," in *Proc. 16th Annu. Conf. Int. Speech Commun. Assoc.*, 2015, pp. 2380–2384.

[17] K. Irie, R. Schlüter, and H. Ney, “Bag-of-words input for long history representation in neural network-based language models for speech recognition,” in *Proc. 16th Annu. Conf. Int. Speech Commun. Assoc.*, 2015, pp. 1–5.

[18] X. Huang, A. Acero, H.-W. Hon, and R. Reddy, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*, vol. 95. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.

[19] E. Cromwell, J. Galeota-Sprung, and R. Ramanujan, “Computational creativity in the culinary arts,” in *Proc. FLAIRS Conf.*, 2015, pp. 38–42.

[20] A. Cordier, J. Lieber, P. Molli, E. Nauer, H. Skaf-Molli, and Y. Toussaint, “Wiki-Taaable: A semantic wiki as a blackboard for a textual case-based reasoning system,” in *Proc. 4th Workshop Semantic Wikis (SemWiki). 6th Eur. Semantic Web Conf., Heraklion*, 2009, pp. 88–101.

[21] R. T. Ng and J. Han, “CLARANS: A method for clustering objects for spatial data mining,” *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 5, pp. 1003–1016, Sep. 2002.

[22] J. B. Kruskal, “An overview of sequence comparison: Time warps, string edits, and macromolecules,” *SIAM Rev.*, vol. 25, no. 2, pp. 201–237, Apr. 1983.

[23] L. Ralaivola, S. J. Swamidass, H. Saigo, and P. Baldi, “Graph kernels for chemical informatics,” *Neural Netw.*, vol. 18, no. 8, pp. 1093–1110, Oct. 2005.

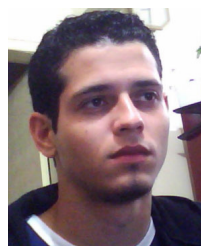
[24] L. M. R. D. Souza and L. F. W. Góes, “Evaluation of the human perception of creativity on the combinations generated by the creative food pairing system,” Dept. Inf. Syst., Pontifícia Universidade Católica de Minas Gerais, Belo Horizonte, Brazil, Tech. Rep., 2017.



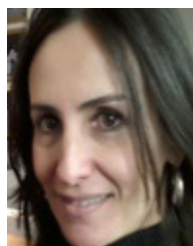
JOÃO RIBEIRO BEZERRA (Member, IEEE) was born in Belo Horizonte, Minas Gerais, Brazil, in 1994. He received the B.S. degree in computer science from the Pontifical Catholic University of Minas Gerais (PUC Minas), Belo Horizonte, in 2018. His research interests include artificial intelligence and applications in natural language processing.



LUÍS FABRÍCIO WANDERLEY GÓES (Member, IEEE) received the Ph.D. degree in computer science from the University of Edinburgh, U.K., in 2012. His main research interests include parallel programming and computational creativity.



WILLIAN ANTÔNIO DOS SANTOS (Member, IEEE) was born in Belo Horizonte, Minas Gerais, Brazil, in 1988. He received the B.S. degree in computer science and the M.S. degree in electrical engineering from the Pontifical Catholic University of Minas Gerais (PUC Minas), Belo Horizonte, in 2018. His current research interests include automatic speech recognition and computational creativity.



FLÁVIA MAGALHÃES FREITAS FERREIRA received the Ph.D. degree in electrical engineering from the Pontifical Catholic University of Rio de Janeiro, Brazil, in 2004. Her research interests include image, video and voice processing, digital processing in hardware, and scientific visualization.

...