With Sophistication and Grace: Tasteful Generation of Recipes

Johnathan Pagnutti

July 19, 2016

Abstract

Recipe generation is an emerging and active field in generative methods. However, what sorts of dishes someone finds good or bad change highly from person to person and even occasion to occasion. I propose that we change the generation goal from recipes that score highly on a quality metric to recipes that taste a particular way. Taste, however, is more complex than just a set of chemical reactions, often engaging in particular cultural concepts.

This first requires us to develop a computational model of taste, and then incorporate that model into a generator. I propose using machine learning to attempt to learn a taste model from online recipe databases. For generation, I propose using a genetic algorithm that is adaptive to user constraints on recipes, such as allergens or complexity level. Finally, I'll look at encoding cultural concepts around food in a semantic network and augmenting the genetic algorithm to generate recipes that have a particular meaning, as well as taste.

Chapter 1: Introduction

We live in an increasingly algorithmic world—from GPS navigation to AI stock brokers, AI routines to help assist in both everyday and specialized tasks are becoming commonplace. The culinary world is no exception: perhaps still in its infancy, recommender systems for recipes[1] and robotic bartenders[2] are starting to emerge and become generally accepted enough to be commercially viable.

Recipe generation has been studied, with generators ranging from case-based reasoning systems to genetic algorithms. Recipes themselves have been investigated as data structures in the search community, machine learning groups have studied recipe recommender systems and kitchen and cooking assistance software has been studied in HCI. However, only limited descriptive computational models of

[1] https://www.foodpairing.com

[2] http://www.barbotics.com/

the human sense of taste and flavor perception have been proposed, which hampers the ability of recipe generators to understand the flavors of the recipes they create, robotic cooks to understand the the texture of the food they prepare, and kitchen assistant tools to understand the nature of the tasks that they assist with.

This is far different from how humans prepare food—we're informed by chemical senses every step of the way. Consider the pop-culture chef, who tastes a bit of what they are cooking, pauses and exclaims a need for more spices. Our future preferences are informed by what our chemical senses tell us about past experiences with food, from both not choosing a foie-gras on a menu during a fine dinner, to trying to create that vegetable soup again, but this time with tarragon.

This advancement proposes a computational model of the human perception of taste based on big data analysis of online recipe repositories. From there, this advancement will explore the affordances of a software a model of taste, starting with a recipe generator for smoothies, Michaire, and then augmenting the generator with concepts from food and flavor aesthetics to enable the creative generation of recipes.

Research Questions

The core part of this advancement is a computational model of the perception of taste. This advancement is mostly focused on the accuracy of such a model. However, software should not exist in a vacuum, and this proposal also looks at an application of such a model: a smoothie recipe generator. We also acknowledge that recipes and food is often strongly tied to cultural symbolism and meaning, so we look at how to augment Michaire to reason over this new space.

Michaire and the taste model will be evaluated for accuracy. In addition, Michaire will be evaluated for usability, which should help reflect back on the usability of the original taste model. This research aims to answer the following questions:

RQ1: How faithfully can we create a computational model of the sense of taste?

The complex and nature of the multi-sensory perception of flavor makes building a fully comprehensive model of flavor a herculean task. However, successful models of even a part of the domain of edible foods is a useful contribution. Therefore, the model will be focused on proposing taste profiles for raw ingredients that do not require preparation to be safe to eat (most fruits and vegetables, for example). The proposed model will take as input combinations of raw ingredients and output a taste profile for the ingredients. Details of the model, as well as how to evaluate its faithfulness to human perception are in chapter 3.

RQ2: How can we generate recipes with a taste model?

Generative methods is a young, but surprisingly complex field. Based on the taste model, I plan to design and implement a smoothie recipe generator, Mischiare. Mischiare will follow the standard generate and test loop, with the ability to test against how the smoothies taste in addition to other metrics, such as recipe complexity and ingredient composition. Details on the implementation of the generator, as well as the evaluation scheme, are in chapter 4.

RQ3: How can Mischiare be augmented to generate creative recipes?

Computational creativity is another field that a taste model has obvious applications in. Although flavor and food aesthetics is not as explored as visual aesthetics, there have been exciting recent theories based around cultural symbol evocation. By augmenting Mischiare with a concept of flavor aesthetics, we can explore what artistic or creative recipe generation might look like. Details on the implementation of this generator, as well as the evaluation scheme, are in chapter 5.

Chapter 2 will focus on past and related work in recipe generation, as well as give a brief overview on the sense of taste. Chapter 3 is a system description of the model of taste, as well as an evaluation scheme for the model's accuracy. Chapter 4 is the system description of Mischiare, as well as an evaluation scheme for the smoothies it generates. Chapter 5 will start with an overview in flavor aesthetics, then describe the new modules that will be added to Mischiare to use this information. The chapter will conclude with the new types of generation these modules afford. Finally, chapter 6 will offer concluding remarks and an advancement timeline.

Chapter 2: Related Work

Recipes, Databases and Search

The search community has done work on how to best structure recipes, for fast indexing, fast retrieval, and minimal use of space. The latter has also been the focus of at least one twitter account; Maureen Evans compresses full recipes down to a tweet by hand[3].

[54] introduces the special properties of recipes that make them different from other data modeling domains. Recipes are loosely structured, behaviour oriented (the main part of the recipe is a procedure to follow for a dish) and bound by constraints (constrains bound particular actions or sequences of actions). Following that, the authors introduce the data structure of the cooking graph as part of a recipe model, shown in figure 1.

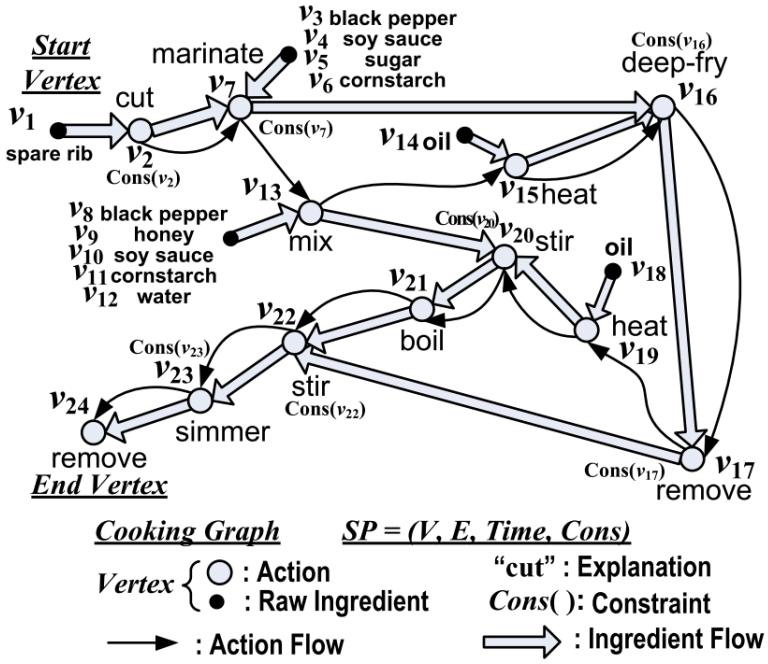

Figure 1. Cooking graph of 'Hu Die Gu' (in Chinese)
—'Braised Spare Ribs' (in English)

Figure 1: A 'cooking graph'. This is an example from [55], but follows the same basic structure as the graph definition in [54]. The black vertices denote raw ingredients, the blue vertices denote actions to take. Actions flow along black arrows, ingredients flow along pale blue arrows. Some action steps have a set of constraints that must be satisfied during that step (cons()).

Formally, [54] defines a recipe as a tuple: $R =< M, RP, SP >$, where M is the set of ingredients, RP is a set of properties applied to the recipe itself and SP is the

[3]https://twitter.com/cookbook

cooking graph. Each ingredient $m \in M$ has a unique ID as well as a set of properties (like name, quantity, an image, etc). Properties in RP are things like cooking style, region, an image of the dish, nutrition information, etc.

In this review of cooking graphs, I'm going to focus on the graphs presented in [55]. The graphs presented here are a simplified version of the model and graphs in [54], but encode more information in the graph representation (rather than tagging each node with extra information). In this case, $SP = (V, E, Time, Cons)$. V is a set of vertexes and E is a set of edges.

Each $v_i \in V$ represents either a raw ingredient or a cooking action. Each has a unique timestamp, $Time(v_i)$, representing the start time of v_i. Each v_i also has a set of associated constraints, $Cons(v_i)$. These constraints focus on things that should be true while a particular action is taking place, like the temperature of an oven or the final shapes of a sliced ingredient.

Each $e_i \in E$ represents either 'action flows' or 'ingredient flows'. Action flows describe the temporal sequence of actions in the graph (so $e^i_j(v_i \rightarrow v_j)$ implies that $Time(v_i) < Time(v_j)$). Ingredient flows describe how the raw ingredients go through each step (backwards tracking through the edges leads to the raw ingredients that are being manipulated at that step).

Although not the main point of this work, which is more focused on various properties and attributes that are derivable from this graph representation, this graph representation serves as an inspiring point for a lot of the work in recipe recommender systems and current recipe generators.

Recipe Recommendation Systems

The work done on recipe recommendation systems is fairly extensive, and this subsection will not provide a comprehensive review. Building on the insight from [55] (that graph structures represent recipes well), [50] uses similar graph structures to encode ingredient relationships in a recipe database to use for a recommender system.

The system is built around a pair of graphs, the ingredient complement network and the ingredient substitute network. The ingredient complement network is an undirected graph where nodes are ingredients and edges mark ingredients that frequently occur together. After building this network for a scraped recipe dataset with review scores, the authors only found a slight correlation between following the complement network and review scores.

However, the authors also leverage information inside of recipe reviews in online repositories. As shown in figure 2, online recipe reviews often (about 60% of the time) contain suggestions for modification to the existing recipe, so it can be difficult to decide of the provided score relates to the recipe at hand, or the suggested modification. Figure 2 also shows that online recipes are no stranger to the problem of online curation where 'the rich get richer'—highly rated recipes get more reviews as well as more ratings overall, which leads to bias in the set.

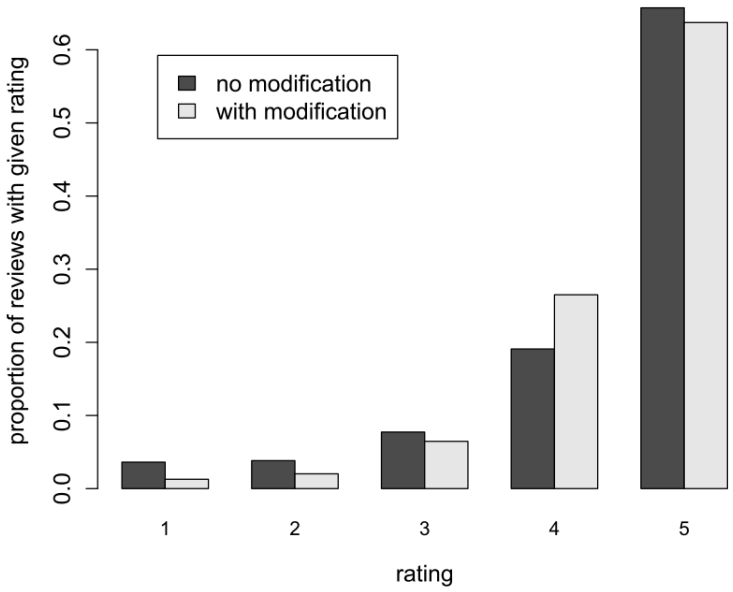

Figure 2: Graph of number of reviews to ratings, comparing and contrasting the reviews with modifications from [50]. This graph showcases inherent biases in online recipe websites, which, as will be discussed later, often requires generators to create bad examples to learn from.

The ingredient substitute network was built by scraping recipe comments for phrases that indicated replacement ('replaced a with b', 'substituted a with b', etc). Each node in the network is an ingredient, and each edge is directed indicating the replacement (replacing sugar with the artificial sweetener splenda would look like sugar $\rightarrow$ splenda in the graph). These edges are weighted by the proportion of the times the ingredient substitution was performed vs other substitutions for that ingredient, so if splenda was substituted for sugar 68% of the time, the sugar $\rightarrow$ splenda edge would have a 0.68 weight.

The substitution network was tested against a third graph, an ingredient preference network. The authors find high correlation between the substitution network and the preference network, indicating that the substitution network encodes something about user preference and high scoring recipes, rather than rote ingredient lists. This work lines up with other studies, in that individual user preference is more subtle than trying to find general 'optimal' ingredient combinations. Only by looking at how users were substituting ingredients did the network line up with user preference, and therefore encode the correct domain information.

Other work in recommender systems for recipes includes collaborative filtering (recommending recipes based on past reviews from similar users, not considering any domain knowledge) [12, 49, 57], finding a user's favorite ingredients based on browsing history [52] or nutritional needs / preferences [43, 13, 20].

Food Pairing Hypothesis

A particularly fateful meeting of Heston Blumenthal and Francois Benzi at an experimental cooking workshop in 2001 gave birth to the food pairing hypothesis (often referred to as the food pairing theory or flavor pairing theory/hypothesis). The main thrust is simple—the more aromatic compounds a pair of ingredients share, the better they taste together [9]. Blumenthal would describe this meeting, as well as his own discoveries about how well caviar and chocolate went together in an article for The Guardian in 2002[4]. The theory has grown to be quite popular, spawning off several new dishes, cookbooks and a website, where people could investigate the flavor pairing hypothesis themselves[5].

Although the flavor pairing theory is popular, it is not all encompassing. Cinnamon and tomato, for example, should taste poorly together according to the theory. However, this is a signature flavor combination of traditional Greek food, so one would imagine Greeks would disagree [9]. Also, the results are dubious when presented to non-experts. In an evaluation of the theory, Kort et al. found that when students were presented with mixtures of ingredients that conformed to the hypothesis, they did not rate them any higher than mixtures which did not [23].

However, it may be that the flavor pairing theory is two-tailed. Ahn et al would take a more data-driven approach to testing the flavor pairing hypothesis. In this work, the authors created a bipartite graph consisting of two types of nodes: 381 ingredients and 1,021 flavor compounds known to contribute to the flavor of each of the ingredients. The projection of this network is the flavor network, where two ingredient nodes are connected if they share flavor compounds. The links are weighted by the amount of compounds two ingredients share. A visualization of this graph is shown in figure 3 [1].

The flavor pairing hypothesis is a topological property of the network—two ingredients work well together if they have a heavily weighted edge between them. Ahn et al tested the network against 56,498 recipes from three websites: allrecipes.com, epicurious.com and menupan.com. The recipes were grouped into geographically distinct cuisines (North American, Western European, Southern European, Latin American, and East Asian). For North American cuisine, the flavor pairing hypothesis held true. The more a pair of ingredients were in a recipe together, the more likely they shared similar chemical compounds. However, the hypothesis proved completely false for East Asian recipes. The more two ingredients were used together, the less likely it was that they shared flavor compounds together [1].

Finally, one criticism leveled against the flavor pairing hypothesis is that it is an inaccurate model of how humans actually perceive flavor. Mautis de Klepper writes that "most people recognize vague groups or combinations of compounds, which they qualify with general descriptive terms like fruity, woody or meaty" [9]. This viewpoint corresponds with recent neuroscience discoveries about the similarities

[4] http://www.theguardian.com/lifeandstyle/2002/may/04/foodanddrink.shopping

[5] http://www.foodpairing.com/

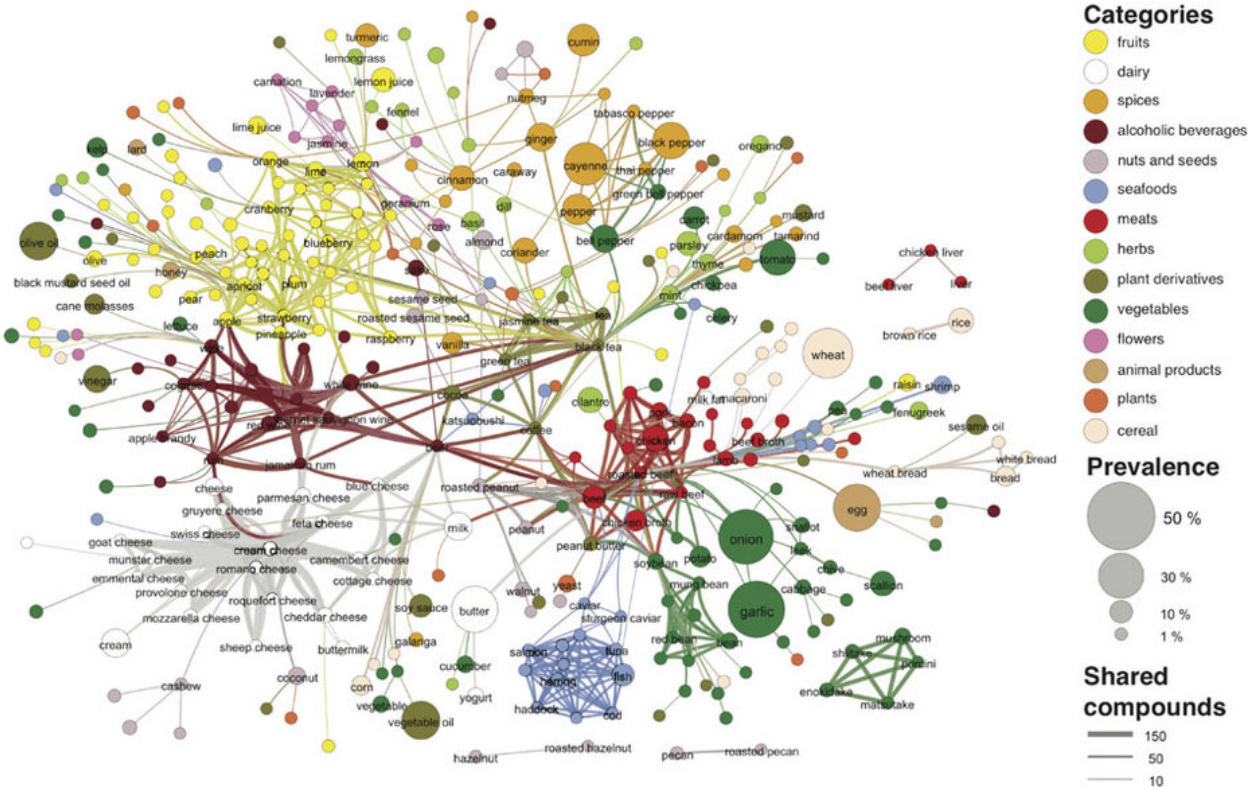

Figure 3: The flavor network. This visualization is not a complete view of the network, and is created via a backbone extraction method. The only links here are the statistically significant ones, where $p < 0.04$. This is why mushrooms appears to be unconnected to anything else—they mostly share compounds with other mushrooms

between smell and sight—namely, the sense of smell is realized by 'odor images' generated in the olfactory processing pathway in the human brain [42]. A review of how the sense of taste works with the tongue and other flavor receptors can be found here [46].

CHEF

One of the first recipe generation systems, CHEF uses case-based planning to create szechuan cooking recipes [17]. CHEF was novel at the time for incorporating a memory of past plans to help assist in avoiding problems and the creation of new ones. CHEF takes, as input, various ingredients and flavors that a new plan (recipe) should achieve (goals). CHEF starts the planning process by looking through a datastore of past problems associated with the input goals. Then, it searches through its memory of past plans to find one that satisfies as many goals as possible while avoiding as many 'accounted for' problems as possible.

After retrieving a usable plan, CHEF then modifies it via a set of rules to meet all the user-specified goals. The plan is then simulated through a separate set of hard-coded causal rules, and then CHEF checks the final state of the simulation against the original goals. If any goals remain unsatisfied, or CHEF hits a state in the simulation it wants to avoid, it marks the plan as a failure and goes into plan repair.

CHEF deals with plan failure by building a causal explanation as to why the plan failed. This is done by using a different set of causal rules that allow it to backchain from a failure node (either the state we want to avoid or unsatisfied goal) to the initial state. This failure chain is then used to find a strategy to deal with this failure. Each strategy contains a way to build a description of the current problem and how to fix it, which is passed to a hand-coded library of plan modifiers that transform the plan to avoid the failure state.

CHEF avoids having to constantly go through this complicated process of repair by storing plans intelligently—after repairing a plan to avoid failure, it updates its knowledge of which failures to be worried about given the original goals ('accounted for' problems), and stores the modified plan as a plan that satisfies the goals and avoids the failures.

Although a notable piece of software, CHEF requires a lot of authorial intent in order to work. A human needs to author the rules CHEF uses to modify a plan, the rules CHEF uses to simulate a plan, the ways CHEF can repair from failure, and the rules on how to do that given a particular failed plan. It would be rather nice to have our culinary AIs learn from existing recipes, so there is less human authorship required to get the system cooking.

PIERRE

PIERRE is a recipe generator for crockpot recipes (specifically, chilis, soups, and stews) that focuses on computational creativity [30]. PIERRE is a culinary case study into how to best split up an inspiring set, the set of artifacts that a computational system has 'seen', to best emulate human creativity. Unlike past work on

this split, PIERRE splits its inspiring set into a set of artifacts used for generation and a set of artifacts used for evaluation.

The recipe domain of crockpot recipes was chosen to reduce the work required to produce a working recipe. PIERRE creates novel ingredient lists, and then appends generic preparation instructions on them. The system builds its inspiring set by scraping recipes off of the World Wide Web. It then uses a manually created list of ingredients and measurements to parse the recipes in a consistent format. The ingredients are also placed in a hierarchy, with general classes at the top (fruit, vegetable), instances of those classes at the middle (beans is contained in vegetables, for example) and types of those instances at the bottom (beans breaks down into Butter Beans, Red Kidney Beans, Garbanzo Beans, etc). The top level of this hierarchy is referred to as the supergroup, the bottom level the sub-group.

PIERRE also scrapes a set of statistics for each ingredient. It keeps track of an ingredient's minimum usage, maximum usage, mean usage and an ingredient's standard deviation or frequency in the set.

This inspiring set is then split between a set for generation and a set for evaluation. For generation, PIERRE uses a genetic algorithm (GA). The population for the GA is initialized randomly, and the objective function is covered in the evaluation section. PIERRE implements crossover by selecting a random pivot index in a recipe, breaking it into two sub-recipes, and combining the top sub-recipe of one with the bottom sub-recipe of another. Mutation is implemented by adding, subtracting or changing one ingredient in a recipe.

For evaluation, PIERRE trains two multi-layer perceptrons (MLPs). PIERRE uses online ratings for each recipe in its inspiring set, the MLPs perform regression based on these ratings. The two MLPs are trained at different abstraction levels. One works on the supergroup level, the other on the sub-group level. The authors of PIERRE assume that any recipe found online has some merit, and therefore randomly construct some recipes that they assign a score of 0 to, in order to give the MLPs negative examples to learn from.

As shown in figure 4, PIERRE's ingredient lists seem feasible at a first glance, but often use strange amounts of particular ingredients (what is a quarter of a slice of bacon going to do against a pound of beef?) and occasionally, strange ingredients (hen? What part of the hen, exactly?). However, in terms of computational creativity, the system performs quite well—coming up with novel ingredient listings that use underrepresented ingredients in the inspiring set.

The use of strange ingredients, or strange amounts of ingredient, come from a lack of feasibility checking on behalf of PIERRE. Part of this feasibility would be knowledge of how the ingredient bill might taste after it's been prepared. The use of a tiny amount of bacon may push the ingredient listing more towards a more novel set of ingredients, but doesn't change the recipe much in terms of flavor. Modeling this knowledge would help allow PIERRE to avoid at least some of it's strange mistakes.

Recipe 2 Exotic beefy bean

Ingredients:

2.2 cups - pinto bean
1.09 pounds - ground beef
1.6 cups - white onion
1.16 cups - diced tomato
1.13 cups - water
1.11 cups - chicken broth
0.77 cup - vegetable broth
0.63 cup - chile sauce
2.74 ounces - pork sausage
4.51 tablespoons - salsa
3.39 tablespoons - stewed tomato
1.43 ounces - chicken thigh
2.5 tablespoons - olive oil
1.09 ounces - hen
0.34 whole - red bell pepper
1.25 tablespoons - lentil
1.16 tablespoons - chopped tomato
2.87 teaspoons - red onion
2.03 teaspoons - garbanzo bean
1.65 teaspoons - cannellini bean
0.26 slice - bacon

Directions: Combine ingredients and bring to boil. Reduce heat and simmer until done, stirring occasionally. Serve piping hot and enjoy.

Figure 4: An example recipe generated by PIERRE

Computational Creativity in the Culinary Arts

[7] describe a system that attempts to generate novel and interesting salads. Much like PIERRE, the authors were only interested in ingredient lists and how they perform in a computational creativity context. Much like PIERRE, this generator also starts with an evaluation set scraped from the World Wide Web. The scrape included ingredient lists and average user-assigned scores. After stripping ingredient lists down to ingredient types (so, ¼ lb of chopped carrots" was converted to "carrot"), the authors used a compliment network augmented with the flavor pairing information in [1]. When an ingredient wasn't represented in the flavor network, it was accommodated with mean value imputation. To combat selection bias in online recipe repositories, much like PIERRE, several randomly generated recipes were created and assigned a low score. Unlike PIERRE, additional human judgment was used to only allow the least palatable examples to be added to the dataset.

After being encoded, the authors then computed 4 centrality features of the ingredient: betweenness, degree, PageRank and eigenvector. These centrality features were augmented with community features in the network, similar to [50]. The authors then trained a classifier on the recipe feature vectors. In order to evaluate a newly generated recipe, the author's use the following formula:

$$f(r) = \frac{\sum_{w \in W} s(w) - \sum_{l \in L}(5 - s(l))}{5|R'|}$$

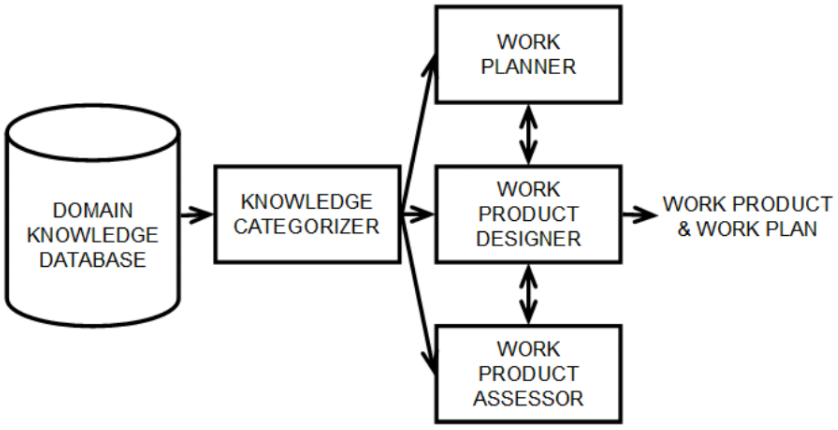

Figure 5: Chef Watson System Architecture. Chef Watson starts with an engineered categorized domain knowledge database. The designer generates new ideas, the assessor evaluates those ideas and the planner calculates which methods to use to realize these ideas. [53]

where $W \subseteq R'$ is the set of recipes that the classifier deems 'worse' than the the recipe r, $L \subseteq R'$ is the set of recipes that the classifier deems 'better' than r, $s(r)$ is the rating the recipe got on the World Wide Web (or 1 for the curated 'bad example' set).

To actually go about creating new recipes, the authors propose a GA, but found the initial randomly generated seed population to be highly performant, according to their evaluation function. This makes the classifier or evaluation scheme suspect—if a random sample all scores highly, then how good is the scheme? The system shows how insights from the recipe recommendation systems can aid in recipe generation, but does not do so in a convincing way.

Chef Watson

Chef Watson is a large-scale recipe generator built on IBM's Watson platform. Chef Watson is general purpose, creating recipes for a large number of dishes from cocktails to pasta salads. Chef Watson focuses mostly on ingredient selection, but does generate complete recipes.

Chef Watson starts by generating all possible ingredient bills from a set of constraints (both learned from data and user provided), then evaluates each potential bill with a set of quality and novelty evaluators. Once a bill scores high enough, Chef Watson calculates portions of each ingredient with a distributional conformance method, and generates recipes steps with a subgraph composition algorithm. The time durations or efforts of atomic steps are estimated by solving the inverse problem from data on complete recipes [36]. The system architecture is described in figure 5.

Chef Watson is a large piece of software, each box in figure 5 has a large amount of algorithmic complexity. It is out of scope for this advancement document to spend

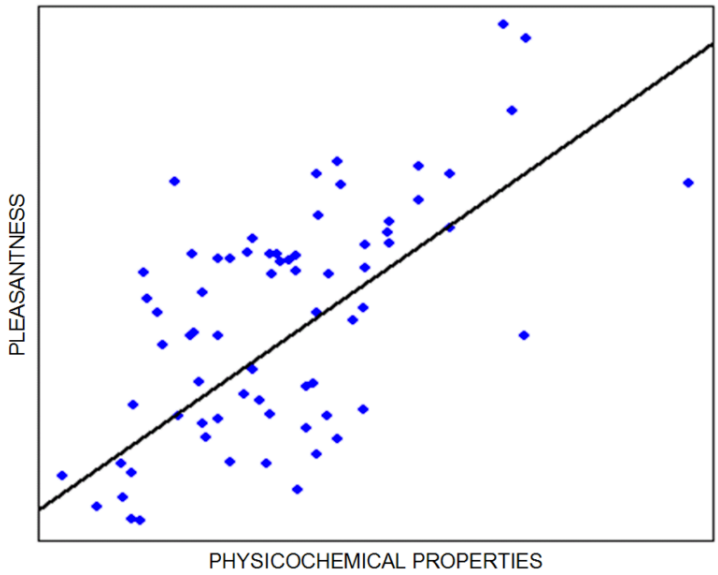

Figure 6: Chef Watson's model of odor pleasantness based on psychophysical properties of chemical compounds. The data points in this image are individual chemical compounds, vertical axis is the human-perceived pleasantness of those compounds and the horizontal axis is the learned combination of physicochemical features.

a large amount of time on every facet of Chef Watson, but there is more detail in [36]. I'd like to focus on Chef Watson's model of flavor.

To evaluate how well a particular ingredient bill will be received, Chef Watson uses a pair of quality metrics (odor pleasantness and food pairing) and a single novelty metric, bayesian surprise [53]. Chef Watson calculates odor pleasantness from a model learned on the labeled dataset from Haddad et al [16]. The dataset is a pleasantness score for 70 different compounds, with various properties of the compounds being features (things like topological polar surface area, heavy atom bond count, etc). Multiple linear regression, with model selection based on the smallest amount of prediction error, was run for each potential feature to reduce dimensionality of the dataset. A final linear regression was run on the smaller number of features to create the final model. This model is shown in figure 6[53].

Chef Watson does use a well-developed idea of the flavor pairing hypothesis, considering positively ingredients that share many compounds or ingredients that share very few, depending on the target cuisine of the recipe that Chef Watson is generating.

Miscellaneous Flavor Models

Shao et al. propose another set of quality / novelty metrics for an artificial chef to use, focused on individual preferences [41]. The quality of an ingredient bill is modeled as the distance an ingredient bill has from a ingredient that an individual

enjoys (such as beef). The metrics also use a personalized novelty metric, that takes the user's social network into account. Although potentially powerful, some model components are non-trivial to figure out how to implement into a system.

Conclusions

Most recipe generators in the wild today focus on a concept of 'recipe quality' without trying to define what that might mean. Both the salad recipe generator and PIERRE use supervised learning approaches to build classifiers that learn ingredient relationships to high star ratings on an online recipe repository, but this learned knowledge is vague.

The vagueness shows up when recipes learned in this fashion attempt to ground out and actually undergo taste tests. In my own prior work with cocktail recipe generation [34], asking participants to give a general Likert-like quality rating for evaluating a cocktail proved to be awkward. Participants were generally looking for different flavor sensations in the cocktails. This was brought to light in a Gimlet (a standard, International Bartender Association approved cocktail) being given both the lowest and the highest rating by different participants.

As shown in neuroscience and psychophysical research, taste is highly subjective. However, we can generally agree on some particular qualities of a flavor profile while differing on an overall value judgment. For example, in general, people find lemons sour. However, they differ greatly on if this is a good or bad quality.

Therefore, it seems to be a logical next step to try and build a model that can capture a taste profile of a particular dish, and leave the value judgment of that profile to a human.

Chapter 3: A Model of Taste

There is one critical component missing in several of the recipe generators and recommender systems discussed earlier in chapter 2: a more precise computational model of taste. Although Chef Watson and the Flavor Pairing Hypothesis make some initial headway, they both give more generic recommendations on how a prepared food item tastes:

- Chef Watson only deals with external olfactory response in terms of 'pleasantness', and uses that as a proxy for flavor pleasantness from the insight that external olfaction is a primary component to flavor perception.
- The flavor pairing hypothesis is too simplistic, and sometimes makes faulty predictions. Even when treated as a two tailed phenomenon, it only speaks to maximizing an ambiguous 'quality' of food pairings, and not to how those pairings actually taste.

Part of this difficulty has been that the human sense of taste has many complex interactions and components. However, chefs have plied their art for centuries, flavor science and sensory taste perception are active research fields, and a computational model can be built from uncovered insights.

Karen Page and Andrew Dornenburg outline a simple linear combination for flavor perception in their book The Flavor Bible. Flavor is the sum of taste, mouthfeel, aroma and what is perceived "by the other senses — plus the heart, mind and spirit" [33]. This chapter will focus on a model that will cover taste. Chapter 5 will look at modeling the more ephemeral qualities of heart, mind and spirit.

The Taste Profile

I define a taste profile as a vector that defines how food tastes. As mentioned in related work, the potential sensory input that goes into banana flavor is complex: not only do the five classical taste components (salty, bitter, sweet, savory, sour) play a part, but also mouthfeel (temperature, astringency, piquancy, creaminess), sight (color, shape), sound (pitch), aroma (pungency, pleasantness, chemesthesis), and more personal qualities (heritage, memory) all contribute to make something taste the way it does[42].

It is not currently known how much each of these parameters plays into the full perception of flavor, however, common wisdom is that as much as 80% of flavor perception is based on smell [9, 33]. But, smell is more complicated than just sniffing. Olfaction occurs externally (sniffing) as well as internally (odorants rising up through the back of the throat to trigger smell receptors [42]).

Therefore, data sources that could provide information for building flavor profiles do not cover all possible parameters for a complete profile. Hence, we only work with a taste profile, which covers the five classical taste sensations that come from receptors on the tongue (salty, bitter, sweet, savory and sour).

When interviewing professional chefs, Karen Page and Andrew Dornenburg found they focus on four classical parameters (bitter, sweet, salty, sour) and claim that, "the essence of great cooking is to bring these four tastes into balanced harmony" [33], so focusing on these sensory qualities has some classical foundation.

As seen in the work presented in the Food Quality and Preference Journal, there is a library of work done on just mapping ingredients to flavor qualities such as sweet, sour, bitter, salty, savory, piquant, astringent, etc. In addition, one can look at recipe comments on websites to find flavor descriptions of particular recipes. However, accessing the flavor information embedded in recipe comments is a tricky natural language processing problem, which makes comments difficult to mine.

More directly, the Yummly recipe API[6] allows for accessing Yummly's recipe repository based on ranges in several flavor parameters. These parameters are labeled 'salty', 'sweet', 'meaty', 'sour', 'bitter', 'piquant' where 'meaty' is synonymous to 'savory' and 'piquant' is synonymous with 'spicy'. Each of the qualities is given a floating-point value from 0 to 1, which allows for searching based on a range of flavor qualities.

More importantly, Yummly's massive size (over a million recipes) and large variety of data sources (70), are large enough to attempt to learn a model of taste for sets of ingredients. There are some shortcomings with using Yummly:

- A lack of authenticity. Yummly's API has been used in academic research in the past [21, 56], but, to date, there are no studies testing the accuracy of Yummly's flavor information.
- The provided flavor parameters do not cover the full spectrum of flavor perception that humans actually sense.

Given all of this, it seems reasonable to propose an initial taste profile as 5-dimensional vector—the amount of sweetness, saltiness, savoriness, sourness, and bitterness of the food. In summary, a taste profile is a five dimensional real-valued vector, where all dimensions range from 0 to 1. Each dimension corresponds to the amount of a taste quality in a food item, with the relevant taste qualities being the sourness, the bitterness, the savoriness, the sweetness and the saltiness of a food item or combination of food items.

Modeling Taste

An obvious first pass at a model of taste would be to take individual ingredients and assign them a taste profile. This is difficult, as most potential data sources deal with combinations of ingredients, and it is difficult to break those ingredients apart and analyze how each of them contribute to the end result. The Yummly API provides a score for the amount of sweetness, saltiness, savoriness, sourness, bitterness and

[6] https://developer.yummly.com/

spiciness for various ingredient bills, but does not provide scores for an individual ingredient. The solution is to look at maps of lists of ingredients to taste profiles.

Because we have a large set of data at hand, we can attempt to leverage machine learning to figure out what this function might be. We can think about this in terms of a multi-target regression problem: what is the real-valued taste profile (a set of response variables) for a set of input ingredients (a set of measured variables)?

There is some reason to believe that several parts of the taste profile will be correlated. Experiments have shown that people have trouble distinguishing sour and bitter [32], and it would be unsurprising to see the two dimensions correlated in the data set. If we don't believe that the output should be correlated (initial statistical tests on a training set don't show any correlations, for example), then it may be better to treat this as a multi-variate regression problem and train an independent model for each dimension of the taste profile.

There are many different strategies in machine learning to handle multi-target regression [5]. Nearly all of them perform supervised or semi-supervised learning. The training set is a set of ingredient lists from Yummly that have taste profiles associated with them. We can keep the input feature vectors small by hierarchically encoding ingredient type. A particular brand of strawberries, for example, can be referred to as "strawberry", for the encoding rather than the particular brand.

We propose to use a Neural Net (NN) to perform this learning task. NNs have shown good results for domains where the feature space has complex topography and may even be disjoint [19]. NNs have also been used in other recipe generation systems.

We need to use a slightly different network topology than has been used in other recipe generation systems—whereas [30] had a single target, we have five. [6] describes a few experiments with multi-target regression NNs (referred to as multi-in, multi-out [MIMO] NNs), detailing how to set up the problem with two popular feed-forward neural networks, multi-layer perceptron (MLP) and radial basis networks (RBN). Specialized NNs also exist for regression problems, both with single outputs and multiple outputs. Generalized Regression Neural Networks (GRNNs) [48] are NNs for regression problems that are based on the RBN. GRNNs are useful due to training speed (the network is trained in a single pass, rather than iteratively [24]), but need very fine tuning of the smoothing parameter, which often requires many training rounds. In addition, GRNNs represent each input vector in the training set as a node in their hidden layers, so the network can grow very large with a large potential input space.

Figure 7 goes over the MLP topology for multiple inputs and multiple outputs. Cross Validation (CV) is often used to find an optimal machine learning models. Standard practice is to use 10-fold CV for finding optimal MLP models [2].

Machine learning techniques often struggle with 'long-tail' distributions [8], where plenty of data exists for a small number of examples, but the vast majority of examples have very few data points. As hinted at in other recipe generators, online

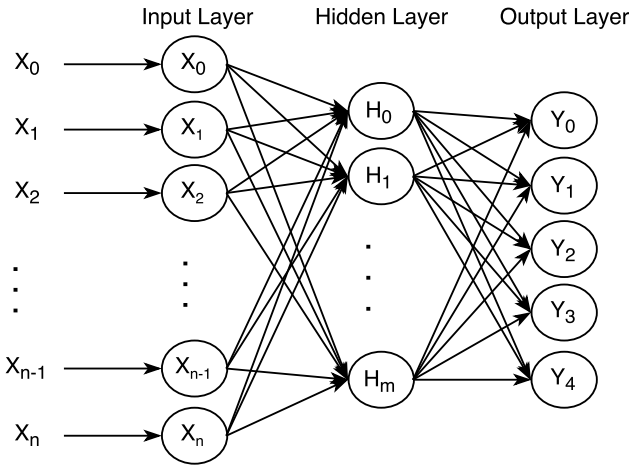

Figure 7: Values in the output layer correspond to a taste profile of the encoded ingredient list from the input layer. The activation function of the output layer is the identity function (all output nodes get activated) and the square error is the loss function.

recipe databases tend to skew heavily towards particular combinations, leaving a vast majority of potential, useful examples in the dark.

We would expect a model trained on data that has a long-tail distribution to have the following properties:

- The model performs very well for a small part of the potential ingredient combination space.
- The potential ingredient combination space containing lots of examples of a few ingredient combinations and then very few examples of the remainder of the space.
- The model, overall, performs poorly (but very well in restricted parts of the space).

If this is the case, then we can move to a more robust set of features. Rather than using sets of ingredients as features for input into a machine learning model, we can shift to using sets of nutrients known to make up those ingredients, and then look at nutrient sets as inputs to the model.

This is more likely to avoid long-tail problems because chemical information generalizes better across novel ingredient combinations. If the model has never seen bananas and blueberries together before, but it has seen strawberries and chocolate, a model trained on just ingredients will be unable to generalize. However, a model trained on chemical components will know that bananas and strawberries share several many chemical compounds (acidic lipids, most sugars, water) and chocolate and blueberries also share several common sugars.

This will, naturally, blow up the dimensionality of the search space. However, the explosion will be less intense than expected due to the fact that food items share many similar compounds and that many compounds function similarly. Furthermore, deep learning techniques have shown to perform well in fine-grained high dimensional spaces, by slowly transforming the input into a lower-dimensional, coarser-grained representation [3].

Let us take a moment to talk about how we can cluster food compounds to reduce the dimensionality of this space. There are four basic molecules that compose food items: water, lipids, carbohydrates and proteins[27]. Some of these categories need to be broken down further. There are many types of lipids, for example, so the category is too broad to be useful. Lipids need to be broken down into true fats (triglycerides) and emulsifiers (diglycerides and monoglycerides). True fats need to be broken down further, into saturated fatty acids, monounsaturated fatty acids and polyunsaturated fatty acids in order to capture important differences in taste[27].

The previous paragraph implies the use of expert knowledge to cluster. We can also use machine learning techniques, such as k-means clustering, to find similar clusters. Other dimensionality reducing techniques, such as PCA (finding which features contribute most to variation over the space, and then only considering those features) can be used. Regardless, there are several potential avenues to make the nutrient space tractable for machine learning.

Information for this model can come from databases on chemical food composition, such as the Volatile Compounds in Food database[7], or the USDA database on chemical food composition[8], as well as research performed in the food science community on the chemical makeup of foods ([38] is a good review).

However, even with the more general dataset of chemical knowledge, the distribution in the training set may still have a long tail. It is possible that the taste model will need to be augmented with case-based knowledge extracted from experts. We can consider using a case based reasoning system to augment the learned model.

Evaluation

The machine learning portion of the taste model can be evaluated via typical ML procedures. As a supervised ML algorithm, we can use a 70/30% training / evaluation split to check it's accuracy.

As a taste profile is defined in terms of real values and not categorical or ordinal values, we can establish a distance metric between the 'true' profile and the profile the model returned. General practice is to use a mean squared error metric, although [5] gives several other potential equations.

Overall, the model (and the data driven evaluation) allow us to finally start attempting to capture something that current recipe recommender systems and generators

[7] http://www.vcf-online.nl/

[8] http://ndb.nal.usda.gov/ndb/foods

do not: how a food item actually tastes, in actionable terms. Due to the limitations of data sources, as well as the limitations of the current understanding of taste and flavor in general, this model still needs to be domain specific, but the methodology should hold regardless of culinary domain, from smoothies to braised meats.

Chapter 4: Mischaire, A Smoothie Generator

The most obvious application of the taste model is in a recipe generator. As dish classes work well as domains, the model should allow us to create a pasta, salad or burger generator. I chose smoothies as the subdomain to investigate with the taste model, due to several useful properties:

- Although consumption temperature does alter the flavors of various food items[10], heat is the driving force behind all chemical reactions that change the chemical structure (and by extension, flavor properties) of foods. Limiting the amount of heat keeps the search space tractable.
- Smoothies are fairly easy to prepare in a uniform fashion, which aids human evaluation.
- Smoothies are delicious.

Mischaire is a recipe generator that uses the model of taste from chapter 3 to aid in creating new smoothie recipes. Given a target taste profile to find, Mischaire uses a goal-driven genetic algorithm to come up with new recipes that try to fit the taste profile. Additionally, Mischaire accepts a set of constraints, defining parts of the potential smoothie space that the system can search over.

Genetic algorithms (GAs) have shown good performance in optimization problems, especially in spaces with many local maxima or minima, because they optimize from a set of starting points, rather than a single location [11]. Furthermore, GAs have a long history of being used for generative methods [51], even in the context of recipes [31].

Constraint-based methods have found a home in generative methods, mostly in the subdomain of procedural content generation (PCG), through the use of solvers to find all possible artifacts that fit a certain set of constraints. Pioneering work was performed by Adam Smith in Variations Forever and Refraction games [45, 44] and Gillian Smith in the mixed-initiative level creator Tenagra [47]. The general paradigm when using constraint methods is to allow a user/designer to specify some constraints, and show a sample set of artifacts that can be generated based on those constraints. The designer then narrows the constrained space to exclude the examples he/she does not like, and more examples are shown under the newly constrained space. This process continues until the designer is satisfied with examples.

Mischaire mixes the two styles of generation. The software assumes that some design targets are best described in terms of goals, “I want artifacts to have properties very similar or exactly like X, Y or Z” as well as constraints, “Generated artifacts should

never have properties T or Q and have value F in between 0 and 6". Due to the known difficulty English speakers have in expressing taste [32], taste makes a good GA optimization target. Users of our software probably don't know enough about the complex interactions of taste to specify constraints on a taste profile, but may be able to make some rough stabs at it (more salty than sweet, not that sour, etc). These rough stabs (think of slider positions) are more befitting a goal than a constraint. Other parts of a recipe, such as prep time, number of steps, cookware required or even nutritional content, is easier to express with constraints.

There are two ways we can incorporate constraint programming and genetic algorithms:

- Constrain first, search over remainder: given a set of constraints, generate all potential smoothies that fit those constraints (via a solver). Then, use the GA to find the most performant smoothie in this restricted space.
- Constraints as part of the GA loop: Constraints are handled as part of the GA. This has been studied in the numeric optimization community under a number of forms—constraints can be transformed into penalties given to population members that don't fit the constraints, constraints can be iteratively applied though time to thin the population down, all constraints can be applied at every iteration of the the GA to immediately remove all members of the population that don't fit those constraints (for review, see [28]).

Due to the large potential space of not only ingredient combinations, but all the ways to prepare various ingredients, we propose that constraints should be part of the GA loop (the second option). The delay in waiting for a solver to find all potential recipes that fit a set of constraints is too long. As in other work, a recipe is modeled well by an acyclic directed graph.

Formally, a recipe graph $G < V, E >$ is a directed, acyclic graph where V is the set of vertices and E is the set of directed edges connecting those vertices. Vertices come in two types: ingredient vertices and step vertices (so $V = Iv \cup Sv$, where Iv is the set of ingredient vertices and Sv is the set of step vertices). Edges in the graph are directed and show how particular ingredients flow through the graph. So, an edge $e \in E$ can be defined by a start vertex i and an end vertex j, such that $\forall e \in E$ there exists a $v_i, v_j | v_i \in V, v_j \in Sv$ and $v_i \rightarrow v_j$.

This graph will end up being a tree, any path from any vertex in Iv will terminate in a final step vertex in Sv, even if that step is 'serve dish'. The leaves of the tree are all in Iv, all other vertices are in Sv. Figure 8 shows an example.

Therefore, we need to consider how to encode a tree structure in a GA. [25, 14, 35] detail three ways to encode a tree in a GA: edge encodings, vertex encodings and edge-and-vertex encodings. Due to the fact that the GA needs to preserve some structure in the trees it generates (Iv nodes can only be leaves, and leaves can only be members of Iv), and that most of the operations we'd want to perform are node

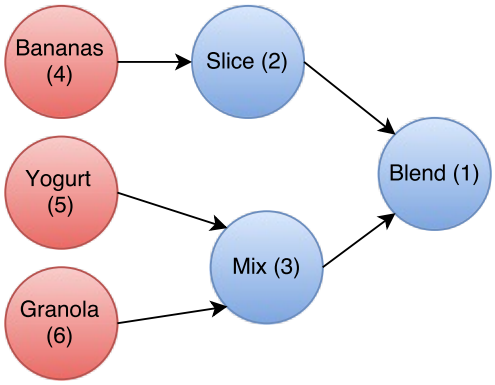

Figure 8: An example of a recipe graph. Vertices are in Iv are ingredients, in red. Directed edges show how ingredients flow through step vertices (Sv), in blue.

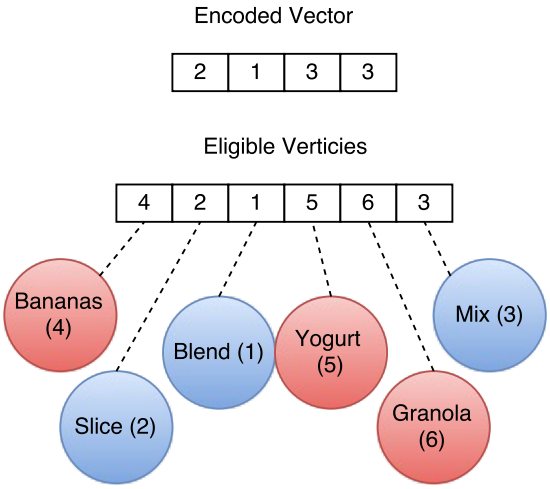

Figure 9: The vertex encoding of the same recipe graph. The encoded vertexes define an ordering on the topology of the graph

based (new steps, ingredient swaps, etc.), we'll use a vertex encoding in[25].

A tree (T) can be encoded in a GA as a Prüfer vector (P) as follows:

1. Let vertex i be the lowest labeled leaf node of T. Let vertex j be incident to vertex i; append j to the end of P
2. Remove vertex i and edge e_{ij}, which connects vertices i and j
3. Go back to step 1 until there is only one edge in T, P is obtained.

Figure 9 shows the same tree using this vertex encoding. We can decode a Prüfer vector (P) and its set of eligible vertices (P') into a tree (T). The eligible vertices

list is all the vertices in the graph before encoding, in wind-up order. The decoding works as follows:

1. Let vertex i be the lowest eligible node of P' and vertex j be the first element of P. If $i \neq j$, add the edge from i to j in T, remove vertex i from P' and vertex j from P
2. Repeat step 1 until no elements are left in P
3. For the last two remaining vertices in P', add the edge between them in T

This encoding is useful because it is topology preserving—as long as no nodes from Iv are in the Prüfer vector, they will continue to be leaves after the mutation operation. Prüfer vectors define an ordering of internal nodes.

Our tree encodings, therefore, are a Prüfer vector (P) and its set of eligible vertices (P'). We've already seen why the Prüfer vector makes a good representation, but we can also define mutation and crossover operations on the set of eligible vertices that will keep the important topology of the ingredient graph in tact. In addition, we can define a mutation operation that changes the graph topology while keeping the nodes in Iv leaves. Referred to in [25] as the shifting Prüfer vector operation, this tweaks inner tree topology.

Crossover and mutation operations are easy in the leaf case—we can edit leaf all we want without changing tree topology, from swapping to adding in new leaves. Swapping or adding in new step nodes is trickier, but still possible. By carefully tweaking the labeling of the nodes and adding / removing numbers to the Prüfer vector, we can add in new branches to the tree.

This gives us a genotype, and a set of operations to tweak the genotype. The phenotype is just the recipe tree that the phenotype describes. We now need to look for ways to evaluate the phenotype for fitness. We can now go back to one of the first questions posed in this chapter: how will we handle a mixed strategy of constraints and objectives?

Objective Functions / Heuristics / Goals

However, before we get to this point, we need to discuss what potential constraints Michaire can have, and that requires us to talk a little more about the vertices in the tree, Iv and Sv.

Ingredients consist of a general name, and some physical properties (such as potential allergen categories, dietary restriction constraints, etc). Steps consist of some basic action verb (such as 'mix', 'blend' or 'chop') and some additional properties (such as time the step takes, the number of utensils required, etc). User constraints can come in two categories: numerical constraints on the full recipe (such as total recipe time) or categorical constraints on particular nodes (never use blueberries or 'blending', for example).

One constraint that will probably not be user specified is one of validity—all ingredient nodes should have a reasonable path to the root node. We can get a list of valid ingredient paths by scraping recipe repositories online, and looking at cook books.

So, an optimal recipe:

1. Minimally violates user constraints
2. Contains a minimal amount of invalid edges and nodes
3. Gets as close as possible to some target taste profile

We base our objective function on [29], augmented with the additions listed in [28]. We start, like many others, by noting that an objective function for a GA can be considered as a minimization problem, where we are trying to optimize some function $f(X)$ where X is an n sized set of real-valued variables. Variables in X are also subject to an additional set of constraints:

$$g_j(X) \geq 0, for j = 1, ..., q$$

$$h_j(X) = 0, for j = q + 1, ..., m$$

Also note that q and m are not dependent on n; the number of constraints on the variables has no relation to the number of variables. We can then frame GA as a search to minimize $eval(X)$, where:

$$eval(X) = f(X) if no constraints are violated, f(X) + penalty(X) otherwise$$

We can construct $penalty(X)$ from a set of functions $f_j$where the function f_j measures the violation of the jth constraint as:

$$f_j(X) = max0, g_j(X), if 1 \leq j \leq q; |h_j(X)|, if q + 1 \leq j \leq m$$

Essentially, we frame the objective function of a GA as a process that is trying to minimize an objective function that takes in a set of constrained real valued parameters. If any of the parameters violate a set of constraints, a penalty is added to the objective function.

Some of the constraints discussed earlier were categorical (such as containing ingredients of a particular type). We can encode these as large constant penalties, although the actual numbers chosen will depend on experimentation. For derived real values, such as number of steps or time taken or number of calories, we can define functions that can traverse the graph to come up with the correct information.

The core objective function, $f(X)$ is taste profile distance, as specified in the evaluation section of chapter 3. Some constraint functions ($g_j(X)$ or $h_j(X)$) can be a

maximum or minimum number of nodes in the graph (a good representation of complexity) or the required inclusion or exclusion of particular step or ingredient nodes (users allergic to nuts can specify that a graph should never have an ingredient node in the class 'nut', for example).

From [29], we evaluate the fitness of a member of the population of the GA with:

$$eval(X, \tau) = f(X) + \frac{1}{2\tau} \sum_{j=1}^{m} f_j(X)^2$$

where τ is some temperature, given a starting temperature of τ_0 and decreased with each new population until less than some final temperature τ_f. To use this temperature based evaluation, we require a few things:

- The initial population consisting of copies from a single initial potential recipe graph that satisfies all constraints on the problem.
- Each new population starts as a set of copies from the best solution from the previous population

[29] gives standard values for the starting temperature, the final temperature and a step size for each new temperature. This particular strategy is desirable because it seems to perform well on a variety of problems. It also does not require very many parameters, unlike some other formulations of GA optimization formalizations (such as [18]).

It is highly important to note that this strategy does not ensure that, when the GA is finished running, the final result does not violate some constraints. For this solving strategy user constraints are penalties transformed into soft constraints that the algorithm makes a best-effort to uphold.

In [37], the authors describe an algorithm that uses a GA to solve a numeric optimization problem with nonlinear constraints that will return a solution which violates a minimum number of constraints. The algorithm ensures that $eval(X) < eval(Y)$ if X violates less constraints than Y regardless of how well $f(X)$ or $f(Y)$ perform. This seems more optimal for a space that has hard, inflexible constraints embedded in it, like food and recipes do (if a user is allergic to pine nuts, there is no give on that constraint boundary. The resultant recipe must not have pine nuts in it).

However, this requires finding the least good, yet still satisfying all constraints, solution before starting the GA. This round of searching may be intractable in our space, and therefore, we propose to start with the temperature algorithm. If initial evaluations of the above are deemed unsuitable, we can shift strategies and identify constraints that must hold true, search for the minimum scoring recipe that holds those constraints true, and then start searching.

Our current scheme does require an initializing search, as we need to find a valid recipe before we can start our GA, but the search is more general. We need to find a tractable solution, not the minimum tractable solution. However, if this search proves too difficult (as graph search is not a particularly easy problem), [18] provide a framework that allows for random initialization. This particular GA uses levels of violation (with increasing penalties as level goes up) on each of its variables to help steer the algorithm towards solutions that fit inside of the constraint boundaries. However, as the authors admit, the amount of parameters that need to be set before starting this GA can be staggering (5 variables with 4 levels of volition require setting 45 parameters)!

With the current strategy, (using temperature in our objective function), we can still report which constraints we've violated and by how much.

Evaluation

The first round of evaluations should lie on how well the GA stays in various constraint spaces. Being able to characterize the GA in terms of which constraints it does well on vs which ones it tends to violate is a good first metric for quality.

We can look at how well the generator covers the potential smoothie space by performing an expressive range analysis on it. Unlike the taste model, where we could compare and contrast predicted results of a prediction on an evaluation set, here we may not have a smoothie recipe to compare to. What if a user has the GA search for a recipe that has an inordinate number of steps, because that's what the user wants?

Looking at how many different ingredient combinations the model considers when given a particular taste profile and set of constraints is still useful, and can show what parts of the generative space get ignored under certain conditions.

Finally, algorithm characterization is an interesting form of analysis, but are the smoothies actually pleasing to a human consumer? A similar study to the one performed in [34] and [7] will be performed, where average users will try some smoothies designed by humans and some smoothies designed by Michaire. In both cases, we'll be looking to see if the participants report the predicted flavor profile of the smoothie.

A final note about the proposed user study, unlike some past work, we are not looking to say that the smoothie recipes are creative. Although rough in some cases, we're looking for actual grounding out of the flavor data set. Do Yummly's flavor profiles match up with the taste reported by a random sample of people? Do Michaire's smoothies reflect the flavor profiles provided for by participants? If both these sources have error, is the error the same? What can the error tell us about the generator?

We'd also like to perform the same study with taste experts. Average users and experts have been shown to come from separate evaluation populations, and keeping the studies separate can reveal key differences between expert and average perception

and bias.

Chapter 5: Heart, Mind and Spirit

There is a part of cooking that is not captured by taste profiles or recipe trees optimized to dirty the minimum amount of bowls. Food, and the reasons why we choose dish over the other, is tightly intertwined with ephemeral qualities of memory, heart and spirit. These are the trickiest qualities of cooking to address algorithmically, but to ignore them is to miss what brings people together for Thanksgiving, Passover or even Taco Tuesday.

Carolyn Korsmeyer writes in Making Sense of Taste, her investigation into the aesthetics of flavor, that, "Foods qualify as symbolic and meaningful in a host of ways" [22]. Carolyn applies Nelson Goodwin's theories about symbols in art to food, with a focus on how food represents, exemplifies, and expresses artistic properties. Carolyn argues that food satisfies all of Goodwin's symbol types and also has the property of 'relative repleteness', a condition where "comparatively many aspects of a symbol are significant" [15]. Carolyn does not argue that food is art, but that argument is more in the realm of aesthetics than AI.

The relevant parts are the first three symbol categories: food can represent something else, usually performed by making the dish look like something else (a gingerbread house or a red velvet cake baked to look like a puppy), food can exemplify properties (posses a general property, apples are red, therefore they exemplify red), and food can express properties (apples possess wickedness because the evil Queen tricks Snow White with an apple in the fairy tale Snow White, Goodman refers to this as 'metaphorical exemplification').

This realm of exemplification, both metaphorical and physical, gets at the heart of culinary artistry and creativity. Korsmeyer spends the last two chapters in Making Sense of Taste taking a whirlwind tour of what qualities food exemplifies by looking at representations of food in visual and written art. Deborah Lupton (in her book Food, The Body and The Self), writes about meaning-making in food from a poststructuralist perspective. Lupton's work is couched in sociology rather than Korsmeyer's backing in philosophy and art history, but both authors agree that meaning in food is highly personal and based on local culture.

Lupton focuses on how meaning with food items is tethered to meaning around the body and meaning around the self. However, her book contains plenty of references to qualities food exemplifies. These qualities can come through a dish or ingredient's color, "The colors white, green and yellow in particular are valorized in foods: white stands for purity, innocence, refinement, green symbolizes freshness, nature, rural spaces, while yellow is redolent of sunshine, the open air and culturally valued gold" [26]. These qualities can also be inherent to the ingredient itself, "Milk, for example, stands as a restorative, a symbol of purity, the innocence of a child, natural goodness, calm strength and reality" [26]. These qualities can also come from when a particular food item is eaten (for example, the traditional dishes of an American Thanksgiving have special significance and meaning because of their strong association with this

holiday), as well as personal experiences and memories.

This advancement proposes trying to capture these sorts of information in a semantic network. The proposed semantic network is a set of concepts, which, in this case, can be ingredients, dishes, physical properties, taste profiles and more nebulous cultural concepts like purity, refinement or strength. The relationships between these concepts follow from Korsmeyer and Goodwin: the food items in the semantic network can exemplify the physical properties in the semantic network and express the cultural concepts.

Formally, items in a semantic network can be considered as facts F, where:

$$f =< p_1, r, p_2 >, f \in F$$

p_1 and p_2 are concepts in the network and r is the relation between them. Concepts fall under three classes: ingredients, physical properties and cultural properties. Relations also have three classes:

1. Representation, which relates ingredients to things they physically resemble.
2. Exemplification, which relates ingredients to physical properties they have.
3. Expression, which relates ingredients or physical properties to the cultural concepts they express.

It's possible that existing semantic networks like ConceptNet[9] contain the required entities and links, however, it is likely that such an semantic network will need to be built. This is mostly due to the focused subdomain for the semantic network that we're considering—ConceptNet proposes to be a general purpose, common-sense reasoning resource, which may or may not focus on food, dishes and social meaning around food enough to be useful.

It is also important to note that the semantic network proposed here would not be complete, and would obviously reflect the biases and viewpoints of a particular culture or even a particular person. However, such a network can enable another level of reasoning over a proposed recipe—what cultural meaning does it invoke?

Creativity

Computational Creativity has long argued over how to formalize and operationalize creativity: is it a set of properties that a creative work has [40]? A set of properties that a creative process has [4]? A set of properties inherent in a creative multi-agent generative system[39]? Depending on which camp you fall into, your definition of creativity changes, as well as how you attempt to measure and evaluate it.

This makes creativity very difficult to evaluate. Perhaps it's better to look at purposeful recipe creation based on exemplification and expression as an expressive

[9] http://conceptnet5.media.mit.edu/

ability that current recipe generators lack. A sufficiently developed symbolic recipe generator may generate recipes that fall alarmingly close to art (according to previously discussed aesthetic philosophy), and focusing on this aspect leads to a generator that seems to exhibit creative behavior.

Incorporating the proposed semantic network of food symbolism into Michaire requires two things: all the ingredients or dishes that Michaire can create be represented in some way in the semantic network, and that Michaire have some way to search the symbol space. Representing Michaire's dishes is easy enough. The component ingredients should already be encoded in the network, and we can consider a dish to have some of the cultural meaning that comes from its ingredients. Furthermore, taste profiles can be encoded in the network as physical properties that ingredients express.

To search over this the semantic network, we'll add another term to Michaire's objective function, $n(X)$. $n(X)$ is the coherence of the set of cultural properties that are expressed by the taste profile and component ingredients. Trying to invoke too many different cultural concepts at once is likely to be confusing and dilute meaning. For each similar cultural symbol among the ingredients, $n(x)$ decreases. This function will take a lot of hands-on tweaking to nail down—should expressed symbols from exemplified properties count for less than core expressed symbols?

Other interesting concepts include trying to invoke certain set of cultural symbols, and penalizing the generator for every symbol it finds not in that set. This sort of purposeful generation allows us to generate towards some sort of meaning. We can come up with a set of symbols that have importance for a particular place, then generate a set of recipes around that symbol set.

The result would be a book of recipes for a place—a local cookbook of meaning.

Evaluation

Adding the proposed semantic network as an additional structure in addition to the model of taste, and changing Michaire's evaluation function is interesting, but there is a hard wall to climb: how do we evaluate it?

As stated earlier, meaning in food is highly shifting and personal—there is no objective standard to evaluate the semantic network against. It isn't even a certainty that everyone who interacts with this version of Michaire will agree with what the generator believes is it metaphorically exemplifying in a particular recipe. In fact, being blunt about Michaire's metaphorical intentions can actually make a user think less of the generator if their beliefs about food metaphor don't align.

However, we can show the recipe along with other things that also try to reference the same metaphorical concept. By adding a surface renderer that juxtaposes the generated recipes with images or short bits of text that also surface the same underlying metaphorical content, we can generate associations that a viewer might not agree with, but may still find the end result pleasing or useful (A user may not understand why a unicorn (a common symbol of purity) is doodled in the margins

in a recipe using milk, but it doesn't make make them balk against the reference as much as if we had told them directly, "the milk means purity"). Such 'annotated' recipes could be considered aesthetic in their own right, and we can build a recipe book from them for showcasing or home use.

We can also open up the semantic net to playful exploration—allowing users to try and select a set of symbols and see what the AI comes up with, or allowing users to try and force the AI into awkward positions ('let's have it try to generate a sour recipe that expresses safety') can be a fun. This sort of interaction paradigm has been used by Microsoft for How Old Bot[10], an AI that tries to guess how old you look, and was summarily pushed to the limit by users who would don masks to 'fake' the bot out, or ask it to identify the ages of their dogs.

Overall, the semantic net starts to push Michaire towards trying to understand ephemeral qualities of food, giving it a way to access and conceptualize more personal and social concepts around food. This opens up new avenues for interactions with food and AI, between artistic juxtaposition and college around recipes to allowing playful exploration of semantic space.

Chapter 6: Conclusion and Timeline

This advancement does not go into all aspects of flavor—the human perception of it is not understood well enough to completely operationalize, and unlike other human senses (such as sight or sound) easy sensor-based proxies currently don't exist. However, current recipe generators either completely abstract flavor away, or deal with it on a very high level. To improve upon existing techniques, we need a computational model of taste.

The model is learned from existing recipes online, and maps a multidimensional taste profile to an input vector of ingredients. This problem looks similar to multi-target regression, and we use a MIMO neural network to try and learn this relation.

This taste model enables an example recipe generator, Michaire, to recipes that fit a user specified taste profile in a particular domain. Michaire is also novel in the context of generative methods, as it encodes constraints into a genetic algorithm. This is a requirement for recipe generation, as it needs to accommodate real world requirements like food allergens or limited prep time.

Finally, we look at some more ephemeral qualities of food and recipes, taking a strong focus on cultural symbolism and personal meaning that gets imbued into particular ingredients and recipes. We propose a mapping of cultural symbols to ingredients (and by extension recipes). By searching over this more fantastic space, we can allow Michaire to generate recipes that capture more of the 'spirit of the ingredients'.

The dissertation will take two years. I expect that answering each question will take around the same amount of time: the first 8 months will be spent developing and

[10] https://how-old.net/

evaluating the taste model. Progress milestones include evaluating the effectiveness of the machine learned taste model. Papers on the taste model can find a fit in computational creativity conferences, as well general AI and flavorist venues.

The second 8 months will be spent developing Michaire. At first, we'll focus on using the taste model in a recipe generator for smoothies, and allowing for additional constraints to be specified as part of its generation process. As the mixing of constraint satisfaction and genetic algorithms is fairly unique in the generative methods community, this work can find a fit in Procedural Content Generation or more general generative methods workshops or conferences.

The final eight months will be focused on building the food-symbol semantic network, and arming Michaire with it. This final work is highly intertwined with social practice in food, and may find a better fit in an art project setting. The final deliverable will be a cookbook of a place, incorporating all the discussed aspects of recipe generation into a final piece.

Overall, this work is a strong contribution to the emerging multidisciplinary realm of food science, social practice around food and computer science. We think that there is a space for AI to assist human chefs in the kitchen, and allow for the playful exploration of both potential flavor and potential meaning in food.

References

[1] Yong-Yeol Ahn, Sebastian E Ahnert, James P Bagrow, and Albert-László Barabási. Flavor network and the principles of food pairing. Scientific reports, 1, 2011.

[2] Tim Andersen and Tony Martinez. Cross validation and mlp architecture selection. In Neural Networks, 1999. IJCNN'99. International Joint Conference on, volume 3, pages 1614–1619. IEEE, 1999.

[3] Yoshua Bengio. Learning deep architectures for ai. Foundations and trends® in Machine Learning, 2(1):1–127, 2009.

[4] Margaret A Boden. What is creativity. Dimensions of creativity, pages 75–117, 1994.

[5] Hanen Borchani, Gherardo Varando, Concha Bielza, and Pedro Larrañaga. A survey on multi-output regression. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 5(5):216–233, 2015.

[6] Oscar Claveria, Enric Monte, Salvador Torra, et al. Multiple-input multiple-output vs. single-input single-output neural network forecasting. Technical report, University of Barcelona, Research Institute of Applied Economics, 2015.

[7] Erol Cromwell. Computational creativity in the culinary arts. In Proceedings of the 46th ACM Technical Symposium on Computer Science Education, pages 715–715. ACM, 2015.

[8] Ernest Davis and Gary Marcus. Commonsense reasoning and commonsense knowledge in artificial intelligence. Communications of the ACM, 58(9):92–103, 2015.

[9] Maurits De Klepper. Food pairing theory: A european fad. 2011.

[10] Jeannine Delwiche. The impact of perceptual interactions on perceived flavor. Food Quality and preference, 15(2):137–146, 2004.

[11] David B Fogel. An introduction to simulated evolutionary optimization. Neural Networks, IEEE Transactions on, 5(1):3–14, 1994.

[12] Peter Forbes and Mu Zhu. Content-boosted matrix factorization for recommender systems: experiments with recipe recommendation. In Proceedings of the fifth ACM conference on Recommender systems, pages 261–264. ACM, 2011.

[13] Gijs Geleijnse, Peggy Nachtigall, Pim van Kaam, and Luciënne Wijgergangs. A personalized recipe advice system to promote healthful choices. In Proceedings of the 16th international conference on Intelligent user interfaces, pages 437–438. ACM, 2011.

[14] M Gen and R Cheng. Genetic Algorithms & Engineering Design. Weily, 1997.

[15] Nelson Goodman. Languages of art: An approach to a theory of symbols. Hackett publishing, 1968.

[16] Rafi Haddad, Abebe Medhanie, Yehudah Roth, David Harel, and Noam Sobel. Predicting odor pleasantness with an electronic nose. PLoS Comput Biol, 6(4):e1000740, 2010.

[17] Kristian J Hammond. Chef: A model of case-based planning. In AAAI, pages 267–271, 1986.

[18] Abdollah Homaifar, Charlene X Qi, and Steven H Lai. Constrained optimization via genetic algorithms. Simulation, 62(4):242–253, 1994.

[19] William Y Huang and Richard P Lippmann. Neural net and traditional classifiers. In Neural information processing systems, pages 387–396, 1988.

[20] Felix Kamieth, Andreas Braun, and Christian Schlehuber. Adaptive implicit interaction for healthy nutrition and food intake supervision. In Human-Computer Interaction. Towards Mobile and Intelligent Interaction Environments, pages 205–212. Springer, 2011.

[21] Hanna Kang-Brown and Jacob Kang-Brown. Mapping the flavors of new york city. Contexts, 13(3):62–70, 2014.

[22] Carolyn Korsmeyer. Making Sense of Taste: Food and Philosophy. Cornell University Press, 2014.

[23] Miriam Kort, Ben Nijssen, Katja van Ingen-Visscher, and Jan Donders. Food pairing from the perspective of the 'volatile compounds in food'database. In Expression of Multidisciplinary Flavour Science: Proceedings of the 12th Weurman Symposium, Interlaken, Switzerland, pages 589–592, 2010.

[24] Savita G Kulkarni, Amit Kumar Chaudhary, Somnath Nandi, Sanjeev S Tambe, and Bhaskar D Kulkarni. Modeling and monitoring of batch processes using principal component analysis (pca) assisted generalized regression neural networks (grnn). Biochemical Engineering Journal, 18(3):193–210, 2004.

[25] Chi-Chun Lo and Wei-Hsin Chang. A multiobjective hybrid genetic algorithm for the capacitated multipoint network design problem. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 30(3):461–470, 2000.

[26] Deborah Lupton. Food, the Body and the Self. Sage, 1996.

[27] Harold McGee. On food and cooking: the science and lore of the kitchen. Simon and Schuster, 2007.

[28] Zbigniew Michalewicz. Genetic algorithms, numerical optimization, and constraints. In Proceedings of the sixth international conference on genetic algorithms, volume 195, pages 151–158. Morgan Kaufmann, San Mateo, CA, 1995.

[29] Zbigniew Michalewicz and Naguib Attia. Evolutionary optimization of constrained problems. In Proceedings of the 3rd annual conference on evolutionary programming, pages 98–108. Citeseer, 1994.

[30] Richard G Morris, Scott H Burton, Paul M Bodily, and Dan Ventura. Soup over bean of pure joy: Culinary ruminations of an artificial chef. In Proceedings of the 3rd International Conference on Computational Creativity, pages 119–125, 2012.

[31] Richard G Morris, Scott H Burton, Paul M Bodily, and Dan Ventura. Soup over bean of pure joy: Culinary ruminations of an artificial chef. In Proceedings of the 3rd International Conference on Computational Creativity, pages 119–125, 2012.

[32] M O'Mahony, M Goldenberg, J Stedmon, and J Alford. Confusion in the use of the taste adjectives 'sour'and 'bitter'. Chemical Senses, 4(4):301–318, 1979.

[33] Karen Page and Andrew Dornenburg. The Flavor Bible: The Essential Guide to Culinary Creativity, Based on the Wisdom of America's Most Imaginative Chefs. Little, Brown, 2008.

[34] Johnathan Pagnutti and Jim Whitehead. Generative mixology: An engine for creating cocktails. In Proceedings of the Sixth International Conference on Computational Creativity June, page 212, 2015.

[35] Charles C Palmer and Aaron Kershenbaum. An approach to a problem in network design using genetic algorithms. Networks, 26(3):151–163, 1995.

[36] Florian Pinel, Lav R Varshney, and Debarun Bhattacharjya. A culinary computational creativity system. In Computational Creativity Research: Towards Creative Machines, pages 327–346. Springer, 2015.

[37] David Powell and Michael M Skolnick. Using genetic algorithms in engineering design optimization with non-linear constraints. In Proceedings of the 5th International conference on Genetic Algorithms, pages 424–431. Morgan Kaufmann Publishers Inc., 1993.

[38] Jorge Regueiro, Nelia Negreira, and Jesús Simal-Gándara. Challenges in relating concentrations of aromas and tastes with flavour features of foods. Critical reviews in food science and nutrition, (just-accepted):00–00, 2015.

[39] Rob Saunders and John S Gero. The digital clockwork muse: A computational model of aesthetic evolution. In Proceedings of the AISB, volume 1, pages 12–21, 2001.

[40] R Keith Sawyer. Explaining creativity: The science of human innovation. Oxford University Press, 2011.

[41] Nan Shao, Pavankumar Murali, and Anshul Sheopuri. New developments in culinary computational creativity. In Proceedings of the Fifth International Conference on Computational Creativity, 2014.

[42] Gordon M Shepherd. Smell images and the flavour system in the human brain. Nature, 444(7117):316–321, 2006.

[43] Yuka Shidochi, Tomokazu Takahashi, Ichiro Ide, and Hiroshi Murase. Finding replaceable materials in cooking recipe texts considering characteristic cooking actions. In Proceedings of the ACM multimedia 2009 workshop on Multimedia for cooking and eating activities, pages 9–14. ACM, 2009.

[44] Adam M Smith, Erik Andersen, Michael Mateas, and Zoran Popović. A case study of expressively constrainable level design automation tools for a puzzle game. In Proceedings of the International Conference on the Foundations of Digital Games, pages 156–163. ACM, 2012.

[45] Adam M Smith and Michael Mateas. Variations forever: Flexibly generating rulesets from a sculptable design space of mini-games. In Computational Intelligence and Games (CIG), 2010 IEEE Symposium on, pages 273–280. IEEE, 2010.

[46] David V Smith and Robert F Margolskee. Making sense of taste. Scientific American, 284(3):32–39, 2001.

[47] Gillian Smith, Jim Whitehead, and Michael Mateas. Tanagra: A mixed-initiative level design tool. In Proceedings of the Fifth International Conference on the Foundations of Digital Games, pages 209–216. ACM, 2010.

[48] Donald F Specht. A general regression neural network. Neural Networks, IEEE Transactions on, 2(6):568–576, 1991.

[49] Martin Svensson, Kristina Höök, and Rickard Cöster. Designing and evaluating kalas: A social navigation system for food recipes. ACM Transactions on Computer-Human Interaction (TOCHI), 12(3):374–400, 2005.

[50] Chun-Yuen Teng, Yu-Ru Lin, and Lada A Adamic. Recipe recommendation using ingredient networks. In Proceedings of the 4th Annual ACM Web Science Conference, pages 298–307. ACM, 2012.

[51] Julian Togelius, Georgios N Yannakakis, Kenneth O Stanley, and Cameron Browne. Search-based procedural content generation: A taxonomy and survey. Computational Intelligence and AI in Games, IEEE Transactions on, 3(3):172–186, 2011.

[52] Mayumi Ueda, Mari Takahata, and Shinsuke Nakajima. User's food preference extraction for personalized cooking recipe recommendation. In Workshop of ISWC, pages 98–105, 2011.

[53] Lav R Varshney, Florian Pinel, Kush R Varshney, Debarun Bhattacharjya, Angela Schoergendorfer, and Yi-Min Chee. A big data approach to computational creativity. arXiv preprint arXiv:1311.1213, 2013.

[54] Liping Wang and Qing Li. A personalized recipe database system with user-centered adaptation and tutoring support. In ACM SIGMOD Ph. D. workshop on Innovative database research (IDAR). Citeseer, 2007.

[55] Liping Wang, Qing Li, Na Li, Guozhu Dong, and Yu Yang. Substructure similarity measurement in chinese recipes. In Proceedings of the 17th international conference on World Wide Web, pages 979–988. ACM, 2008.

[56] Longqi Yang, Yin Cui, Fan Zhang, John P Pollak, Serge Belongie, and Deborah Estrin. Plateclick: Bootstrapping food preferences through an adaptive visual interface. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, pages 183–192. ACM, 2015.

[57] Lijuan Yu, Qing Li, Haoran Xie, and Yi Cai. Exploring folksonomy and cooking procedures to boost cooking recipe recommendation. In Web Technologies and Applications, pages 119–130. Springer, 2011.