Computational Creativity in the Culinary Arts

Erol Cromwell
Davidson College
Davidson, NC 28035, USA
ercromwell@davidson.edu

Jonah Galeota-Sprung
Museum of Mathematics
New York, NY 10010, USA
galeota-sprung@momath.org

Raghuram Ramanujan
Davidson College
Davidson, NC 28035, USA
raramanujan@davidson.edu

Abstract

This work examines the potential for computational creativity in the culinary arts. In particular, we present the outcome of our efforts to create an artificial chef that produces novel salad recipes with limited human assistance. Our system was designed in two steps: first, we constructed a statistical model to rank recipes. Then, we experimented with various search algorithms to explore the salad recipe space and discover novel ingredient combinations. Surprisingly, we discovered that the top ranked recipes from a randomly generated population were already of high quality, obviating the need for search. To validate the quality of our automatically generated salads, we conducted a blind taste test pitting three computer-designed and three human-designed salads against each other. We discovered that the best performing computer salad was competitive with the human generated salads.

1 Introduction

Can machines be creative? This is a question that is central to the field of Artificial Intelligence, and whose history can be traced all the way back to Alan Turing's seminal paper that introduced the "Imitation Game" (Turing 1950). In the past few decades, impressive strides have been made in the field of computational creativity in areas as diverse as music composition (Cope 1992; 2005; Quintana et al. 2013), the visual arts (Cohen 1995; Colton 2012), storytelling (Turner 1993; Birnbaum et al. 2014), humor (Binsted and Ritchie 1994), scientific discovery (Schmidt and Lipson 2009), and mathematical problem-solving (McCune 1994; 2005). In this paper, we explore artificial creativity in a domain that has been mostly overlooked — cooking.

In particular, we investigate the feasibility of creating a program that autonomously generates novel and flavorful salad recipes. In this preliminary work, we limit our attention to salads to avoid having to model the complex chemical transformations that the cooking process can introduce (though this modeling problem offers a rich setting for future investigations). Within this restricted domain, our problem is this: given that ingredients can be combined in an exponential number of ways to produce recipes, what is an effective strategy to navigate this space and discover combinations that surprise and delight the average human palate?

Our approach comprises two steps. First, we train a discriminative classifier that, given two recipe descriptions, predicts which one is better. We use this pairwise recipe comparator as a heuristic to guide a local search algorithm whose goal is to construct new ingredient combinations. Surprisingly, our experiments reveal that very little search is actually necessary to successfully create novel recipes. Indeed, we found that our baseline approach — namely, choosing to combine ingredients that are selected uniformly at random — is startlingly successful. The salads generated in this manner were received favorably by anonymous human subjects in a blind taste test. This suggests that ingredient combinations that make for good salads are distributed rather densely throughout the ingredient space; unlike in many other combinatorial search spaces, finding good salad recipes is not a problem of looking for a needle in a haystack. Phrased differently: creating novel salads does not seem to require particularly great creative leaps.

The remainder of this paper is organized as follows. In section 2, we discuss some related work on using computational approaches to understand the structural properties of cooking recipes. Sections 3, 4, and 5 describe our data acquisition, processing, model construction, and search methodology. Section 6 describes and analyzes our simulation and taste test results. We conclude with some suggestions for future work in section 7.

2 Related Work

While there is little prior work that investigates computational recipe generation, there is an extensive body of literature devoted to the problem of *recommending* recipes. Typically, these systems suggest recipes for users to try based on parameters such as their past assigned ratings (Forbes and Zhu 2011; Freyne and Berkovsky 2010b; 2010a), their browsing history (Ueda, Takahata, and Nakajima 2011), and ingredient availability and nutritional needs (Kamieth, Braun, and Schlehuber 2011; Shidochi et al. 2009). Crucially, however, none of these systems *generate* novel recipes that weren't already part of a human curated database.

More recently, Ahn et al. (2011) studied the structural properties of recipe networks by using graph theoretic approaches. Specifically, they constructed a weighted, undirected graph where individual ingredients formed the nodes and edges connected vertices that shared some minimal number of flavor compounds. By analyzing this flavor network, the authors were able to extract interesting insights about the nature of ingredient pairing in various world cuisines. For example, they discovered that North American and Western European dishes tend to favor recipes where ingredients share many flavor compounds; East Asian and Southern European recipes, on the other hand, tend to combine ingredients with disparate flavor compositions.

Teng, Lin, and Adamic (2012) similarly used a combination of network analysis and machine learning techniques to build a model that, given a pair of recipes, could predict which one would receive a higher "star" rating. The dataset used in this study was scraped from allrecipes.com, an online aggregator of user-contributed recipes. The authors' model was trained on pairs of recipes: each training example was a vector of features that described the two recipes, with a binary label indicating which of the recipes was better (i.e., had received a higher star rating from online users). The features used to describe the recipes were derived by applying dimensionality reduction techniques to two graphs: a complement network and a substitution network. In the former, ingredients form the nodes, with edges connecting two nodes only if they co-occur in some minimal number of recipes. In the latter, ingredients share an edge only if they can be effectively substituted for each other in a recipe. Using this approach, the authors created a model that could correctly identify the higher-rated recipe from a pair 79% of the time.

Building on this prior work, we ask the question: how can these insights into the structure and nature of ingredient networks be used to *create* novel recipes? We note that this work is most similar in spirit to that of the "cognitive cooking" effort at IBM (Bilow 2014). However, most of IBM's work in this area remains proprietary; very little has been published or otherwise released into the public domain.

3 Data Collection and Cleansing

We began by scraping the ingredient lists for salad recipes with at least five user reviews from allrecipes.com (1802 in all). In the remainder of this paper, we will refer to this master set of recipes as R. We cleaned each ingredient name by eliminating amount descriptors ($3/4$ cup, 1 lb., a pinch, etc.) and preparation information (chopped, baked, sliced, etc.), and by changing the word stem to its singular form. So, for example, "$1/2$ lbs. sliced carrots" was converted into "carrot". Ingredients with multiple names were manually renamed to a single, consistent name (for example, "scallion" and "green onion" were both remapped to "green onion"). Finally, following the approach of Teng, Lin, and Adamic, we removed from consideration any ingredient that did not appear in at least 6 unique recipes. This eliminated any ingredients that were rare, branded, misspelled, or not edible. After these preprocessing steps, we were left with a list of 319 unique ingredients. We denote this set of ingredients by I.

4 Ranking Recipes

We built a discriminative classifier that, given two recipes, could predict which of the pair was better, i.e., received a higher star rating. Our first approach used a binary vector of length $|I| = 319$ to represent each recipe. The setting of each of the bits corresponded to the presence (1) or absence (0) of a specific ingredient in that recipe. A training example consisted of the concatenation of the vectors representing two different recipes, with a binary label indicating whether the first or the second recipe received a higher star rating. Following the approach of Teng, Lin, and Adamic, we were selective in deciding which recipes were paired and presented as training examples to the machine learning system. Specifically, only recipe pairs that had a cosine similarity greater than 0.2 were used as training examples. These pairs, on average, shared 2.7 ingredients in common. The similarity criterion was used to ensure that the learner did not attempt to generalize from salad pairs that were overly dissimilar (which may happen if the salads are drawn from, say, different cuisines). The set R was partitioned into a 60:20:20 train-validation-test split. All recipe pairs with a similarity score above 0.2 in the training fold were supplied as training examples to a boosted decision stump learner (Friedman 2002) with 8 terminal nodes. The parameters of the learner (namely, the number of boosting iterations) were tuned based on performance on *all* pairs of recipes in the validation fold. The accuracy of the learned model was evaluated on all pairs of recipes in the test fold by measuring the percentage of cases in which the model correctly picked out the better recipe. With this simple bit vector recipe representation, our predictive model achieved an accuracy of 52%.

To improve on this performance, we decided to abandon the bit vector representation and instead combine the network-based approaches of both Teng, Lin, and Adamic, and Ahn et al.. We constructed a complement network for the ingredients in I, as outlined in Teng, Lin, and Adamic (2012), and used the flavor compound data provided by Ahn et al. (2011). There were, however, several ingredients in I for which flavor compound information was missing. We used mean value imputation to accommodate these ingredients. Further, due to selection bias, the vast majority of recipes from allrecipes.com have a rating between 4 and 5 stars, out of 5. This is unsurprising, since people do not knowingly contribute poor recipes to online collections. However, this created an interesting conundrum: the purpose of the classifier was to help us automatically winnow out poorly performing recipes that would be generated by a search process. However, this search process was likely to generate *many* poor recipes, from a part of the recipe space that the learner would not have seen, were it to be trained strictly on elements drawn from R. In other words, the target distribution for the classifier would not match the distribution of examples on which it was trained. To remedy this, we supplemented the set R with recipes that were generated by choosing between 6 and 12 ingredients uniformly at random from the set I. We carefully curated these randomly generated recipes, eliminating those that seemed even remotely palatable; others, that were obviously bad ingredient pairings, were assigned a 1-star rating and added to the set R. Figure 1 shows a couple of examples of these 1-star combinations. In all, 1894 such 1-star recipes were added to our initial recipe set, creating an augmented recipe set R', of size 3696.

Recipe 1: Butter, condensed tomato soup, lemon zest, mustard, strawberry, tomato sauce

Recipe 2: Baby arugula, potato, corn chip, cucumber, flour, orange flavored jello, pineapple, watermelon, white wine

Figure 1: Examples of randomly generated 1-star recipes.

We extracted network centrality features to describe each recipe based on the complement and flavor compound graphs. Following the approach of Teng, Lin, and Adamic, we computed four centrality measurements for each ingredient: betweenness, degree, PageRank, and eigenvector. Each ingredient was thus associated with a vector of eight centrality scores — its betweenness in the complement network, its betweenness in the flavor network, its degree centrality in the complement network, its degree centrality in the flavor network, and so on. A recipe's centrality features were computed by summing up the centrality vectors of its ingredients.

Network community features were also extracted using the prescription of Teng, Lin, and Adamic. Starting with the adjacency matrix W of a network, its rank-k approximation $W_k = U_k \Sigma_k V_k^T$ was computed using the singular value decomposition. A recipe's full ingredient bit vector $\vec{b}$ was transformed into a lower-dimensional vector $\vec{b}' = \Sigma_k^{-1} V_k^T \vec{b}$ that encoded the community information in the network. This dimensionality-reduction process was carried out with both the complement and the flavor networks to produce two separate network community vectors, that were then concatenated together. The appropriate value for k was selected based on the classifier's performance on the validation set. We found that $k = 60$ was optimal for the complement network, while $k = 80$ worked best for the flavor compound network. By representing each recipe as a vector of centrality and community features, and with the augmented training set, we were able to construct a predictive model that achieved an accuracy of 82% on the held-out test data. This surpasses the accuracy of the model constructed by Teng, Lin, and Adamic (79.2%).

5 Scoring New Recipes

In order to evaluate a newly generated recipe, we used the following scoring function f that maps any given recipe r to a real number:

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

Here, $W \subseteq R'$ is the set of recipes that the learned classifier deems are "worse" than the recipe r, and $L \subseteq R'$ is the set of recipes that the classifier deems are "better" than r. The function $s(k)$, where $k \in R'$, denotes the star rating of recipe k on allrecipes.com (or the value 1, in case k was one of the artificially generated recipes). Intuitively, f awards a high score to a recipe r if it consistently "beats" highly rated recipes from R'. Similarly, it awards a low score to a recipe that consistently loses out to lowly rated recipes in R'. Thus, every newly generated recipe is scored in a consistent manner by running pair-wise comparisons against a standardized set of curated recipes (namely, the set R').

6 Results

Simulation Results

In our final step, we investigated effective search strategies to explore the salad recipe space so as to discover high-scoring, novel ingredient combinations. As a baseline, we generated 800 random recipes, each with seven, eight, and nine ingredients, and ranked them with our scoring function f. We chose these particular recipe lengths since the average length of a recipe in R is eight. Our initial plan was to use these 2400 recipes as the seed population for a genetic algorithm that would seek to iteratively improve on them. Surprisingly, we discovered that the top ranked recipes from this seed population were already of high quality. The average score of the top 20 generated recipes was 0.39 (as computed using the scoring function f), with a maximum score of 0.56. In comparison, the average score of the top 20 salads from the set R was 0.09, with a maximum score of 0.38. As such, attempting to improve on the seed population using hill-climbing methods proved to be unnecessary. Simply building thousands of random recipes and retaining the top ranked ones was sufficient. However, while our program believed that these top recipes were "good", we needed to validate this with one final metric: human tastebuds.

Taste Test Results

To verify the quality of our computationally generated salad recipes, we organized a blind human taste test. For this test, we selected three recipes at random from the 20 top ranked salads in R. We supplemented these with three computer-generated recipes. The latter were randomly drawn from a set of twenty recipes, each of which was the top ranked one in an independently generated population. All of the salads were then prepared by the staff in the Davidson College Dining Services. The human designed salads were prepared by following the instructions on allrecipes.com. For the computer-generated recipes, the chefs were instructed to strictly adhere to the following guidelines:

- Every ingredient listed was to be used.
- No extra ingredients were to be added.
- Every ingredient was to make "its presence felt", i.e., its flavor was to be clearly discernible.

The six salads were placed in a randomly permuted order, with the ingredient lists placed next to the respective salads. Volunteers were invited to sample the six salads and filled out a questionnaire that asked them to rate the taste and the novelty of the ingredient combinations of each salad on a Likert scale from 1 to 5. The prompts used on the questionnaire are shown in figure 2. Participants were not told which of the salad recipes were human-designed and which were computer-generated, but in the spirit of the Turing Test, were invited to make their best guess.

Taste Rating: *1: Strongly disliked it, 2: Disliked it, 3: Indifferent, 4: Liked it, 5: Strongly liked it*

Novelty Rating: *1: Extremely common combination (I've seen this before), 2: Somewhat common (I've seen variants of this before), 3: Fairly common, with some interesting twists , 4: Somewhat uncommon (It's quite different from anything I've seen before), 5: Extremely uncommon combination (I would have never thought to combine those)*

Figure 2: Human taste test questionnaire prompts.

Here are the six salads that were used, with the letter H/C indicating the source of the recipe (human and computer respectively).

Salad 1 (H): Paprika, poppy seed, sesame seed, spinach, strawberry, vegetable oil, white sugar, white wine

Salad 2 (C): Cherry, chive, granny smith apple, mushroom, onion powder, pine nut, salsa, salt

Salad 3 (C): Bok choy, feta cheese, green onion, mango, radish, red onion, rotini pasta

Salad 4 (H): Bacon, celery, egg, mayonnaise, onion, pepper, salt, red potato

Salad 5 (H): Bacon, broccoli, mayonnaise, red onion, raisin, sunflower seed, white sugar, white whine vinegar

Salad 6 (C): Black-eyed pea, chive, egg, granny smith apple, mustard seed, peanut, roma tomato, salad dressing

In all, 62 volunteers sampled our salads. The highest scoring computer salad (Salad 3) received a mean rating of 3.72, which was competitive with the worst performing human salad (Salad 5) that received a rating of 4.06 as seen in figure 3. However, the score for Salad 3 was noticeably lower than the score of the best performing human salad (Salad 1, 4.56). The computer-generated salads did, however, consistently rank higher in novelty than the human-designed salads (figure 4). The two top-rated computer salads by novelty, Salads 2 and 6, scored significantly higher than the most novel human one (Salad 1). Only 9 participants (14.5%) correctly guessed the appropriate labeling for every salad. Furthermore, participants mistook the highest rated computer salad for a human designed one 56% of the time. Overall, our computer-generated recipes were competitive in taste with the human recipes, but did score lower. However, in novelty, the computer-generated recipes excelled against the human ones.

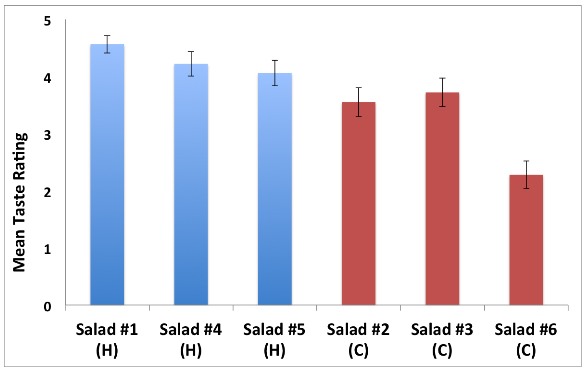

Figure 3: The mean taste rating of each salad.

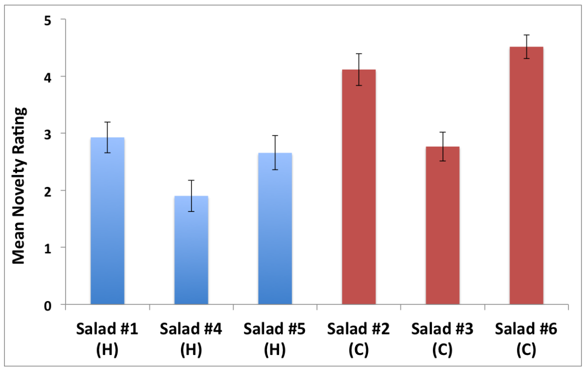

Figure 4: The mean novelty rating of each salad.

7 Conclusions

This paper investigated the possibility of building machines that could exhibit creativity in the domain of cooking. We improved upon prior work on creating predictive models for ranking recipes. We then used this ranking function as a search heuristic for discovering novel ingredient combinations. Surprisingly, we discovered that there was little need for search in finding novel, tasty salad recipes.

We envision multiple ways in which this work can be extended. One could try alternate machine learning approaches, particularly regression techniques — rather than attempting to predict pairwise recipe winners, we could instead directly predict the star rating of a recipe. Further feature engineering could also improve the performance of the system. At present, we ignore many other features that could potentially be relevant, including nutritional information, ingredient color, and texture. Our approach also ignores the relative quantities of ingredients, which is a crucial determinant of whether a recipe will succeed. In the future, we would also like to expand the scope of the project to a wider class of recipes — soups, drinks, desserts, etc.

8 Acknowledgements

The authors would like to thank Tim Chartier and Ýmir Vigfússon for useful discussions. We would also like to thank Dee Phillips and her staff at Davidson College Dining Services for helping organize the taste test and the many volunteers who sampled and rated the salads for this study.

References

Ahn, Y.-Y.; Ahnert, S. E.; Bagrow, J. P.; and Barabási, A.-L. 2011. Flavor network and the principles of food pairing. *Sci. Rep.* 1.

Bilow, R. 2014. How IBM's Chef Watson actually works. Retrieved from http://www.bonappetit.com/entertaining-style/trends-news/article/how-ibm-chef-watson-works on Oct. 1, 2014.

Binsted, K., and Ritchie, G. 1994. An implemented model of punning riddles. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 1)*, AAAI '94, 633–638. Menlo Park, CA, USA: American Association for Artificial Intelligence.

Birnbaum, L.; Hammond, K.; Allen, N.; and Templon, J. 2014. System and method for using data to automatically generate a narrative story. US Patent 8,688,434.

Cohen, H. 1995. The further exploits of aaron, painter. *Stanford Hum. Rev.* 4(2):141–158.

Colton, S. 2012. The painting fool: Stories from building an automated painter. In McCormack, J., and dInverno, M., eds., *Computers and Creativity*. Springer Berlin Heidelberg. 3–38.

Cope, D. 1992. Computer modeling of musical intelligence in emi. *Computer Music Journal* 16(2):pp. 69–83.

Cope, D. 2005. *Computer Models of Musical Creativity*. The MIT Press.

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

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

Freyne, J., and Berkovsky, S. 2010b. Recommending food: Reasoning on recipes and ingredients. In *Proceedings of the 18th International Conference on User Modeling, Adaptation, and Personalization*, UMAP'10, 381–386. Berlin, Heidelberg: Springer-Verlag.

Friedman, J. H. 2002. Stochastic gradient boosting. *Comput. Stat. Data Anal.* 38(4):367–378.

Kamieth, F.; Braun, A.; and Schlehuber, C. 2011. Adaptive implicit interaction for healthy nutrition and food intake supervision. In Jacko, J., ed., *Human-Computer Interaction. Towards Mobile and Intelligent Interaction Environments*, volume 6763 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 205–212.

McCune, W. W. 1994. Otter 3.0 reference manual and guide. Technical report, Argonne National Laboratory, Argonne, IL.

McCune, W. W. 2005. Prover9 and mace4. http://www.cs.unm.edu/~mccune/prover9/.

Quintana, C. S.; Arcas, F. M.; Molina, D. A.; Rodriguez, J. D. F.; and Vico, F. J. 2013. Melomics: A case-study of ai in spain. *AI Magazine* 34(3):99–103.

Schmidt, M., and Lipson, H. 2009. Distilling Free-Form Natural Laws from Experimental Data. *Science* 324(5923):81–85.

Shidochi, Y.; Takahashi, T.; Ide, I.; and Murase, H. 2009. 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*, CEA '09, 9–14. New York, NY, USA: ACM.

Teng, C.-Y.; Lin, Y.-R.; and Adamic, L. A. 2012. Recipe recommendation using ingredient networks. In *Proceedings of the 4th Annual ACM Web Science Conference*, WebSci '12, 298–307. New York, NY, USA: ACM.

Turing, A. M. 1950. Computing machinery and intelligence. *Mind* 59(236):433–460.

Turner, S. R. 1993. *Minstrel: A Computer Model of Creativity and Storytelling*. Ph.D. Dissertation, University of California at Los Angeles, Los Angeles, CA, USA. UMI Order no. GAX93-19933.

Ueda, M.; Takahata, M.; and Nakajima, S. 2011. User's food preference extraction for cooking recipe recommendation. In de Gemmis, M.; Luca, E. W. D.; Noia, T. D.; Gangemi, A.; Hausenblas, M.; Lops, P.; Lukasiewicz, T.; Plumbaum, T.; and Semeraro, G., eds., *SPIM*, volume 781 of *CEUR Workshop Proceedings*, 98–105. CEUR-WS.org.