Recipe Recommendation

Jeremy Cohen, Robert Sami, Aaron Schild, Spencer Tank

May 13, 2014

1 Introduction

We tackled what we call the *recipe interest problem*: given a person's favorite recipes, which other recipes are they likely to be interested in viewing? We imagine that the recipe interest problem might be relevant to a web site such as the recipe aggregator AllRecipes.com. For example, the site might want to build a system to recommend new recipes to users. In addition, the idea of building a model to explain how people like foods is interesting in its own right.

We experimented with content-based, collaborative filtering, and hybrid recommendation techniques to the recipe recommendation problem. We evaluated these approaches qualitatively, by inspecting their recommendations and spotting trends, and quantitatively, by running the algorithms against held-out data. Finally, we built an online recommendation tool available at `http://recipeguru.me` that allows our three most successful approaches to be compared and contrasted.

2 Background and Overview

2.1 Dataset

AllRecipes.com is a popular recipe-sharing web site with a sizeable community of active users. This made it a good data set to use to train a recipe recommendation system.

We scraped all 43K recipes and all 130K users from the site. For each recipe, we downloaded the list of ingredients and the quantity, in grams, of each. For each user, we downloaded the names of the recipes that the user stored in his/her "recipe box"—essentially, a list of the user's favorite recipes on the site.[1]

We discarded all recipes that occurred in fewer than 300 recipe boxes.[2] This left us with 9,546 recipes (consisting of 2,502 unique ingredients), 115,308 users, and 9.6 million binary ratings.

2.2 Recommender Systems

Recommendation systems usually fall into one of two classes: content-based approaches or collaborative filtering approaches. Content-based systems consider only item properties, and recommend items with similar properties to items that the user has liked in the past. Collaborative-filtering systems consider only user ratings; the intuition is that users who share some preferences are likely to share other preferences. Hybrid systems use both item properties and user ratings [3].

3 Approaches

3.1 Content Based Approaches

We developed two content-based approaches to recipe recomendation. Both of these content-based approaches start with a list of feature vectors $\{\mathbf{v}_i\}_i \in \mathbb{R}^d$ for some fixed d. These feature vectors

[1]On AllRecipes.com, users can interact with a recipe in two ways. First, they can give a recipe a numerical rating ($\in \{1...5\}$). Second, they can put a recipe in their recipe box ($\in \{0, 1\}$). Either of these interactions could be used to build a recommendation system. We considered both options carefully and decided to base our collaborative filtering on the recipe boxes. We reasoned that a user giving a recipe only 1 star does not mean that the user is uninterested in the recipe, but rather that the user *is* interested in the recipe, but the recipe turned out badly. Users who know they dislike grilled cheese do not browse through AllRecipes.com giving all "grilled cheese" recipes 1 star. Therefore, we decided that recipe boxes are a better data source for answering the question of recipe *interest*.

[2]Matrix factorization (described later) was taking too much time, so we needed to reduce the size of the matrix.

represent recipes. The user provides a recipe box, i.e. a list of vectors $\{\mathbf{u}_i\}_i \in \mathbb{R}^d$. Content based approaches compute a user profile as the average of the $\mathbf{u}_i$s and recommend its nearest neighbors in $\{\mathbf{v}_i\}_i$ with respect to the cosine similarity metric [3][p. 315]. Given two vectors $\mathbf{x}$ and $\mathbf{y}$, the *cosine similarity* of the two vectors is

$$\frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}||||\mathbf{y}||}$$

We used two different methods for specifying a recipe as a vector in d-dimensional space, which we discuss in the next two sections.

3.1.1 Ingredient Similarity

The *ingredient similarity* approach assumes that a user's preference for a recipe is entirely a function of the recipe's ingredients (and not, for example, the recipe's name or its preparation instructions).

In this approach, each $\mathbf{v}_i$ is a vector of mass fractions of all ingredients in recipe i (including ones that do not appear). The coordinate corresponding to each ingredient is the total mass of the ingredient in the dish divided by the total mass of all ingredients in the dish.[3]

3.1.2 Ingredient + Dish Similarity

The idea behind the *ingredient + dish similarity* approach is that the gist of a recipe is not fully captured by the ingredients alone. A chocolate muffin and a chocolate cake may be composed of the same ingredients in the same proportions, but everybody knows that one is a muffin and the other is a cake. We term this piece of information that is obvious to humans (but missing from the ingredient vector) the *dish.* In the *ingredient + dish similarity* approach, the dish is given weight of its own; to that end, we represent each recipe by its ingredient vector concatenated with an indicator vector that represents the recipe's dish.

We extract the dish of a recipe by simply taking the final word in the recipe's title. It turns out that this simple algorithm produces reasonable assignments of dishes to recipes.

The *ingredient + dish similarity* approach fixes one parameter λ for all recipes. For recipe r, let $\mathbf{u}_r$ be the ingredient-weights vector for r and let $\mathbf{v}_r$ be the indicator vector for r's dish. Then, $\mathbf{w}_r$, the *ingredient + dish vector* representation of r, is the following:

$$\mathbf{w}_r = \text{cat}(\mathbf{u}_r, \lambda \mathbf{v}_r)$$

High values for λ mean that the dish is given high importance compared to the ingredients. We set λ using cross-validation on training data (see Appendix C, Figure 2).

3.2 Collaborative Filtering Approach

Our collaborative filtering approach assumes that recipes are combinations of latent factors. Users have preferences over these recipe factors. A user's preference for a recipe is a function of his preference for the factors in that recipe.

Let U be the number of users and let R be the number of recipes. Let M be a $U \times R$ matrix where M_{ur} is 1 if user u has recipe r in his recipe box, and 0 otherwise. We learn $K = 40$ latent recipe factors by factorizing M into a $U \times K$ matrix and a $K \times R$ matrix:

$$M^{U \times R} \approx W^{U \times K} \times H^{K \times R}$$

Sure enough, this factorization technique produces coherent recipe factors that seem to correspond to real tastes. Appendix A shows, for each factor k, the names of the 5 recipes with the highest coefficients H_{rk} in factor k. There appear to be factors for chicken (0, 4, 13, 16), vegan (2, 9), seafood (3), slow-cooker (5), Superbowl (6), cake (7), cookies (8), salad (11), comfort food (12), cheesecake (18), cocktails (21), bread (23), Indian (27), Thanksgiving (31), fried foods (35), casserole (38), scones (39), and other interpretable food categories.

[3]We experimented with alternative feature vectors for recipes. Specifically, we considered (1) a binary feature vector that encodes whether or not each ingredient occurs in the recipe (in any quantity); and (2) a tf-idf feature vector, in which the ingredient mass is normalized by its mass throughout the entire data set. The mass fractions representation yielded recipe similarities that were intuitively better than binary and tf-idf.

Having factorized $M^{U \times R}$, there are several ways that we could recommend recipes to users. If the user u was part of the training data set, then we could simply take row u in WH to be the predicted preference of user u for each recipe. However, we prefer a method that is extensible to new users. Therefore, we represent each recipe with its column in H (a K-vector), and we compute a profile for user u by averaging all recipes in u's recipe box. There are two ways that we could use u's profile to estimate u's preference for a recipe r. First, we could take the cosine similarity between u's profile and r's column in H. Second, we could take the dot product between u's profile and r's column in H. How do these approaches compare? Recall that the cosine similarity is simply the dot product divided by the product of the magnitudes of both vectors. Since the magnitude of the user profile vector does not change, the only difference between these two approaches, when it comes to ranking recipes in order of u's estimated preference for them, is that the dot product metric takes the global popularity of a recipe into account, while the cosine similarity metric does not. Suppose that there exist two recipes whose recipe-factor vectors are off by a constant multiplier. This means that both recipes have the same breakdown of latent recipe factors, but one is more popular. The cosine similarity metric will treat both recipes the same, while the dot product metric will always rank the more popular recipe higher. We discuss the difference between the performance of these two metrics in the Results section below.

3.2.1 Matrix Factorization

We considered a number of algorithms for matrix factorization, but after trial and error with Python SKLearn, Apache Mahout, Vowpal Wabbit, and other libraries—some of which did not produce interpretable factors, and some of which took too long—we settled on a MATLAB implementation of Non-negative Matrix Factorization (NMF) [2], available at `https://sites.google.com/site/nmftool/`.

The goal of NMF is to decompose a non-negative matrix V into two non-negative low-rank matrices W and H such that:

$$V \approx W \times H$$

We used $D(V||WH)$, the cost function given by Seung, to assess the difference between V and WH. In general, $D(A||B)$ (a metric related to the KL divergence) is defined as:

$$D(A||B) = \sum_{ij} A_{ij} \log \frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij}$$

Seung [1] proved that $D(V||WH)$ is nonincreasing under the following multiplicative update rules for successively approximating W and H:

$$H_{a\mu} \leftarrow H_{a\mu} \frac{\sum_i W_{i\alpha} V_{i\mu}/(WH)_{i\mu}}{\sum_k W_{ka}}$$

$$W_{ia} \leftarrow W_{ia} \frac{\sum_\mu H_{a\mu} V_{i\mu}/(WH)_{i\mu}}{\sum_\nu H_{a\nu}}$$

We ran the multiplicative update algorithm (`nmfrule()` in the nmftool MATLAB package) on a machine with a 2.2GHz processer and 128 GB RAM. The algorithm took 8 hours to complete 2000 iterations on a 9546×115308 matrix.

Note that we also experimented with an alternative NMF algorithm based on alternating least squares [2]. This algorithm (`nmfnls()` in the nmftool package) did not produce a factorization as interpretable as the multiplicative update algorithm.

3.3 Hybrid Approach

We also tried a hybrid approach to recommendation that exploited both item properties and user preferences.

The intuition behind our hybrid model is that recipes are combinations of observed ingredients, and ingredients are in turn combinations of latent "ingredient factors." Users have preferences over these ingredient factors. A user's preference for a recipe is a function of his preference for the ingredient factors in the ingredients in that recipe.

Therefore, instead of factorizing the user-recipes matrix (as above), we factorize the user-ingredients matrix. Let I be the number of ingredients. Let M be the $U \times R$ user-recipes matrix from the collaborative filtering approach. Let N be the $R \times I$ recipes-ingredients matrix, i.e. N_{ri} is the weight fraction of ingredient i in recipe r. We compute the user-ingredients matrix P by multipling M and N:

$$P^{U \times I} = M^{U \times R} \times N^{R \times I}$$

Note that P_{ui} is a measure of user u's preference for ingredient i. Next, to learn $K = 50$ latent ingredient factors, we factorize P into matrices W and H:

$$P^{U \times I} \approx W^{U \times K} \times H^{K \times I}$$

W_{uk} is user u's preference for factor k, and H_{ki} is the weight of factor k in ingredient i.

Finally, we derive a vector representation of each recipe by computing the $R \times K$ recipes-factors matrix A:

$$A^{R \times K} = N^{R \times I} \times H_T^{I \times K}$$

To recommend new recipes to a user, we compute a user profile by averaging all of the rows of A for recipes in his recipe box, then recommend recipes using either the cosine similarity or dot product metrics described in the previous section.

The ingredient factorization really does appear to produce coherent ingredient factors (see Appendix B). For example, there appear to be factors for baked goods (3), granola foods (4), beans (7), bread (8), fish (11), tuna salad (12), beef (13), Indian (15), shellfish (24), chicken (25), drinks (27), barbecue (34), fruit (36, 38), vegetarian (37), Italian (42), Mexican (6, 18, 45), turkey (47), pork (44), pumpkin (49), desserts (14, 21, 48), and other interpretable tastes.

3.3.1 Synonymy and Novelty

One way of thinking about our hybrid approach is that we exploit aggregate user preferences to learn a low-dimensional representation of ingredients (ingredient factors). This dimensionality reduction allows the hybrid approach to overcome the problem of *synonymy* (to borrow a term from the text analysis literature) that our pure content-based approaches suffer from. Suppose that pecans is, in effect, "synonymous" with walnuts—everyone who likes one likes the other. Our content-based approaches will, unfortunately, never recommend a pecan-heavy recipe to a user with only walnuts in his recipe box. In theory, the hybrid approach, however, will reduce pecans and walnus to the same "nuts" factor.

Likewise, the hybrid approach is intended to handle the problem of *novelty* better than our pure collaborative filtering approach. Pure CF cannot handle a new recipe that it wasn't trained on. The hybrid approach, in contrast, is fully equipped to handle new recipes so long as the model has trained on all of the new recipe's ingredients.

4 Results

4.1 Qualitative Evaluation

The *ingredient similarity* approach tends to recommend recipes which are extremely similar in ingredient composition to the user's known preferences. For example, asked for 3 recipes similar to "Irresistable Pecan Pie", the system recommends "Honey Crunch Pecan Pie", "Pecan Pie Bars I", and "Pecan Pie IV." The *ingredient + dish similarity* approach, on the other hand, is more likely to rank "pies" above "pie bars", even if the pie bars are pecan. It is hard to tell whether this is a good thing.

The *recipe factors* approach ignores ingredients and recommends recipes that have latent factors in common with the user's known preferences. For example, asked for three recipes similar to "Irresistable Pecan Pie", the algorithm recommends "Yummy Sweet Potato Casserole", "Perfect Turkey", and "Cranberry Stuffing"—we evidently hit upon a "Thanksgiving" latent factor! When recommendations are delivered using the cosine similarity metric, the algorithm is sensitive to the exact recipes in the user's past favorites; but when recommendations are delivered using the dot product metric, the algorithm tends to recommend the most popular recipes in each factor over and over again. It is hard to tell which is better: the latter performs better at our quantitative evaluation (see below), but it may be true that the former is more desirable in a real system.

The *ingredient factors* approach really only works when the cosine similarity metric is used. Asked for three recipes similar to "Irresistable Pecan Pie", the algorithm recommends "Soft Oatmeal Cookies", "Brown Sugar Pound Cake", and "Sour Cream Coffee Cake." Pecans, oatmeal, brown sugar, and coffee were reduced to the same latent ingredient factor. However, when the dot product metric is used, the algorithm often screws up—for example, it always recommends turkey brine to users who have expressed a preference for baked goods, because the flour in the baked goods and the water in the turkey brine reduce to the same combination of ingredient factors.

4.2 Quantitative Evaluation

We devised the following error metric to compare the above recommendation algorithms. We withheld 10% of all ratings from all users with more than 300 ratings, and trained the algorithms on the remaining ratings data. Then, for each algorithm, for each user, we measured the precision and recall of the top $N = 200$ recommendations.

We also tested a dummy algorithm that recommends recipes at random and a dummy algorithm that recommends recipes in strict order of popularity.

Appendix C shows precisions and recalls for each of our approaches. All of our approaches except for *ingredient factors* with the dot product metric outperformed chance. The best was *recipe factors* with the dot product metric, which also outperformed the naive popularity benchmark. As expected, *ingredient + dish similarity* outperformed *ingredient similarity*; this confirmed our intuition that the dish should carry some weight independent of the ingredients.

5 Conclusion

We experimented with content-based, collaborative filtering, and hybrid approaches to the recipe interest problem. Pure collaborative filtering was the most successful on the evaluation procedure that we devised, but the content-based and hybrid approaches each have their own unique strengths. A better evaluation procedure (less biased by popularity) would be to ask AllRecipes users for the percentage of an algorithm's recomendations that look most interesting to them.

We have shown that Non-negative Matrix Factorization produces interpretable factors for both recipes and ingredients (Appendix A, B).

If we had more time, we would have liked to experiment with different ways to normalize the users-by-ingredients matrix before factorization, in order to get better results with our hybrid approach. We would have also liked to incorporate elements from the recipe directions (such as baking vs. frying) into the content-based approaches.

6 Live Online Demo: RecipeGuru

We have created a live version of our recommendation system, available online at `http://recipeguru.me`. The web app allows users to create a recipe box out of any number of AllRecipes recipes. Users can then try out our *ingredients + dish* approach, our *recipe factors* approach with cosine similarity metric, and our *ingredient factors* approach with cosine similarity metric. Inspecting the algorithms' recommendations side by side allows one to see clearly the different recommendation patterns that the algorithms tend towards.

In addition, one can use the web app to visualize both the *recipe factor* and *ingredient factor* factorizations of any recipe. Mousing over a recipe in the recipe box will display the proportions of recipe factors on the left, and of ingredient factors on the right.

References

[1] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2000.

[2] Yifeng Li and Alioune Ngom. The non-negative matrix factorization toolbox for biological data mining. *Source code for biology and medicine*, 8(1):1–15, 2013.

[3] Anand Rajaraman and Jeffrey David Ullman. *Recommendation Systems.* Cambridge University Press, 2011.

Appendix A: Selected Recipe Factors (K=40)

Factor 0
Braised Balsamic Chicken
Spicy Garlic Lime Chicken
Chicken Breasts with Vinegar
Unbelievable Chicken
Yummy Honey Chicken Kabobs

Factor 2
Quinoa and Black Beans
Baked Kale Chips
Quinoa Side Dish
Quinoa Tabbouleh
Butternut Squash Soup

Factor 3
Broiled Tilapia Parmesan
Grilled Salmon I
Baked Dijon Salmon
Maple Salmon
Marinated Grilled Shrimp

Factor 7
Cream Cheese Frosting II
Marshmallow Fondant
Simple White Cake
Special Buttercream Frosting
Buttercream Icing

Factor 8
The Best Rolled Sugar Cookies
Sugar Cookie Icing
Easy Sugar Cookies
Sugar Cookie Frosting
Soft Christmas Cookies

Factor 9
Black Bean Veggie Burgers
Sweet Potato Burritos
Vegetarian Chili
Eggplant Parmesan II
Tofu Parmigiana

Factor 11
Jamie's Cranberry Spinach Salad
Roquefort Pear Salad
Strawberry Spinach Salad I
Spinach and Strawberry Salad
Baked Asparagus

Factor 12
Easy Meatloaf
Banana Banana Bread
Homemade Mac and Cheese
Stuffed Peppers
Old Fashioned Potato Salad

Factor 15
Banana Crumb Muffins
Maine Pumpkin Bread
Apple Pie by Grandma Ople
Clone of a Cinnabon
Mom's Zucchini Bread

Factor 18
Chocolate Raspberry Cheesecake
Tiramisu Layer Cake
Chantal's New York Cheesecake
Chocolate Trifle
Tiramisu II

Factor 21
Sangria! Sangria!
Beer Margaritas
Irish Soda Bread
Classic Spanish Sangria
Guinness Corned Beef

Factor 22
Best Lemonade Ever
Luscious Slush Punch
Chai Tea Mix
Cowboy Cookie Mix in a Jar
Creamy Hot Cocoa

Factor 23
Amish White Bread
Clone of a Cinnabon
Sweet Dinner Rolls
Best Bread Machine Bread
French Baguettes

Factor 25
Jalapeno Popper Spread
Zuppa Toscana
Sausage Stuffed Jalapenos
Jalapeno Cheese Dip
Buffalo Chicken Sandwiches

Factor 27
Chicken Tikka Masala
Naan
Curried Coconut Chicken
Chicken Makhani
Vietnamese Spring Rolls

Factor 30
Best Steak Marinade in Existence
Grilled Baby Back Ribs
Marinated Flank Steak
Garlic Prime Rib
Foolproof Rib Roast

Factor 31
Yummy Sweet Potato Casserole
Perfect Turkey
Awesome Sausage
Homestyle Turkey
Grandma's Green Bean Casserole

Factor 34
Best Brownies
Best Big
Best Chocolate Chip Cookies
Soft Chocolate Chip Cookies
Soft Oatmeal Cookies

Factor 35
Buffalo Chicken Wings
Old Fashioned Onion Rings
Fried Mozzarella Cheese Sticks
Best Ever Jalapeno Poppers
Best Egg Rolls

Factor 38
Easy Mexican Casserole
Ham and Hash Brown Casserole
Christmas Sausage Casserole
Sausage Casserole
Taco Pie

Factor 39
Simple Scones
Grandma Johnson's Scones
Cranberry Pistachio Biscotti
World's Best Scones!
Brownie Biscotti

Appendix B: Selected Ingredient Factors (K=50)

Factor 3	**Factor 4**	**Factor 7**	**Factor 8**
all-purpose flour	whole wheat flour	black beans	bread flour
white sugar	applesauce	black beans	all-purpose flour
eggs	wheat germ	kidney beans	water
butter	rolled oats	ground turkey	milk
confectioners' sugar	skim milk	cannellini beans	white sugar

Factor 9	**Factor 10**	**Factor 11**	**Factor 12**
water	chicken breast	salmon	mayonnaise
rice wine vinegar	whole chicken	tilapia	sweet pickle relish
fish sauce	boneless	halibut	hard-cooked eggs
rice noodles	chicken thighs	pink salmon	cucumbers
soy sauce	red potatoes	salmon	tomato

Factor 13	**Factor 14**	**Factor 15**	**Factor 17**
rump roast	chocolate chips	coconut milk	potatoes
sirloin tip	white sugar	ground turmeric	red potatoes
chuck roast	butter	plain yogurt	cabbage
beef stew meat	all-purpose flour	chickpeas	kielbasa
cubed stew meat	chocolate pudding	garam masala	corned beef brisket

Factor 18	**Factor 19**	**Factor 23**	**Factor 24**
salsa	pear	sweet potato	peeled shrimp
corn tortillas	salad greens	butternut squash	sea scallops
chicken breast	romaine lettuce	sweet potatoes	crabmeat
taco seasoning mix	avocado	chopped	medium shrimp
chile peppers	blue cheese	creamed corn	large shrimp

Factor 26	**Factor 27**	**Factor 32**	**Factor 34**
tomato	ice cubes	butternut squash	chicken wings
feta cheese	lemon-lime beverage	kale	pork back ribs
shredded zucchini	vodka	Brussels sprouts	pork spareribs
medium eggplant	pineapple juice	chicken stock	ketchup
roma tomatoes	club soda	chicken broth	barbeque sauce

Factor 36	**Factor 37**	**Factor 38**	**Factor 42**
strawberries	water	ripe peaches	ricotta cheese
bananas	chickpeas	white sugar	spaghetti sauce
kiwis	tofu	apples	lasagna noodles
fruit jam	vegetable stock	rhubarb	mozzarella cheese
strawberries	black beans	blackberries	crushed tomatoes

Factor 44	**Factor 45**	**Factor 47**	**Factor 49**
boneless pork chops	ground beef	turkey	pumpkin
pork tenderloins	lean ground beef	ham	pumpkin pie spice
boneless pork loin	refried beans	dried sage	canned pumpkin
bone-in pork chops	Colby cheese	turkey stock	ground cloves
pork chops	flour tortillas	turkey sausage	pumpkin puree

Appendix C: Supplementary Tables and Figures

Figure 1. Evaluation Metrics for Recipe Recommendation Approaches

approach	precision	recall	f-measure
ingredient similarity	0.0109	0.0383	0.161
ingredient + dish similarity	0.0113	0.0399	0.167
recipe factors (cosine)	0.0270	0.1001	0.406
recipe factors (dot)	0.0356	0.1340	0.539
ingredient factors (cosine)	0.0077	0.0264	0.112
ingredient factors (dot)	0.0059	0.0198	0.0085
random	0.0063	0.0212	0.0091
naive popularity	0.0314	0.1171	0.473

Figure 1: Two content-based approaches, two collaborative filtering approaches, two hybrid approaches, and two benchmarks were evaluated with the following test: We withheld 10% of all ratings from all users with more than 300 ratings, and trained the algorithms on the remaining ratings data. Then, for each algorithm, for each user, we measured the precision and recall of the top $N = 200$ recommendations, and for each algorithm we computed the mean precision, recall, and f-statistic across all users.

Figure 2. Selecting λ with Cross-Validation

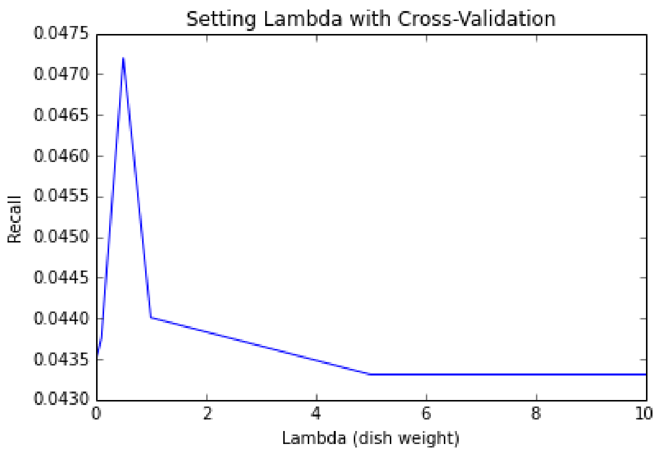

Figure 2: In the *ingredient + dish similarity* approach, λ is set using 3-fold cross-validation on the training data. This figure shows the recall, for various values of λ, of the top 200 recommendations. The optimal value was determined to be $\lambda = 0.5$

7 Note to the Instructors

Two days ago, after we finished this project, we stumbled upon an obscure academic paper from 2011 whose authors came up with many of our ideas independently. They too decided to build a recipe recommendation system, scraped data from AllRecipes.com, and tried a content-based approach based on ingredient fraction vectors, a collaborative filtering approach, and even, incredibly enough, something almost identical to our hybrid approach. We are very disappointed to learn that ideas which we thought were quite original have apparently already been tried before.

Our methods are different from the those of other paper in that we scraped user recipe boxes (binary ratings) rather than numerical ratings (1-5 stars), that we generated user profiles by aggregating the recipes in the user's recipe box rather than by taking the user's row in the decomposed matrix, that we developed the *ingredient+dish similarity* approach, that we used a different algorithm for NMF (multiplicative update rather than alternating least squares), and that we used a different error metric (recall rather than absolute prediction error).

We had conducted a search of the literature prior to submitting our project proposal, but this paper did not show up in any search results at all. We only came across it this week accicentally when we looked through the references section of another paper.

Peter Forbes and Mu Zhu. Content-boosted matrix factorization for recommender systems: experiments with recipe recommendation. In *RecSys '11: Proceedings of the fifth ACM conference on Recommender systems*, pages 261-264, 2011.