Cuisine Classification and Recipe Generation

Juhi Naik, Vinaya Polamreddi

November 2015

1 Introduction

Food is one of the most fundamentals parts of life. We wanted to use machine learning to approach food and specifically recipes in new ways. We tried to answer two different questions using recipe data: Given a set of ingredients, what is the cuisine of that recipe and if we can automatically generate a recipe?

2 Classification

2.1 Related Work

Classification of recipes into respective cuisines based on ingredients has been done by [1] using techniques like associative classification and support vector machines. A graph-like model is proposed by [2] to model recipes and tries to match substructures among the graphs to find similar recipes. On the other hand, [3] tries to model recipes using a variety of features like taste, smell, texture, nutritive value etc. For our project, we have performed a variety of classification techniques and analyzed how they perform in comparison.

2.2 Data

Our dataset is recipes each consisting of a set of ingredients labeled with a cuisine. It is from a publicly available dataset sourced from http://epicurious.com and http://menupan.com . Sample data:

Ingredients	Cuisine
vegetable oil, starch, shrimp, egg, cream	East Asian
honey, cocoa, butter, cereal, gelatin	North American

The y labels consist of 11 cuisines with the following distribution in the data: The data has a very high skew towards North American cuisine. To understand how this skew affects our data we created four different input sets to test our classification algorithms on:

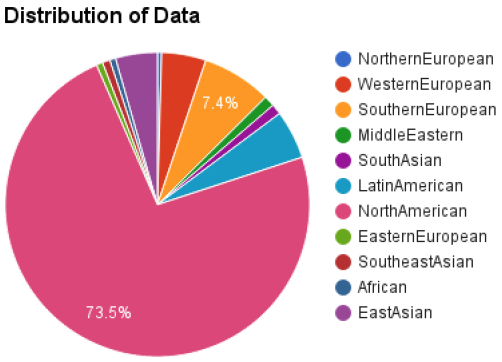


Figure 1: Distribution of Data

1. Input set 1: Full dataset divided 70:30 for train and test respectively
2. Input set 2: Dataset divided 70:30 train and test after removing all North American recipes
3. Input set 3: Dataset subsampled to have an equal amount of each cuisine. Train set of 200 recipes from each cuisine, and test set of 50 recipes from each cuisine with a total of 2200 and 550 recipes for train and test.
4. Input set 4:Dataset randomly sampled for 2200:550 recipes of train and test sets

2.3 Methods

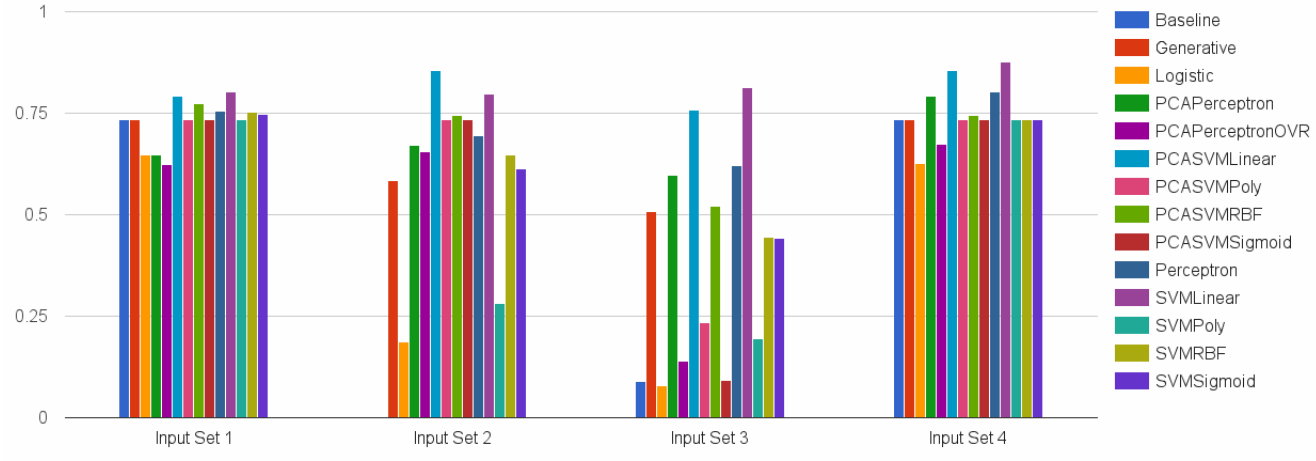
SVM, Logistic Regression, Perceptron and PCA were implemented using the library sklearn [8].

2.3.1 Baseline

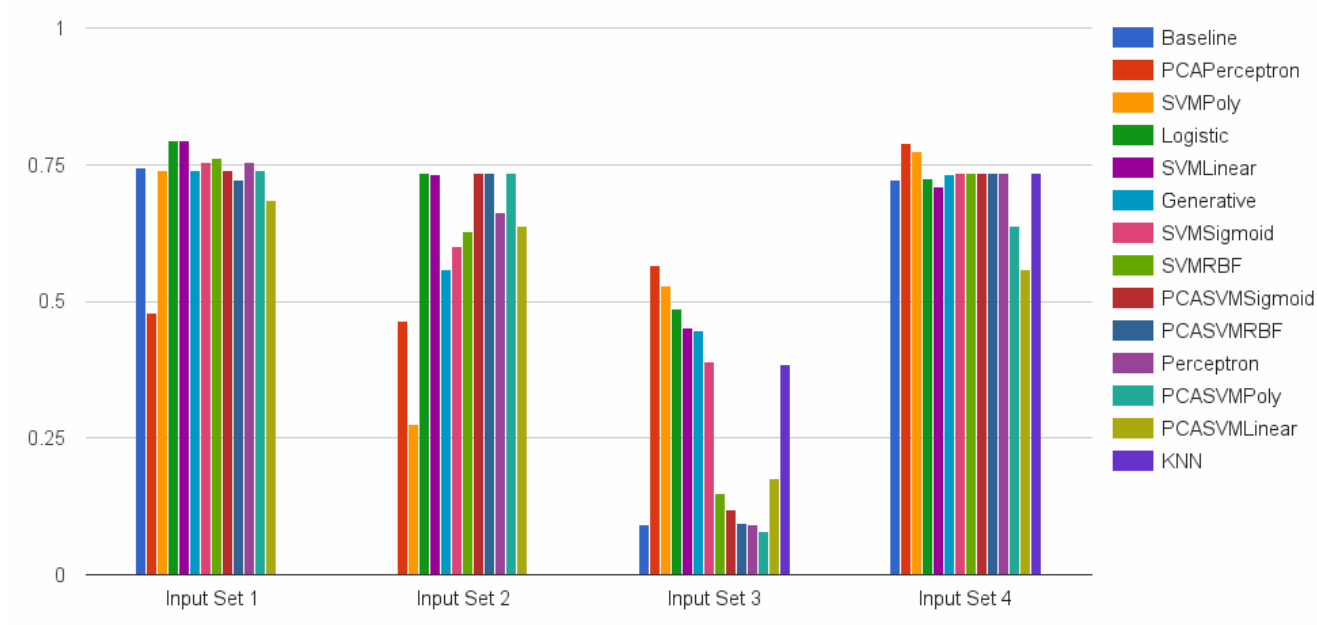
Since 73.5% of the data is North American, even predicting North American for every input will give us 73.5% accuracy, so our baseline is the majority label.

2.3.2 Multi-variate Logistic Regression

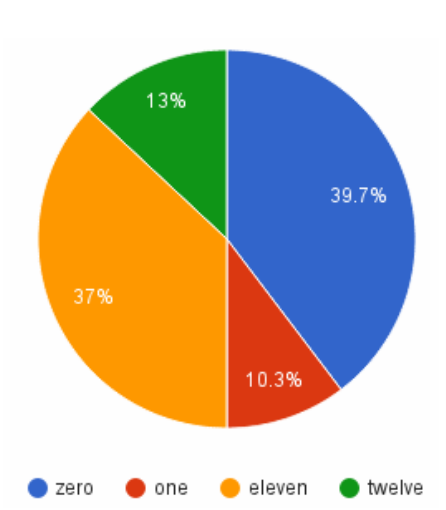
We applied it to each of our input sets tuning hyperaparameters: penalty, inverse of regularization



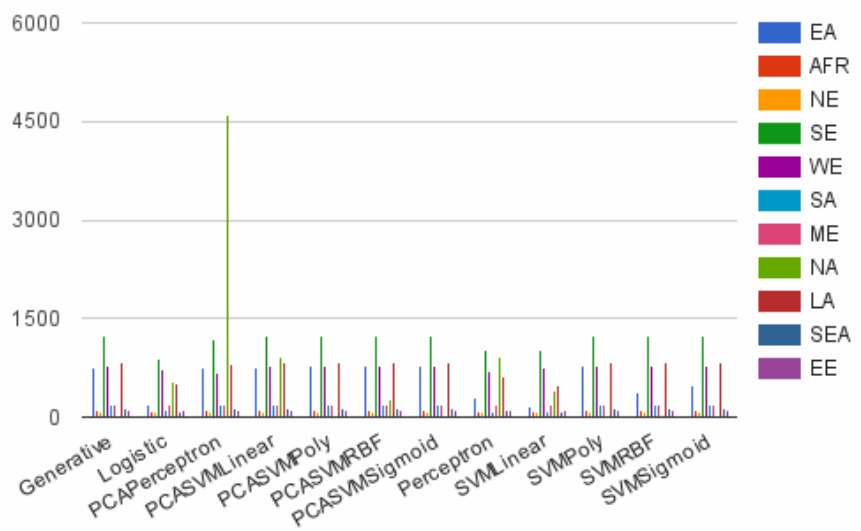
(a) Train Accuracy across Datasets and Algorithms



(b) Test Accuracy across Datasets and Algorithms



(c) % of data misclassified by # of methods



(d) Number of False Negatives of Full dataset by Cuisine

Figure 2: Results for Classification

strength, class weight, etc. Tuning the parameters only changed the test accuracy by a maximum of 2%. Using l2 regularization, we have an accuracy of 79.9% accuracy for train and 79.3% for accuracy on the test set for input set 1.

2.3.3 SVM

Various kernels like linear, polynomial, sigmoid and RBF were used after tuning various hyper-parameters like the penalty term, kernel coefficient, degree of polynomial etc. using cross-validation. However, except linear kernels, all others overfit the training data by predicting North American on all points. Tuning the hyper-parameters mentioned above didn't improve the accuracy by more than 1%.

2.4 Perceptron

Perceptrons were similarly trained on the dataset using 'l1' and 'l2' penalty terms. This was one of the few methods that was not affected by the skewness in the data. Using PCA on the data to reduce the dimensionality before applying Perceptron classification reduces overfitting even more. If we inspect the confusion matrices by cuisine, North American cuisines have the highest False Negatives given by the PCA + Perceptron Algorithm on the full dataset.

2.5 Generative

To use Generative Models for classification, we implemented a form of Naive Bayes by calculating the probabilities of each ingredient appearing in the cuisines and the fraction of each cuisine in the data set. It has an accuracy of about 74% on the full dataset and about 56% on the stratified data. Thus, it doesn't do too well at learning the models of cuisines with the given features.

2.6 Nearest Neighbors

We implemented a modified nearest neighbors algorithm which took all recipes of least distance from the feature vector of test data and took the most common cuisine among those neighbors as the prediction. This algorithm ran exceptionally slow and we could only run it on the two smaller input sets each of which took approximately 10 hours to run. The accuracy on the small random input set was 73.4% and the accuracy on the stratified data was 38.3%.

2.7 Results and Analysis

Figures (a) and (b) show the train and test accuracy for each method for each of the input sets. The first bar in each except for input set 2 is the majority label baseline. Taking a closer look at the test accuracy for each data set, we can see that for input set 1 and 4 - those with the NorthAmerican skew in the input - most of the methods do similarly. However, only Logistic and SVM with a linear kernel do noticeably better than the baseline for input set 1, while some of the other SVM kernels (RBF and Sigmoid) do marginally better. On the other hand, for the smaller and still skewed data, logistic and SVM linear do no better than the baseline while all the other SVMs do exactly as much as the baseline. Looking at the confusion matrix for each method, it is clear that the rest of the SVMs are overfitting and have only predicted NorthAmerican.

For input set 3 where we have removed the skew in training data by undersampling, all the methods have overall accuracy that is lower than each of their accuracies given the other training sets which can lead to hypothesizing that training on the skewed data sets made the classifiers perform better. However, if the accuracy is compared against the baseline, almost all the methods did significantly better when trained on this dataset. This implies that training on the skewed data is making the classifiers perform closer to the majority label. Digging deeper into the confusion matrices for input set 2 vs input set 1, it can be seen that classifiers trained on the less skewed data predict more diversely where the false negatives are spread out more evenly for all cuisines whereas the incorrect predictions from classifiers trained on input set 1 were almost exclusively NorthAmerican.

We have also tried all the SVM and perceptron algorithms after using PCA on the input and without using PCA. There was no significant difference in any data set or algorithm due to using PCA and not.

We also ran analyses comparing each of the methods' actual predictions against each other. In Figure (c) we plotted the results from analyzing how many methods misclassified each data point for input set 1. 39.7% of data was classified correctly by every method while 13.4% of the data was misclassified by every method. This implies that even if we somehow picked the best prediction from all the predictions, we can't do better than 13.4% error rate. Digging down, we found that the one method that uniquely misclassified 10.34% of the data was

logistic regression. This implies that the way logistic regression learns and is predicting is somehow different than the way the SVM's and perceptrons are learning from this data.
Examining the false negatives graph, we can see that most algorithms are similar in which cuisines they misclassify the most except for Perceptron which predicted many NorthAmerican recipes incorrectly which clearly shows that perceptron is not overfitting. However, its accuracy is one of the lowest and looking at the confusion matrix, it can be seen that it is randomly predicting cuisines and hasn't learned much.

3 Generation

3.1 Related Work

[4] works on understanding different types of pairing between ingredients, while [5] does the same for pairing between a dish and a wine. [6] and [7] work on sub parts of recipe generation, the former attempting to predict the amount of a particular ingredient given the amounts of other ingredients and the latter trying to synthesize recipes given an initial seed of favourite ingredients.

3.2 Data

3.2.1 Collection

Allrecipes is a popular recipe sharing website. Recipes contain a variety of information: ingredients, quantities, instructions, cooking time, user ratings, etc. For our project, we scraped the list of ingredients and instructions for approximately 63000 recipes.

To parse the list of ingredients: We wanted the instructions in the form of a list of actions where an action was made of a verb and noun pair. In order to get the data into this format, we created a list of 70 verbs commonly used in recipes.

3.2.2 Input

The input for the model is a set of ingredients available with the user out which a subset can be chosen to generate recipes.

3.3 Methods

We used a Generative Model based on Bayesian pairwise probabilities calculated from the recipes collected above. The input taken from the user consists of a set of ingredients, a subset of which can be used to generate new recipes.
Given the set of ingredients, our aim is to first select a feasible subset of those ingredients and then generate a sequence of instructions to perform on them.

3.3.1 Step 1: Select Ingredients

We first calculate pairwise probabilities of ingredients from our training set by counting how many times each pair of ingredients appears in the set of recipes. To get a new set, we would ideally like to maximize the probability of the entire subset. However, the search space in this case would be impossible to search for. So we begin with the null set and iteratively add new ingredients to the set by taking the most feasible from the remaining ingredients using the joint probabilities of the new ingredient with only the last added one. We stop adding new ingredients once the probability of adding a new one goes below a certain threshold.

3.3.2 Step 2: Generate Instructions

Instructions for a recipe are a sequence of actions, each of which is a tuple of verb and ingredient. To generate this sequence, we use 3 different models and compare the recipes generated by each of them given the same set of initial ingredients.

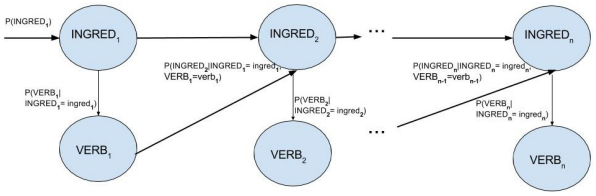
1. Action - Action Probabilities:
   Similar to our previous step, here, we directly take the most probable complete action conditioned on the previous action added to the sequence. The probabilities for the sampling are calculated using the number of times the pair of actions appear in the training recipes.

2. Action-Ingredient-Verb Probabilities:
   Here, we first choose an ingredient to work on, given the previous action performed. Then, a verb is predicted conditioned on both the previous complete action and the new ingredient chosen. Thus, this model assumes a logical ordering of ingredients that we work on during a particular preparation and a logical set of verbs that can possibly be performed on a given ingredient.

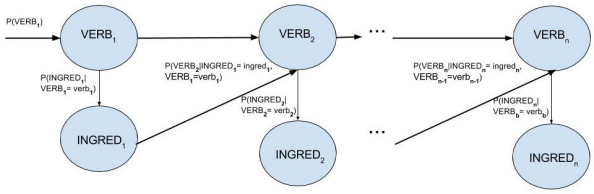
3. Action-Verb-Ingredient Probabilities:
   This model is similar to the previous model except that we first predict a verb given the pre-

(a) Ingredient-Ingredient Probabilities

(b) Action-Action Probabilities



(c) Action-Ingredient-Verb Probabilities



(d) Action-Verb-Ingredient Probabilities

Figure 3: Bayesian Networks for Generation of Recipes

vious action performed and then choose an ingredient conditioned on both the previous complete action and the new verb chosen. Thus, this model assumes a logical ordering of verbs that we do during a particular preparation and a logical set of ingredients that each verb can be done on.

3.4 Results

Below are examples of recipes generated by the first model for selection of ingredients and the 3 models for generation of actions:
User Input:
potato, parsley, butter, corn syrup, cornstarch, cinnamon, onion, celery, garlic, tomato, chicken, corn, turkey, paprika, sugar
Ingredient Selection:
garlic, onion, tomato, salt, butter, celery, potato
Model 1: Action - action probabilities
preheat butter → stir onion → stir garlic → stir tomato → stir salt → cook potato → cook celery
Model 2: Action-Ingredient-Verb Probabilities
preheat butter → stir onion → stir garlic → stir tomato → stir salt → place potato → stir celery → stir parsley
Model 3: Action-Verb-Ingredient Probabilities
preheat butter → preheat salt → beat potato → beat garlic → drain onion → drain tomato → pour chicken → pour celery → add parsley → add paprika

3.5 Analysis

We will rely on human evaluation of output to analyze the results due to the nature of our problem. Given the user inputted list of ingredients, the model first selects a set of ingredients that make sense together as they belong to a savory dish without sweet ingredients. The first two models result in very similar recipes with the sequence of ingredients almost exactly the same and only two verbs that are different. The third model predicts many different verbs and ingredients which is probably a result of the model primarily predicting a sequence of verbs and choosing possible ingredients. It seems to pick verbs that don't make sense with our ingredients resulting in strange instructions such as "pour chicken". For recipes, generating the sequence on ingredients or the whole actions seems to make more sense than generating the recipe based on a sequence of verbs. The first two recipes actually make semantic sense: you would preheat butter and stir the onion, garlic and tomato which take longer to cook before cooking the potato, adding salt or adding parsley. Overall, model 2 seems to be the better model allowing for more dynamic formation of instructions and still generating a reasonable sequence.

4 Future Work

All of our models pick the next most likely ingredient or verb by picking the most probable next step according to pairwise Bayesian probabilities. However, optimizing over the entire sequence would result in much more feasible recipes. We can approach this problem as an MDP where the rewards are the joint probability metrics on the entire set of ingredients and instructions generated. However, our data contains hundreds of unique ingredients and actions, and computing all possible joint probabilities was an unfeasible task. Trying to solve this as an MDP in conjunction with other heuristics would be interesting future work.

5 References

1. Su, Han, et al. "Automatic recipe cuisine classification by ingredients." Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication. ACM, 2014.

2. Wang, Liping, et al. "Substructure similarity measurement in chinese recipes." Proceedings of the 17th international conference on World Wide Web. ACM, 2008.

3. Xie, Haoran, Lijuan Yu, and Qing Li. "A Hybrid Semantic Item Model for Recipe Search by Example." Multimedia (ISM), 2010 IEEE International Symposium on. IEEE, 2010.

4. Ahn, Yong-Yeol, et al. "Flavor network and the principles of food pairing." Scientific reports 1 (2011).

5. Meier, Justin. From Food to Wine. Tech. N.p., 2014.

6. Safreno, Doug, Deng, Yongxing. The Recipe Learner. Tech. N.p., 2013.

7. Agarwal, Rahul, Jachowski, Daniel, Shengxi, David. RoboChef: Automatic Recipe Generation. Tech. N.p., 2009.

8. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.