What Cuisine? - A Machine Learning Strategy for Multi-label Classification of Food Recipes

Hendrik Hannes Holste, Maya Nyayapati, Edward Wong

***Abstract*—We consider the problem of predicting the cuisine of a recipe given a list of ingredients. Such classification can help food databases and recipe recommender systems autonomously categorize new recipes based on their ingredients. Results of our evaluations show that a classification accuracy at least 77.87% is possible on a data set of 39,774 recipes, surpassing the accuracy of a baseline predictor that, for each recipe, trivially guesses the most common cuisine.**

Keywords*—*food, recipe, data mining, machine learning, classification

I. Introduction

Recipe search and recommendation websites such as Yummly are growing in popularity [1]. When users contribute new recipes to these websites, they are faced with a large number of fields that require manual input of ingredient lists, cooking steps, cuisine types, and descriptions, among other data. The presence of a large number of input fields is problematic, as an increase in form inputs has been shown to heighten the probability that a user, out of frustration, abandons a form entirely [2].

In this context, we present a machine learning strategy to automatically categorize recipes by cuisine. Automatic classification has three benefits.

First, machine-driven classification reduces required user input, thus potentially decreasing form abandonment rates.

Second, it is also useful in developing a notion of cuisine similarity, allowing restaurant recommendation systems such as Yelp to compare user cuisine preferences with restaurant meal offerings, thus potentially leading to more relevant suggestions.

Finally, automatic cuisine labeling can also help users discover what cuisines their custom, un-categorized recipes are likely to belong to, allowing them to label their recipes with less cognitive effort [1].

II. Data and exploratory analysis

The dataset, made available by recipe index Yummly [4] through data science competition host Kaggle [5], consists of 39,774 recipes. Each recipe comprises a list of ingredients, a unique identifier, and a cuisine label.

[1]simply choosing a likely match from a set of suggested labels remains easier than recalling from scratch [3]

A. Basic Statistics

The distribution of recipes is as follows:

Cuisine	Number of recipes
Brazilian	467
Russian	489
Jamaican	526
Irish	667
Filipino	755
British	804
Moroccan	821
Vietnamese	825
Korean	830
Spanish	989
Greek	1175
Japanese	1423
Thai	1539
Cajun Creole	1546
French	2646
Chinese	2673
Indian	3003
Southern US	4320
Mexican	6438
Italian	7838

Italian is the most popular cuisine, with 7,838 recipes. 6,703 distinct ingredients exist, though this number may over-count ingredients that are categorically similar, for example *monterey jack cheese* and *swiss cheese*.

B. Frequency of ingredients and outliers

Among the set of 25 least common ingredients, the following ingredients each only occur in one recipe. Optimizing a predictor to look for these least common ingredients may over-fit training data.

Minute white rice, bottled low sodium salsa, clam sauce, kraft mexican style shredded four cheese with a touch of philadelphia, mahlab, broccoli romanesco, flaked oats, country crock honey spread, saffron road vegetable broth, black grapes, orange soda, ginseng tea, adobo all purpose seasoning, chinese buns, custard dessert mix, gluten-free broth, burger style crumbles, egg roll skins, cooked vegetables, schnapps, mild sausage, vegetarian protein crumbles, white creme de cacao, gluten flour, dried neem leaves.

The most popular ingredients that are present across many dishes, which we suspect may have lower predictive power in distinguishing cuisines, are as follows:

Ingredient	Occurrences in recipes
salt	18049
onions	7972
olive oil	7972
water	7457
garlic	7380
sugar	6434
garlic cloves	6237
butter	4848
ground black pepper	4785
all-purpose flour	4632
pepper	4438
vegetable oil	4385
eggs	3388
soy sauce	3296
kosher salt	3113
green onions	3078
tomatoes	3058
large eggs	2948
carrots	2814
unsalted butter	2782
ground cumin	2747
extra-virgin olive oil	2747
black pepper	2627
milk	2263
chili powder	2036

C. Noise in ingredient strings

Further analysis revealed the presence of noise in ingredient names, which may lead to inaccurate predictions. For example, certain ingredient strings encode adjectives and unit information, while others are polluted with unicode escape sequences and punctuation. Finally, the presence of undesirable ingredient descriptors adjectives such as ”big” may reduce classification accuracy too, because two ingredient strings that describe the same ingredient may be treated as distinct. For example, ”medium eggs” and ”large eggs” are distinct ingredients, though discerning between the size of eggs may not provide differentiating information about a recipe’s cuisine and thus merely represent noise.

Further examples of ingredients that may need preprocessing and normalization are:

- (15 oz.) refried beans
- 33% less sodium smoked fully cooked ham
- 2 1/2 to 3 lb. chicken, cut into serving pieces
- kraft mexican style 2% milk finely shredded four cheese

D. Additional features

The training data was lean, only containing a list of ingredients per recipe, so we explored encoding additional features.

We observed that certain ingredient names directly encode valuable hints about the cuisine, for example, ”jamaican jerk rub” is likely to be included in jamaican recipes, and ”crme fraiche” is likely to be included french cuisines.

We also hypothesized that the number of ingredients per recipe may be a useful indicator of which cuisine a recipe belongs to and thus be a useful feature. For example, certain cuisines may be defined by very simple recipes with few ingredients while others may be defined by complex recipes with many ingredients. However, plotting the number of ingredients per recipe proved otherwise; the distribution of ingredients per recipe was too homogeneous distribution and encountered high variance, making it a noisy and thus weak feature.

Cuisine	Median number of ingredients per recipe	Standard deviation
irish	9.00	3.70
mexican	10.00	4.66
chinese	12.00	4.04
filipino	10.00	3.85
vietnamese	12.00	5.25
moroccan	13.00	4.80
brazilian	9.00	5.55
japanese	9.00	4.24
british	9.00	4.16
greek	10.00	3.73
indian	12.00	5.02
jamaican	12.00	4.76
french	9.00	4.14
spanish	10.00	4.16
russian	10.00	4.05
cajun creole	12.00	4.61
thai	12.00	4.41
southern us	9.00	3.87
korean	11.00	3.88
italian	10.00	3.81

III. Predictive Task

A. Task and error measure

Given a list of ingredients belonging to a recipe, our model should predict its cuisine. We use the classification error to evaluate our model.

B. Training set and validation set

Data is divided into a training set containing 80 percent of samples and a validation set containing the remaining 20 percent of samples. The training set will be used to train predictive models, while the validation set shall be used to to assess how well our machine learning strategies generalize to unseen data, and to reduce our risk of over-fitting our model to training data.

C. Baseline model

The baseline model, provided by Kaggle [5], simply predicts the most popular cuisine for any given recipe. In this data set, the most popular cuisine was Italian. This trivial predictor achieved a training set error rate of 80.15% and validation set error rate of 80.85%, slightly more accurate than simply randomly guessing a recipe's cuisine, which resulted in a training and validation set error rate of 95.00%.

D. Features

1) Custom tf-idf scoring: For our model that implemented tf-idf scoring, we did not generate a typical feature vector to train a machine learning algorithm. Instead, we aimed to find the ingredients that were unique to each cuisine by calculating the tf-idf score for each ingredient in a recipe with respect to each cuisine, and finally predicting the cuisine with the highest total score. The calculations are described in detail below.

2) Logistic and random forest regression: The feature representation that was the most useful for our task was a "bag-of-ingredients". To produce the feature vector for logistic and random forest regression, we generated a sparse vector of occurrence counts of each ingredient name in a given recipe, conceptually similar to the "bag-of-words" feature model.

After considering additional features as described in the exploratory analysis, we deemed those would provide only minor accuracy gains and thus did not calculate and encode them.

E. Feature pre-processing

As mentioned in the exploratory analysis, it was evident that pre-processing ingredient strings could lead to better predictive outcomes, especially in a "bag-of-ingredients" model.

Each ingredient string in the training set was pre-processed as follows:

1) Convert all letters into lowercase text.
2) Strip escaped unicode, e.g. $\backslash$*u2012*.
3) Strip punctuation such as semicolons and commas.
4) Strip parentheses and the strings they enclose, e.g. *(16 oz.)*
5) Strip food descriptors such as "hot" or "unsweetened", drawing from a pre-defined list [6].
6) Strip excess whitespace, including preceding and trailing space characters.

As the pre-processing procedure utilized several regular expressions and was thus computationally expensive, we implemented a caching mechanism that only re-processed data if the training set changed or pre-processing procedure was modified, significantly reducing average pre-processing time and allowing for more frequent model testing and analysis.

IV. Approach and Machine Learning Model

A. Custom tf-idf scoring model

1) Model construction: Our goal was to design a scoring model that indicates what cuisine an ingredient most-likely belongs to, and given scores for each ingredient against each cuisine, could predict the cuisine of a recipe.

We thought we could use term-frequency inverse-document-frequency (tf-idf), which involves calculating the relative frequency of a ingredient in a particular cuisine compared to all other cuisines. Doing so would reveal the "indicative" ingredients that could be used to, with relatively high confidence, predict the cuisine of a recipe. For example, the ingredient *garam masala* is an "indicative" ingredient i that is often associated with cuisine c; a higher tf-idf for i for a cuisine c score means that i occurs more frequently in cuisine c than in any other cuisines.

On the other hand, the ingredient *water* is common across many cuisines, and thus not a good "indicative" ingredient.

For each ingredient in the training set, we calculated its tf-idf score with respect to cuisine type as follows:

$$tf(i,c) = \text{number of times ingredient } i \text{ appears in cuisine } c$$

$$idf(i,C) = \frac{\text{number of cuisines in } C}{\text{number of cuisines that contain ingredient } i}$$

$$tfidf(i,c,C) = tf(i,c) * idf(i,C)$$

2) Scoring example: Applying the scoring mechanism to the aforementioned examples *garam masala* and *water*, the tf-idf score calculated for *garam masala* with respect to Indian cuisine is 3584.9572, and for contrast, the tf-idf score calculated for *garam masala* with respect to Italian cuisine is 4.1588. The score reveals that, as expected, *garam masala* occurs significantly more frequently in Indian cuisine.

To contrast, *water* has a tf-idf score of 0.00 with respect to all cuisines, because it occurs in all cuisines.

3) Prediction mechanism: In order to make a prediction given only a list of ingredients for a recipe, the model calculates the tf-idf scores for each ingredient with respect to each cuisine and summed the scores by cuisine. Then, the model predicts whichever cuisine has the highest score.

$$score(R,C) = max(\sum_{i \in R} tfidf(i,c,C) \forall c \in C)$$

4) Prediction example: Suppose the set of cuisines were A, B, and C, and the model was given a recipe with ingredients X, Y, and Z.The tf-idf scores for each ingredient-cuisine pair are then calculated as follows:

	A	**B**	**C**
X	311	260	0
Y	0	58	0
Z	0	12	15
Total	311	330	15

In the last row, it is evident that cuisine B has the highest total score for the three ingredients, so the model predicts B. We counted the number of times each ingredient occurred in each cuisine in the training set and used those number to calculate tf-idf scores on both the training and validation sets.

5) Performance: This tf-idf scoring model had a 34.25% error rate on the training set and 33.73% error rate on the validation set, which was significantly lower than the baseline predictor.

While the model performed well, it is still considerably naive, as it simply sums the tf-idf scores per cuisine and predicts the cuisine with the highest score. This is problematic when two cuisines have similar scores because the model has no consistent method for resolving such a close tie.

In the example above, the difference between the scores of cuisine A and B is relatively small, implying that a recipe, according to the tf-idf model, could belong to cuisine A with almost equal likelihood.

Conversely, considering the scores for ingredients X and Y, it is evident that cuisine B is a more favorable candidate than cuisine A. Although ingredient X favors cuisine A, the difference between 311 and 260 is not as stark as the difference between 0 and 58 for ingredient Y. Our naive summation approach does not take the relative strength of differences in scores into consideration.

B. Random Forest and Logistic Regression

The shortcomings of the custom tf-idf scoring model motivated our attempt at applying machine learning to the predictive task, namely random forest and logistic regression.

1) Model construction: As mentioned earlier, to train the regression models, we generated a ”bag-of-ingredients” feature representation for each recipe - a sparse vector of occurrence counts of each ingredient name in a given recipe. We generated a ”bag-of-ingredients” rather than a ”bag-of-words” feature vector because many ingredients contain multiple words that require semantic grouping.

For example, in a ”bag-of-words” vector, the ingredient *green onions* would be split into separate categorical features *green* and *onions*, thus discarding semantic information; the disjoint words *onions* or *green* independently do not encode the same meaning as the ingredient *green onions*.

Finally, we also tested logistic regression on feature vectors that transformed the original vector of ingredient occurrence counts into ones that used normalized tf-idf scores (with sub-linear tf-scaling).

2) Performance: The performance of the below models is as follows:

Model	Training Error Rate	Validation Error Rate
Baseline (predict Italian)	0.8015	0.8085
Random Forest Regressor	0.0077	0.3089
Logistic Regressor, no tf-idf	0.1361	0.2104
Logistic Regressor, with tf-idf	0.1844	0.2227

Although random forest regression outperforms logistic regression on the training set, it over-fits the training data and performs worse than the logistic regressor (no tf-idf) on unseen validation set data. Logistic regression, a simpler model, appears less susceptible to over-fitting.

3) Discussion of logistic regressor and tf-idf: Surprisingly, using normalized tf-idf scores rather than simple occurrence counts in the feature vector performed did not reduce the error rate on both training and validation sets. In fact, it worsened it. Later analysis revealed that using normalized tf-idf scores that range from 0.0 to 1.0 rather than using ingredient occurrence counts (no tf-idf) performed worse for the following reason:

Consider the exemplary ingredient *soy sauce*. The number of times *soy sauce* appears in each cuisine in the training data is as follows:

Cuisine	occurrences of *soy sauce*
irish	42
mexican	306
chinese	11538
filipino	1650
vietnamese	1194
moroccan	24
brazilian	18
japanese	4098
british	18
greek	36
indian	138
jamaican	456
french	36
spanish	18
russian	18
cajun creole	78
thai	2604
southern us	162
korean	3018
italian	120

As expected, *soy sauce* is more prevalent in Asian recipes than others. However, since *soy sauce* appears at least once in all of the cuisines, the tf-idf score for *soy sauce* for each ingredient is 0.0. In other words, the tf-idf score suggests that knowing a recipe contains *soy sauce* would not provide discerning information about its cuisine, which is obviously false. Thus, this example illustrates why a simpler bag-of-ingredients model performed better without tf-idf scoring.

4) Logistic regressor hyperparameter tuning: After deducing that logistic regression with bag-of-ingredients was the best performing model out of the ones tested, we fine-tuned the regularization parameter C^2 and obtained the following results:

C	Training Error Rate	Validation Error Rate
0.01	0.3392	0.3551
0.1	0.2230	0.2492
0.5	0.1603	0.2158
1.0	0.1361	0.2104
2.0	0.1156	0.2103
10.0	0.0807	0.2188
100.0	0.0641	0.2437

[2] C: the inverse of regularization strength where smaller values specify stronger regularization.

Conclusively, the logistic regressor with a regularization parameter of $C = 2.0$ was the best performing model for our predictive task.

V. LITERATURE

We used a data set provided by a Kaggle competition [5] *What's Cooking*, which sourced its data from Yummly [4]. While building our models, we did not reference existing literature that focused on the similar problem domain, though upon completing the project, we discovered various approaches used in other submissions to the Kaggle competition.

Some approaches involved training decision-tree models with a bag-of-words feature representation [**?**], often without pre-processing ingredient strings. Another notable approach trained a neural network with 2 hidden layers and one embedding layer with dropout regularization and a NAG optimizer [7], resulting in an low error rate of 0.2092 on the Kaggle leaderboard's test set.

Notably, in the domain of applying machine learning to recipes, IBM built a software system that can generate "creative recipes" [8].

VI. RESULTS AND CONCLUSION

A. Expected real-world test-set performance

Model	**Training Error Rate**	**Validation Error Rate**	**Test Error Rate**
Baseline (predict Italian)	0.8015	0.8085	N/A
Custom tf-idf scoring	0.3425	0.3373	N/A
Random Forest Regressor	0.0077	0.3089	N/A
Logistic Regressor, with tf-idf	0.1844	0.2227	N/A
Logistic Regressor, no tf-idf	0.1361	0.2104	N/A
Logistic Regressor (C=2.0), no tf-idf	0.1157	0.2103	0.2213

Our logistic regression model ($C = 2.0$, no tf-idf) with ingredient name pre-processing and bag-of-ingredient counts as described above resulted in a test set error of 0.2213 on Kaggle, which does not deviate significantly from the above reported validations et error. Overall, our model performs drastically better than provided baseline model and even the tf-idf scoring models that we initially used.

B. Final feature representation

Our final feature representation used distinct ingredient occurrence counts with pre-processed ingredient names, as described earlier. This classifier may appear to work exceptionally well due to the the distribution of the data set: it is most effective at classifying recipes that also happen to belong to the most popular cuisines, namely Italian and Mexican.

C. Confusion matrix discussion

The confusion matrix, attached in the appendix, visualizes the results of the logistic regressor, giving deeper insight into its performance as well as (dis-)similarity between cuisines.

Evidently, the most difficult cuisine to classify is Russian; the easiest cuisine to classify is Mexican.

Furthermore, the similarity between cuisines can be inferred from the confusion matrix. For instance, according to the model, French food is the most similar to Italian food in terms of ingredients. Not surprisingly, Japanese food is reportedly similar to Chinese food, confirming our intuition as they both belong to the class of Asian cuisines.

VII. FURTHER WORK

A. Investigating mis-classification

The confusion matrix raises interesting results that identify weak-points in our model, such as the high rate of mis-classification of Russian recipes.

It may be worth investigating why certain cuisines are intrinsically more difficult to accurately classify. Perhaps Russian recipes are challenging because they are simplistic; perhaps they comprise only common ingredients, and there are few ingredients that are unique to the Russian cuisine?

The confusion matrix further reveals that certain cuisines are often misclassified as another, thus the model considers such cuisine pairs very "similar".

One question that arises from this observation is whether very "similar" cuisine pairs have relatively more common ingredients than cuisine pairs that are considered "dis-similar". Our isolated analysis of instances of mis-classification confirmed that Asian cuisines were often misclassified as other Asian cuisines because the cuisines share many common ingredients.

B. Ingredient independence assumption

In addition, our initial models that leveraged custom tf-idf scoring of each ingredient were premised on the assumption that each ingredient should be treated independently. But does this assumption hold true?

Intuitively, this does not actually seem to be a valid assumption, as we can anecdotally observe dependence, or coupling, between two or more ingredients.

For instance, the method of tf-idf scoring that we developed could statistically deem *sesame oil* as a strong indicator of Chinese cuisine. However, it is plausible that *sesame oil* coupled with *bok choy* may be an even stronger indicator of Chinese cuisine, while *sesame oil* coupled with *wasabe* in the same recipe otherwise suggests it belongs to the Japanese cuisine.

It may be worth capturing this notion of dependency and encoding it the tf-idf calculations to improve accuracy.

C. More sophisticated ingredient string pre-processing

Finally, it may be worth experimenting with further ingredient string pre-processing. Though we did simple transformation, lemmatization and stemming of words may lead to further gains in classification accuracy. For example "Brown eggs from a Free-Range Chicken" is currently transformed to "eggs from a chicken", though it would likely be worth developing an algorithm to reduce it to the string "egg".

ACKNOWLEDGMENT

The authors would like to thank Professor Julian McAuley at the University of California, San Diego, for his continuing support and wisdom.

REFERENCES

[1] J. Shieber. Yummly raises $15 million at $100m valuation for its recipe recommendation and food delivery business. [Online]. Available: http://techcrunch.com/2015/09/01/yummly-raises-15-million-at-100m-valuation-for-its-recipe-recommendation-and-food-delivery-business/

[2] S. Krug, *Don't Make Me Think: A Common Sense Approach to Web Usability*. Peachpit, 2000.

[3] D. Rock, *Your Brain at Work*. HarperBusiness, 2009.

[4] Yummly. The best site for recipes, recommendations, food and cooking — yummly. [Online]. Available: http://www.yummly.com

[5] Kaggle. What's cooking? [Online]. Available: https://www.kaggle.com/c/whats-cooking

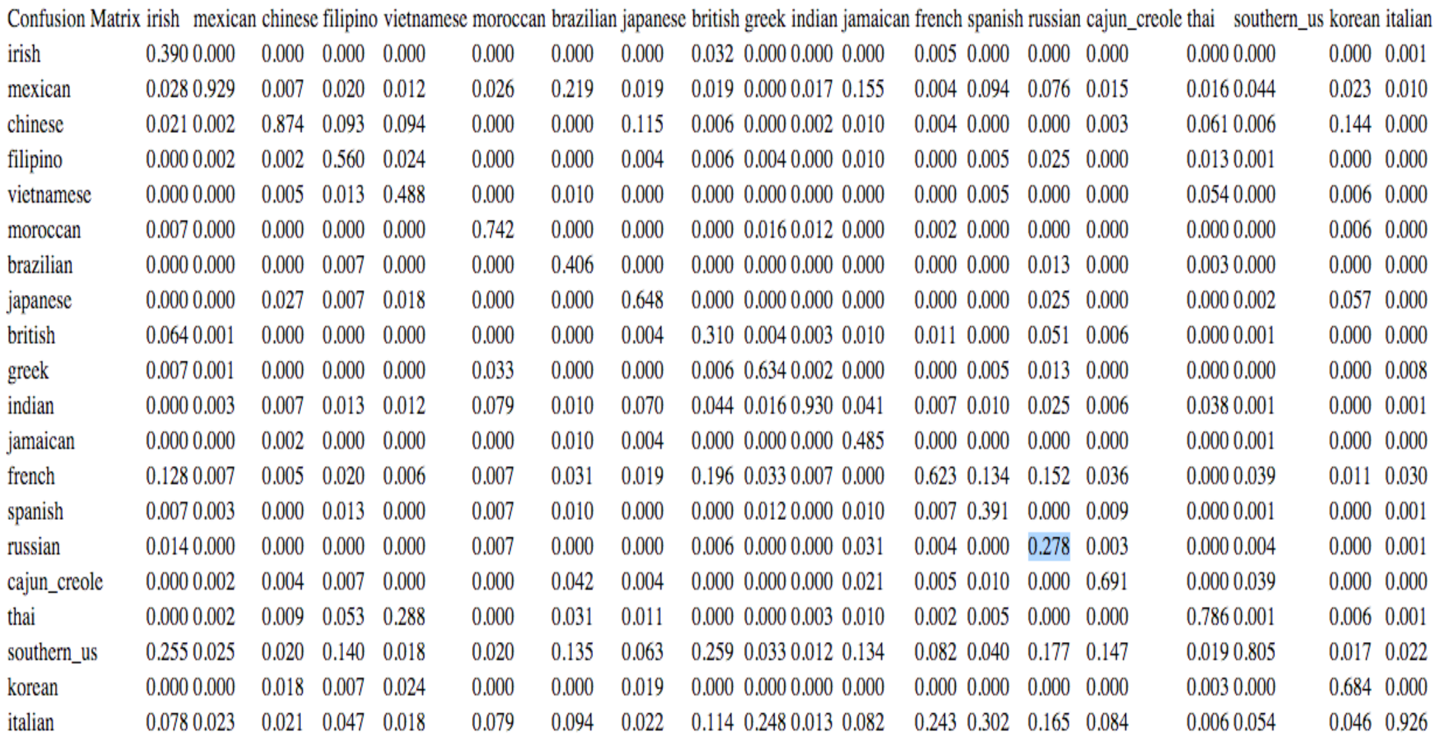
[6] M. N. An anonymous GitHub contributor, Hendrik Hannes Holste. [Online]. Available: https://github.com/HannesHolste/recipe-cuisine-machine-learning/blob/develop/data/food$_{a}djectives.txt$

[7] Simple theano script with 0.79033 on leaderboard. [Online]. Available: https://www.kaggle.com/c/whats-cooking/forums/t/16538/simple-theano-script-with-0-79033-on-leaderboard

[8] A new kind of food science: How ibm is using big data to invent creative recipes. [Online]. Available: http://www.wired.com/2013/11/a-new-kind-of-food-science/

APPENDIX

The confusion matrix where C_{ij} represents the percentage that the classifier classified the recipe as cuisine i but given that recipe was labeled cuisine j



Confusion Matrix	irish	mexican	chinese	filipino	vietnamese	moroccan	brazilian	japanese	british	greek	indian	jamaican	french	spanish	russian	cajun_creole	thai	southern_us	korean	italian
irish	0.390	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.032	0.000	0.000	0.000	0.005	0.000	0.000	0.000	0.000	0.000	0.000	0.001
mexican	0.028	0.929	0.007	0.020	0.012	0.026	0.219	0.019	0.019	0.000	0.017	0.155	0.004	0.094	0.076	0.015	0.016	0.044	0.023	0.010
chinese	0.021	0.002	0.874	0.093	0.094	0.000	0.000	0.115	0.006	0.000	0.002	0.010	0.004	0.000	0.000	0.003	0.061	0.006	0.144	0.000
filipino	0.000	0.002	0.002	0.560	0.024	0.000	0.000	0.004	0.006	0.004	0.000	0.010	0.000	0.005	0.025	0.000	0.013	0.001	0.000	0.000
vietnamese	0.000	0.000	0.005	0.013	0.488	0.000	0.010	0.000	0.000	0.000	0.000	0.000	0.000	0.005	0.000	0.000	0.054	0.000	0.006	0.000
moroccan	0.007	0.000	0.000	0.000	0.000	0.742	0.000	0.000	0.000	0.016	0.012	0.000	0.002	0.000	0.000	0.000	0.000	0.000	0.006	0.000
brazilian	0.000	0.000	0.000	0.007	0.000	0.000	0.406	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.013	0.000	0.003	0.000	0.000	0.000
japanese	0.000	0.000	0.027	0.007	0.018	0.000	0.000	0.648	0.000	0.000	0.000	0.000	0.000	0.000	0.025	0.000	0.000	0.002	0.057	0.000
british	0.064	0.001	0.000	0.000	0.000	0.000	0.000	0.004	0.310	0.004	0.003	0.010	0.011	0.000	0.051	0.006	0.000	0.001	0.000	0.000
greek	0.007	0.001	0.000	0.000	0.000	0.033	0.000	0.000	0.006	0.634	0.002	0.000	0.000	0.005	0.013	0.000	0.000	0.000	0.000	0.008
indian	0.000	0.003	0.007	0.013	0.012	0.079	0.010	0.070	0.044	0.016	0.930	0.041	0.007	0.010	0.025	0.006	0.038	0.001	0.000	0.001
jamaican	0.000	0.000	0.002	0.000	0.000	0.000	0.010	0.004	0.000	0.000	0.000	0.485	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.000
french	0.128	0.007	0.005	0.020	0.006	0.007	0.031	0.019	0.196	0.033	0.007	0.000	0.623	0.134	0.152	0.036	0.000	0.039	0.011	0.030
spanish	0.007	0.003	0.000	0.013	0.000	0.007	0.010	0.000	0.000	0.012	0.000	0.010	0.007	0.391	0.000	0.009	0.000	0.001	0.000	0.001
russian	0.014	0.000	0.000	0.000	0.000	0.007	0.000	0.000	0.006	0.000	0.000	0.031	0.004	0.000	0.278	0.003	0.000	0.004	0.000	0.001
cajun_creole	0.000	0.002	0.004	0.007	0.000	0.000	0.042	0.004	0.000	0.000	0.000	0.021	0.005	0.010	0.000	0.691	0.000	0.039	0.000	0.000
thai	0.000	0.002	0.009	0.053	0.288	0.000	0.031	0.011	0.000	0.000	0.003	0.010	0.002	0.005	0.000	0.000	0.786	0.001	0.006	0.001
southern_us	0.255	0.025	0.020	0.140	0.018	0.020	0.135	0.063	0.259	0.033	0.012	0.134	0.082	0.040	0.177	0.147	0.019	0.805	0.017	0.022
korean	0.000	0.000	0.018	0.007	0.024	0.000	0.000	0.019	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.003	0.000	0.684	0.000
italian	0.078	0.023	0.021	0.047	0.018	0.079	0.094	0.022	0.114	0.248	0.013	0.082	0.243	0.302	0.165	0.084	0.006	0.054	0.046	0.926