CSE 255- Assignment 2

Bon Appetite:

Prediction of cuisine based on Ingredients

Pooja Bhat
University of California San Diego
pkbhat@ucsd.edu

Sakshi Gupta
University of California San Diego
sag045@ucsd.edu

Tanvi Nabar
University of California San Diego
tgnabar@ucsd.edu

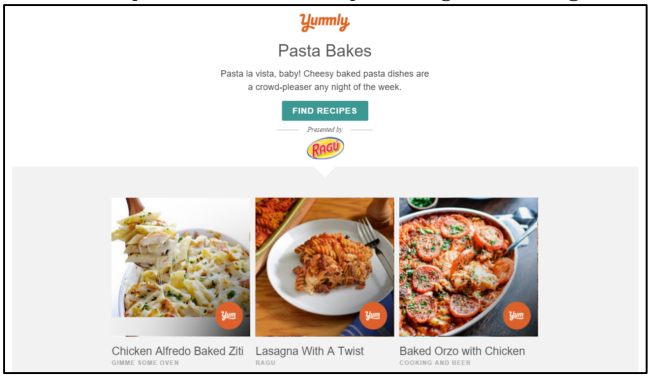
***Abstract*---In this paper, we explore the patterns related to the ingredients of a cuisine. A model has been built in order to predict the cuisine of a recipe given the ingredients used for the recipe.**

Keywords---Multiclass Classification, SVC, TF-IDF, k-Nearest Neighbors, KNN, Cosine similarity, Yummly

1. INTRODUCTION

Food is an integral part of a cultural experience. Each cuisine has its own authentic flavors and recipes passed down generations, which ensure the continuation of the culture in the face of mainstream food. It is important to identify the typically used ingredients, to understand a culture's cuisine. Ingredients are often associated with a cuisine as a result of geographical circumstance, cultural history or both. For example, the Italian cuisine focuses on natural ingredients like tomatoes, garlic cloves, olive oil etc. along with varieties of cheese. The Chinese cuisine mainly includes the ingredients rice, soybeans (soy milk/soy sauce) etc. Indian cuisine focuses on local ingredients like spices, herbs, vegetables and fruits.

Yummly provides a large dataset of all recipes across different cuisines. Using this dataset, we aim to predict the cuisine of a recipe using its ingredients. This task can help compare different ingredients within and across cuisines. We can also use this predictive task to identify common ingredients across cuisines and come up with creative recipes using similar ingredients.



2. DATASET

We are using the dataset provided by Yummly for the What's Cooking? competition on Kaggle. The dataset contains 39774 recipes each a part of one of 20 cuisines, with the following features:

(1) Recipe ID
(2) Cuisine
(3) Ingredients list

Since we do not have many attributes, it is important to perform useful analysis on the list of ingredients and develop a model to help recognize cuisine.

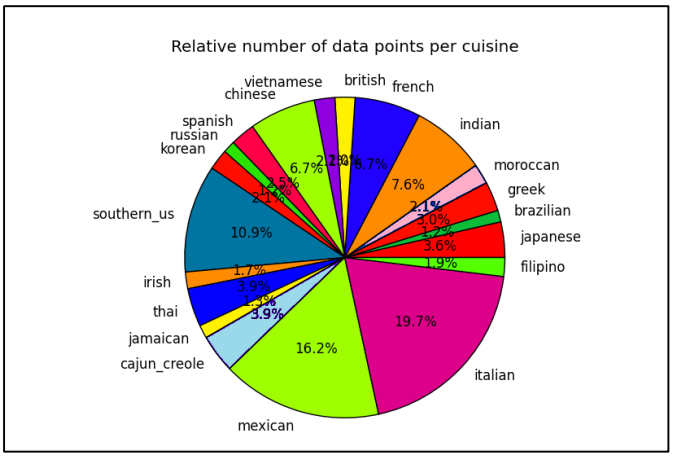
Table2.1: Number of data points per Cuisine

Cuisine	# data points
Brazilian	1175
British	804
Cajun_Creole	1546
Chinese	2673
Filipino	755
French	2646
Greek	1175
Indian	3003
Irish	667
Italian	7838
Jamaican	526
Japanese	1423
Korean	830
Mexican	6438
Moroccan	821
Russian	489
Southern_US	4320
Spanish	989
Thai	1539
Vietnamese	825

3. EXPLORATORY ANALYSIS

We perform initial analysis to help us decide on the models that will work best with the nature of data.

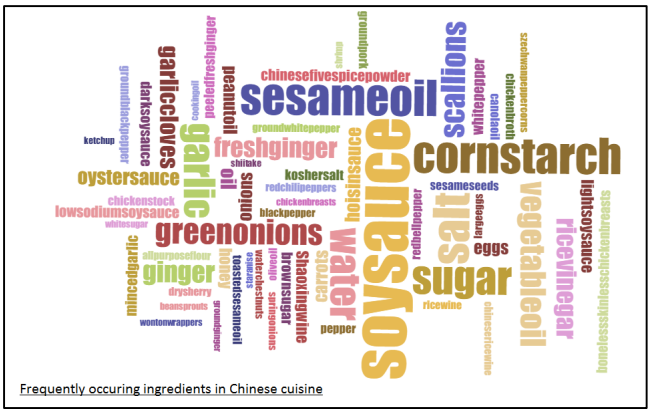
A. Number of data points per cuisine

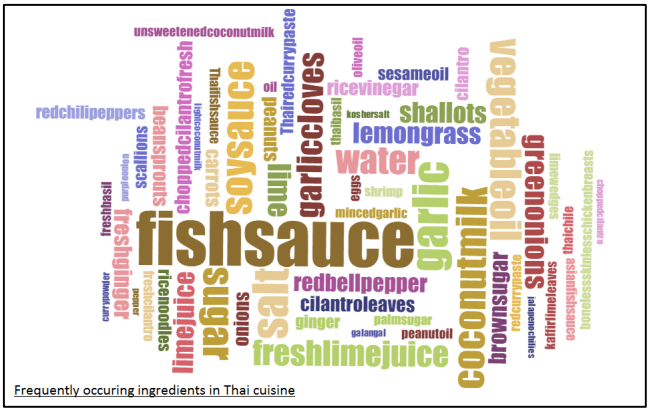


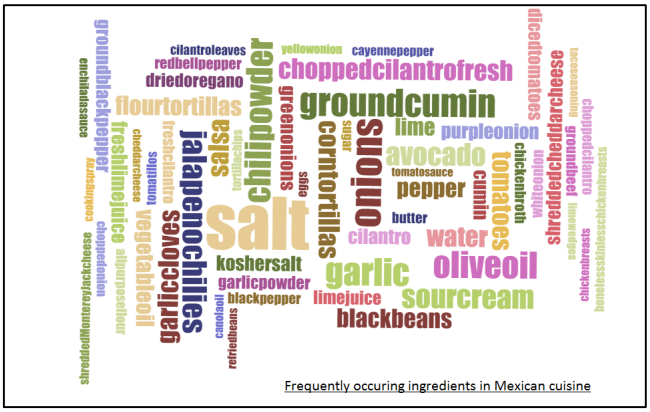
The training set has about 20 percent of Italian cuisine samples. Next most observed are Mexican and Southern US cuisines with 16.2 percent and 10.9 percent of the training samples respectively. On the other hand, certain cuisines like Brazilian and Russian, have very few data points (1.2%).

We need to optimize balanced error measures to avoid errors due to partiality of data, either by reducing the sample set to take an equal number of data sets for every cuisine, or by adding a balancer to neutralize the effect of each cuisine. We use the second method to avoid loss of training data.

B. Most frequently occurring ingredients in cuisines



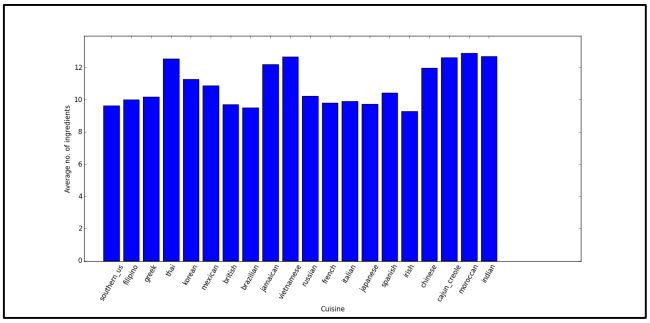




The above graphs list the 20 most frequently used ingredients in the Thai and Chinese cuisine recipes against the number of occurrences in the training set. We can see that *fish sauce* is used frequently in the Thai cuisine, whereas the Chinese recipes include heavy usage of *soy sauce, sesame oil* and *corn starch*. The Mexican cuisine includes *chili powder*, *ground cumin* and *onions*.

It is also observed that certain ingredients occur frequently in most cuisines. Some examples of these ingredients are *salt*, *garlic* and *garlic cloves*. Hence, they are not useful in classifying the recipes. These can be equated to the stop words such as *a*, *and*, *the*, *is*, *of*, *at*, *in*, *on*, etc. in the classic document classification example.

C. Average number of ingredients per cuisine



From the graph, we realize that the number of ingredients do not vary a lot per cuisine (between 9 and 13). Hence, the number of ingredients should not be used as a feature in determining the cuisine.

4. PREDICTIVE TASK

The model predicts the cuisine of a dish given the ingredients used to prepare the dish. It can be seen that the number of Italian recipes is the highest. Hence the baseline code predicts every recipe as belonging to Italian cuisine.

This a multiclass classification problem. The input to the model is the different ingredients used for particular recipes of a cuisine. The output of the model is one of the 20 cuisines: Irish, Mexican, Chinese, Filipino, Vietnamese, Moroccan, Brazilian, Japanese, British, Greek, Indian, Jamaican, French, Spanish, Russian, Cajun Creole, Thai, Southern US, Korean and Italian.

The dataset is split into training set (85%) and validation (15%) set. The prediction model is built on the training set. The validity of the model's prediction is checked by running on the validation set.

5. DATA PREPROCESSING

Our predictive task is similar to a document classification task i.e. the task of assigning a document to one or more classes or categories, where each document here is analogous to the set of ingredients and the classification label is the cuisine.

A. Vector Space Model

We have used a Vector Space Model/Term Vector Model to represent each document as a vector of identifiers.

We tried the following ways to represent vector matrix:

(1) Term binary: This is a naïve method where we check the occurrence of the name of the cuisine in the ingredients list. For example, the Irish cuisine has an ingredient *Irish Whisky* which will set the Irish cuisine term to 1. Similarly, the ingredient *Indian spice* can contribute to the recipe be attributed to the Indian cuisine. This method isn't relevant to many ingredient lists and hence we chose to discard it.

(2) Term frequency: In this method, we compute the number of times an ingredient occurs in the list of ingredients. However, term frequency is not always proportional to the relevance of the ingredient. As we see, *salt* is the most frequently occurring ingredient, but since it appears across all cuisines, it is a less helpful token in the prediction task.

(3) Term frequency – inverse document frequency (TF-IDF): We compute the product of two terms - term frequency and inverse document frequency (an inverse function of the number of ingredient list in which a particular ingredient occurs). A high weight in TF-IDF would then mean a high term frequency (in the given data point) and a low document frequency of the term in the whole collection of ingredients.

Each row in our TF-IDF matrix is a vector of the TF-IDF weights for each ingredient in the data set. Here, we divided our input ingredient set into tokens and calculated the TF-IDF for these tokens.

B. Tokenization

We divided our train set of ingredients into a set of tokens, and performed the following tuning methods over our baseline model for better performance.

- *Lemmatization*: We represented each token with its lemma, by using the Word Net Lemmatizer from the NLTK library. E.g. Plurals such as *carrots* were converted to their singular form (*carrot)*. Verbs as *chopped* in their past tense were converted to their stemmed word *chop*.
- *Stop words*: We noticed that certain ingredients like salt, water, onions etc. appeared frequently and uniformly across all cuisines. In order to achieve a better performance, we found the most frequently occurring words across all cuisines and included them in the list of stop words and pruned these from the token list. We also pruned adjectives like *light, dark, large, small* etc. as they are less relevant to the predictive task.
- *Unigrams and Bigrams*: Our initial baseline models used unigrams as tokens. We later used a mixture of unigram and bigram model and observed a better performance with this model. We used thresholds such as minimum document frequency (min_df= 0.007), maximum document frequency (max_df= 0.57) and maximum features (=2500) to prune our list of tokens

6. MODELS

We tried the following models after building the TF-IDF matrix:

1. Baseline model

Our training data set is biased towards Italian cuisine, and hence one baseline, or rather benchmark we used was predicting all test cuisines as Italian.

2. Support Vector Classification

We chose a linear SVC model as it scales better to a large number of samples. Also it supports both dense and sparse input, is helpful to us as our TF-IDF matrix is sparse, and the multi class support is handled according to a one-vs-the-rest scheme.

3. K-Nearest Neighbors (KNN) Classification:

In this model, for each TF-IDF vector of the test data, we found the k training vectors that are closest to it, where we defined 'closeness' in terms of Euclidean distance, and applied the label of the largest cluster among the k selected tuples. We also used the cosine similarity as a measure of distance, and found that it performed slightly better.

4. Centroid clustering and cosine similarity model:

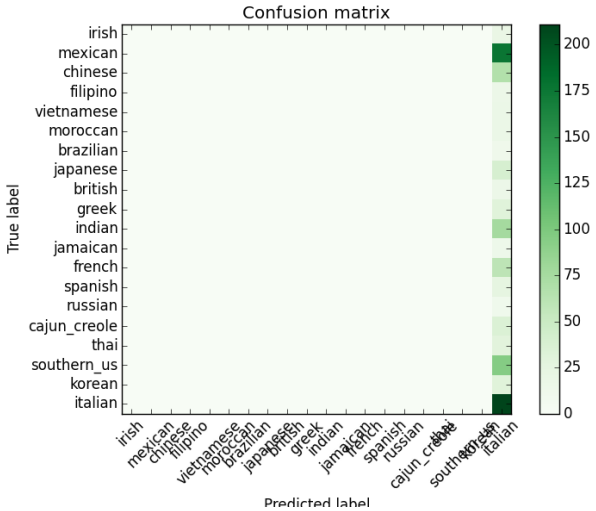
We modified the above model for better performance as follows. We found the centroid of the TF-IDF vectors for each cuisine, and using cosine similarity as a distance measure, we classified the test tuple by the label of the centroid TF-IDF vector it is closest to.

7. RESULTS

The following results were obtained:

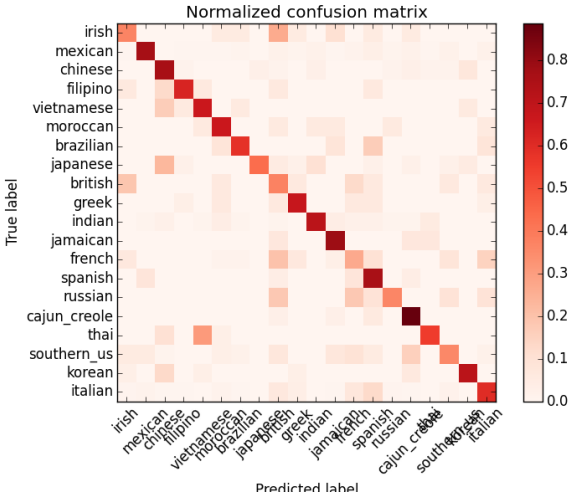
1. Baseline Model

In this model, the baseline model gives an average accuracy of 0.19268. This is expected because about 20 percent of our data had Italian labels. Only the column with Italian predicted cuisine has matching data points.

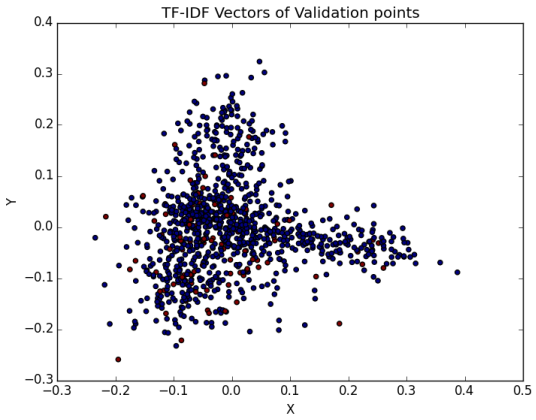


2. Support Vector Classification

The accuracy we got for this method on the validation set is 0.6528.



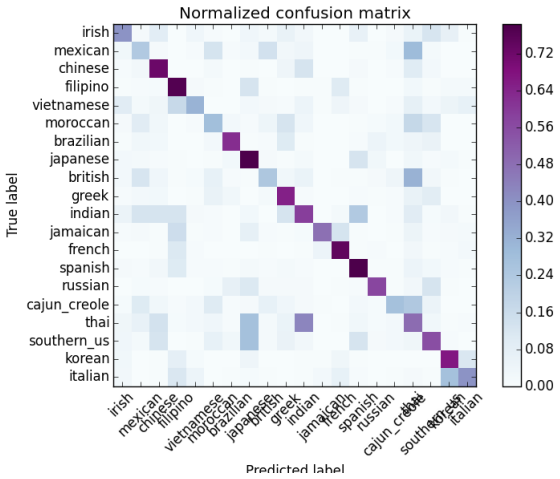
We see that a majority of the data points in the normalized confusion matrix lie on the diagonal in the matrix which means that the actual cuisine and predicted cuisine is the same.



Here, we performed dimensionality reduction using Principal Component Analysis with 2 components, to denote the TF-IDF vector on a 2D plane. The *blue* scatter points are the TF-IDF values of data points that were correctly classified, whereas the *red* points signify the misclassified points.

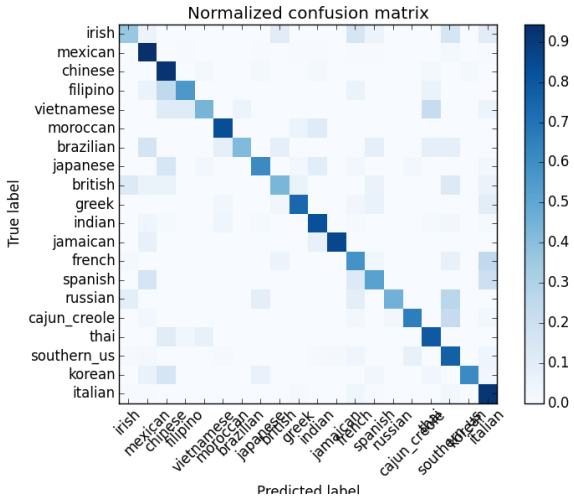
3. K-Nearest Neighbors(KNN) Classification:

The accuracy we got for KNN classification is 0.619. Here we checked the data point with 4 of its nearest neighbors. Hence, the K nearest neighbors approach tanks as compared to linear regression.



4. Centroid clustering and cosine similarity model:

The accuracy we get by using cosine similarity is 0.7882 on the validation set.



So in conclusion, KNN gives us the worst result, followed by Linear Regression. Multiclass classification Cosine Similarity gives the best results.

8. REFERENCES

[1] Kaggle *(http://www.kaggle.com)*

[2] Yummly Dataset *(http://www.yummly.com/)*

[3] Term frequency Inverse document frequency – Wikipedia (*https://en.wikipedia.org/wiki/Tf%E2%80%93idf*)

[4] Document Clustering with Python - brandonrose.org (*http://brandonrose.org/clustering*)