Recipe Type Classification Using Ingredient Lists

Project Proposal

Huaxin Pang
Zayd Hammoudeh

CS256 – Fall 2015

Table of Contents

1. Introduction 1
2. Project Goal 2
3. Abstracting an Ingredient List into a Data Point 2
4. Proposed Solution Overview 2
   4.1. Ingredients Preprocessor 3
   4.2. Learning Algorithms 3
      4.2.1. K-Nearest Neighbors 4
         4.2.1.1 Inter-Recipe Distance Measures 4
            4.2.1.1.1 Overlap Coefficient 4
            4.2.1.1.2 Modified Value Difference Metric 4
         4.2.1.2 Determining "K" 5
      4.2.2. Naïve Bayesian Network 6
      4.2.3. Ensemble Method 6

List of References 7

Recipe Type Classification Using Ingredient Lists

1. Introduction

Launched in 2010, Yummly.com is a social media platform that is food and cooking centric; it bills itself as the "largest, most powerful, and most helpful food site in the world". Yummly allows users to search, organize, and share recipes based off user-specific requirements (e.g. all recipes without mustard, dietary restrictions, etc.) [1].

In September 2015, Yummly posted a dataset of recipes on the data science website Kaggle [2]. Each recipe record in the dataset consists a list of ingredients, a class value (i.e. cuisine type), and an identification number. Records are formatted using JavaScript Object Notation (JSON) as shown in figure 1.


```
{
    "id": 24717,
    "cuisine": "indian",
    "ingredients": [
        "tumeric",
        "vegetable stock",
        "tomatoes",
        "garam masala",
        "naan",
        "red lentils",
        "red chili peppers",
        "onions",
        "spinach",
        "sweet potatoes"
    ]
},
```


Figure 1 – Example Record for a Recipe of Cuisine Type "Indian"

The Yummly training set is comprised of 39,774 recipes consisting of 6,714 different ingredients spread across 20 international cuisine types. The following is a list of the cuisine types along with the number of recipes for that type in parenthesis: Brazilian (467), British (804), Cajun/Creole (1,546), Chinese (2673), Filipino (755), French (2646), Greek (1,175), Indian (3,003), Irish (667), Italian (7,838), Jamaican (526), Japanese (1,423), Korean (830), Mexican (6438), Moroccan (821), Russian (489), Southern US (4,320), Spanish (989), Thai (1,539), and Vietnamese (825).

2. Project Goal

Develop an algorithm that can classify a recipe's type of cuisine (e.g. "Italian", "Indian", "Cajun/Creole", etc.) based solely off a list of ingredients.

3. Abstracting an Ingredient List into a Data Point

Each record in the training set represents a single recipe. A record of the type shown in figure 1 does not naturally lend itself well to a machine learning algorithm since the core data (i.e. ingredient list) is relatively unstructured. As such, a structure must be applied to the record in order to adapt it to traditional algorithms.

The simplest logical structure for this data set's records is transform the ingredients list into an N-dimensional vector (where N is the number of ingredients in the entire data set). Hence, each attribute in the vector would be binary value representing whether a particular ingredient is present in the specific recipe record. Using this approach, it then becomes possible to use machine learning algorithms that support nominal attributes.

4. Proposed Solution Overview

Figure 2 shows the overall structure of our planned implementation for this project. The training set is a text file of JSON records as described in section #1. An unseen record will be a list of ingredients, and the predicted cuisine type will be one of the 20 possible values in the Yummly dataset. Our plans for the two remaining components (i.e. the ingredients pre-processor and the learning algorithm(s)) are described in the subsequent subsections.

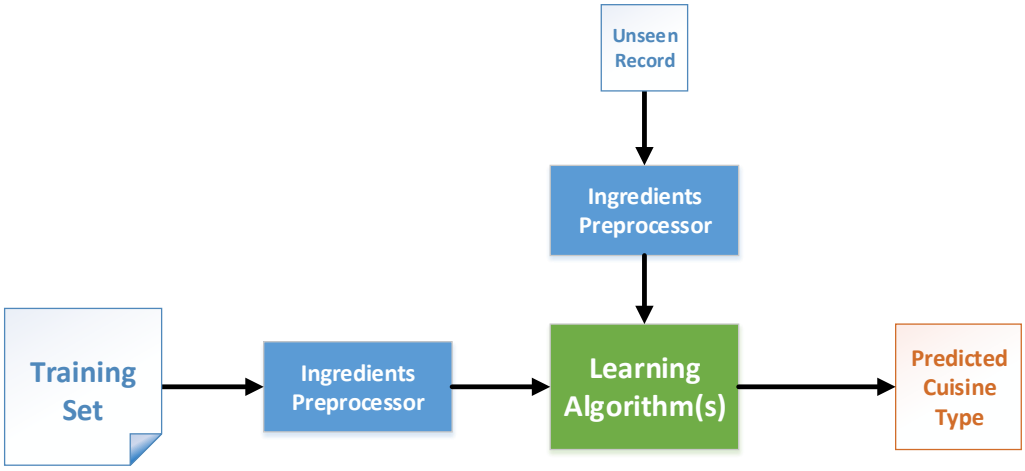

Figure 2 – Planned Project Implementation Structure

4.1. Ingredients Preprocessor

In order to achieve the best results out of any machine learning algorithm, it is important that all of the training and testing records be as stylistically consistent as possible. For example, it is often considered disadvantageous for a training dataset to represent the same piece of information using multiple different notations. Unfortunately, the Yummly dataset has significant variation; this arises from the fact that the ingredient lists were written by countless different people, who each have their own notational style. For example, the ingredient "cilantro" appears as "chopped cilantro fresh", "fresh cilantro", "cilantro leaves", "chopped cilantro" as well as just "cilantro" depending on the recipe. Similarly, garlic appears as "garlic cloves", "minced garlic", "chopped garlic", "crushed garlic" and simply "garlic". As such, for our algorithm to have the best results, the ingredients information will need to go through a preprocessor to strip away terms that lead to a distinction without a difference. Example terms we are considering removing include size (e.g. "medium", "large", "extra large"), temperature (e.g. "hot", "warm"), freshness (e.g. "frozen", "fresh"), and preparation (e.g. "chopped", "sliced", "ground", "crushed") descriptors.

When preprocessing the data, we will need to show care not to remove important descriptive terms. For example, three ingredients in the dataset are: "clam sauce", "chipotle sauce", and "bean sauce". Any approach we use should not simplify all of these ingredients to solely "sauce". Otherwise, false matches of ingredients will occur.

Our planned design for the ingredients preprocessor is two stage. The first stage will be manual curation of the ingredient list where some of the previously mentioned terms (as well as possibly others) are removed from ingredient names. We expect that this approach will remove most of the unnecessary variation in the ingredient record style. If this approach proves insufficient, we plan to optionally explore more complex natural language processing (NLP) algorithms.

4.2. Learning Algorithms

The high dimensionality of the encoding scheme described in section 3 does not lend itself well to many of the machine learning algorithms discussed in class. For example, a decision tree would be prohibitively large in size; similarly, a rule-based classifier would be potentially difficult to construct and would have hundreds or thousands of rules.

In our view, there are three algorithms that at first thought appear the most promising fit for this dataset. The first is a neural network which is well suited for these types of classification problems with large numbers of dimensions. However, given the remaining duration of the semester, we foresee that this approach may be prohibitively complex. The following subsections describe the two remaining learning algorithms we are considering for this project.

4.2.1. K-Nearest Neighbors

We foresee that recipes of the same cuisine type will have high levels of similarity of ingredients; if this holds, then K-NN may be a good fit for determining the cuisine type. What is more, with fusion cuisine becoming increasingly prevalent, K-NN may allow us to predict a set of closely aligned cuisine types when an ingredient list fits well into more than one cuisine category.

To use K-NN for this problem, the two primary challenges that need to be solved are:

1. Distance Metric Definition
2. Selection of K

The subsequent subsections describe our planned approach for these issues.

4.2.1.1 Inter-Recipe Distance Measures

A key component in the K-Nearest Neighbors algorithm is metric used to quantify the difference between two records. The following subsections describe two possible distance metrics we are considering.

4.2.1.1.1 Overlap Coefficient

A simple metric to determine the similarity of two recipes is the number of ingredients that they have in common. For example, if a recipe, R_1 has three ingredients, $\{A, B, C\}$, and if a second recipe, R_2, also has three ingredients $\{B, C, D\}$, then the similarity of these two recipes could be estimated using the Overlap Coefficient [3] to be:

$$Overlap(R_1, R_2) = \frac{|R_1 \cap R_2|}{\min(|R_1|, |R_2|)} = \frac{|\{B, C\}|}{\min(|\{A, B, C\}|, |\{B, C, D\}|)} = \frac{2}{3} \quad (1)$$

One potential downside of the Overlap Coefficient is that it may overemphasize direct matches and may miss more subtle matches. For example, if recipe R_1 contains the ingredient "sea cucumber", it is almost certainly a Chinese recipe, but it is unlikely to have a direct match with many recipes since it is relatively rarely used ingredient. However, sea cucumber is usually paired with other common Chinese ingredients such as ginger and soy sauce. As such, while it is not a direct match with either of those ingredients, it may still be able to determine similarity. To determine more subtle relationships, a more complex metric would be needed.

4.2.1.1.2 Modified Value Difference Metric

The Modified Value Difference Metric (MVDM), shown in equation (2), is commonly used to quantify the difference between two nominal attribute values; we will modify this approach slightly by using MVDM to quantify the difference between two ingredients I_1 and I_2. Hence, if two ingredients (e.g. basil and tomatoes) are commonly associated with the same cuisine type (e.g. Italian), then the distance between the two ingredients will be low (minimum 0). In contrast, if two ingredients (e.g. Kimchi and garam masala) are never in the same type of cuisine, then the distance will be high (with a maximum value of 2).

$$d(I_1, I_2) = \sum_{i=1}^{k} \left| \frac{n_{1,i}}{n_1} - \frac{n_{2,i}}{n_2} \right| \quad (2)$$

For our algorithm, k is the number of cuisine types (20); n_1 and n_2 are the total number of recipes that have ingredients I_1 and I_2 respectively. $n_{1,i}$ is the number of recipes of cuisine type i that have ingredient I_1; similarly, $n_{2,i}$ is the number of recipes of cuisine type i that have ingredient I_2.

The training set will allow us to use MVDM to calculate the similarity between every possible pair of ingredients in the entire dataset. This inter-ingredient distance information will then be used to calculate the distance between two recipes $R_1$and R_2 using equation (3).

$$d(R_1, R_2) = \frac{1}{M \cdot N} \cdot \sum_{i=1}^{M} \sum_{j=1}^{N} d(I_1, I_2) \quad (3)$$

M represents the number of ingredients in recipe R_1 while N represents the number of ingredients in recipe R_2. (3) calculates the sum of the distances between each pair of ingredients. To ensure that recipes are not penalized for having more ingredients, we normalize the sum of sums by the total number of ingredient pairs (i.e. $M \cdot N$).

4.2.1.2 Determining "K"

Depending on what time allows, there are different possible approaches that we can use to determine the optimal value of K. One option would be to use cross validation to divide the data set into a set of d disjoint partitions. For each training run, we could look at how accurately different values of K classify the data. Using that information, the simplest option would be to select a single value of K for our algorithm. An alternative option we may choose to explore is to allow the algorithm to use multiple values of K, each of which has its own weight; after the class value has been selected for each value of K, the algorithm could then select the class value with the highest total weight as the final result.

4.2.2. Naïve Bayesian Network

Each ingredient's presence (or absence) from the recipe may have an influence on the category of the recipe. Using the record structure described in section 3 (i.e. each record is an attribute of whether each known ingredient is present in the recipe), then it may be possible to use Naive Bayes classification to calculate the posterior possibility for each possible cuisine type.

The intuitive foundation for this approach is based on the concept that certain ingredients (and/or combinations of ingredients) may be more important for specific types of cuisine. For example, if a recipe contains ginger, garlic and green onions, it is very likely that it is Chinese. What is more, certain ingredients may almost never be used in certain cuisines types (e.g. cardamom in Russian cuisine). Using this information, you may be able to use each ingredient's conditional probability to predict cuisine type.

4.2.3. Ensemble Method

Sections 4.2.1 and 4.2.2 described two possible approaches we are considering to predict an ingredients list's cuisine type. While these techniques are different, they should not be considered mutually exclusive. Rather, using the Ensemble Method, we may be able to leverage both techniques collaboratively to predict the cuisine type.

List of References

[1] "Introducing the Ultimate Cooking Tool," Yummly. [Online]. Available at: http://www.yummly.com/how-it-works/. [Accessed: Nov-2015].

[2] "What's Cooking?,". [Online]. Available at: https://www.kaggle.com/c/whats-cooking. [Accessed: Nov-2015].

[3] "Overlap Coefficient," Wikipedia. [Online]. Available at: https://en.wikipedia.org/wiki/overlap_coefficient. [Accessed: Nov-2015].