CS-401 APPLIED DATA ANALYSIS

Authors:
Jonathan CHANSIN
Etienne DUBOIS
Tom SUTER

December 19, 2017

***Abstract*—Our project's goal is to extract the combination of ingredients and try to find an interconnection between them in order to map which combinations of different ingredients blend the best together to give tasteful meals. We used K-NN regression to find proximity within ingredients in order to create recipes, and, we used a linear regressor to predict different nutrients as fats, carbohydrates, proteins and cholesterol.**

I. Introduction

A cooking recipe is a list of diverse ingredients which are combined together for preparing a meal. Nowadays, there exists millions of cooking recipes all around the world found in many cooking books and cooking websites.

However, all recipes aren't necessarily always great or tasteful, indeed the choice and the combination of ingredients is actually extremely important. We believe that ingredients in a cooking recipe are like harmonies in music, separated they are pleasant to hear, but when perfectly combined, harmonies can come up with something special (Schubert' Symphonies). Therefore, we are convinced that new combinations and new cooking recipes are yet to be found, and there are no better ways than using Data analysis coupled with a bit of Machine learning to achieve that.

To do so, a dataset provided with a wide variety of cooking recipes collected (web scrapped) on more than a dozen different cooking website, is used to extract the list of ingredients for each recipe and try to find the interaction between one another. Basically, we are building recipes based on the proximity between ingredients.

The data analysis is divided on many sequential parts, first of all, the text data preprocessing in which recipe's ingredients are cleaned. The second part consists of building the homogeneous quantity matrix and the co-occurrence matrix. Finally, the last part consists of applying a machine learning model which is K Nearest Neighbors Regression to find the proximity between ingredients.

As related work we found : The Silicon gourmet which is a project using training on neural network to generate cooking recipes inspired by Tom Brewe's neural network-generated recipes. However, results are, let's say interesting and sometimes even funny (for example, "*Completly Meat Chocolate Pie*") [Covucci D., 2017]. Along with this project, many others are also trying to use machine learning to create recipes.

II. Data collection & description

The dataset provided contains 65499 different recipes collected using web scrapping on 14 different websites such as "*allrecipes.com*", "*cookeatshare.com*" or "*food.com*". For each recipe, there are many features like the url of the website, the nutrients information which are total calories, carbonate calories, fat calories, protein calories, sodium mass and cholesterol mass. Also, the dataset contains the title of the recipe and a text in string format with the description of the ingredients.

The nutrients information and the text with the ingredients are separated into two Dataframes. Text Dataframe is preprocessed and ingredients are extracted for each recipe with their quantity and preparation techniques (like sliced, chopped,...), the steps to achieve that are described in the following sections. After preprocessing and extraction, our dataset contains 863 different ingredients.

As recipes come from distinct websites, it is interesting to know the proportion for each websites. Thus, the histogram below shows the percentage of each website where the recipes were collected. A significant number of recipes were web scrapped from 2 principal sites "*allrecipes.com*" and "*food.com*"

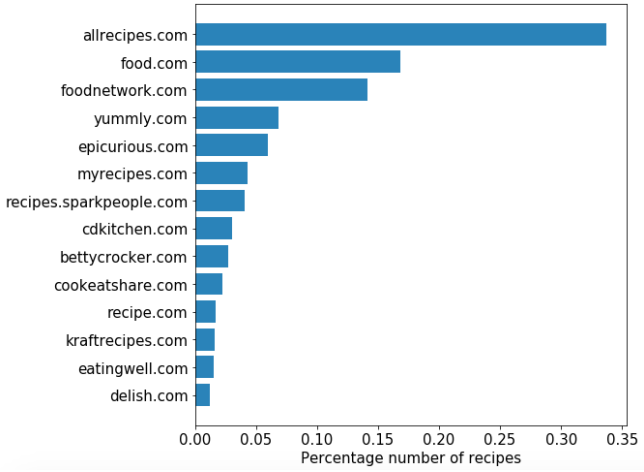

Figure 1: Number of appearance for each ingredient in dataset

III. Data preprocessing

Data preprocessing is the pillar of any text base project, and it wasn't an exception for our project. As recipes are from different websites, they were presented in different ways. Some of them were well structured and concise, while others had long descriptions with typos, special characters, and missing quantities.

We had to design multiple filters in order to create a data frame, allowing us to perform analysis on it. Obviously filters were settle step by step, by identifying mistakes in the processed recipes. We first wanted to find a way to identify relevant information concerning the ingredient that were in between parenthesis. Here we talk about quantities, or a unit. The example 1 in the annexe section illustrates this situation. If there was no relevant information in the parenthesis, its content was deleted. The second filter that we implemented was to reduce noise in a recipe. Indeed, recipes can be a bit messy, they can contain mistakes (typo), abbreviations or even special characters (like # or @). We dealt the special characters problem using the regex library, knowing that some of them had to be kept. Indeed in example 2 and 3 in the annexe section shows why we couldn't just delete all special characters. By deleting "/" we would get 34 cup white sugar instead of 3/4 and in the example 3, we would get 25 c maple syrup instead or 0.25 c. Later we created a filter to convert those values in a way python can process them. 3/4 becomes 0.75 and .25 becomes 0.25.

Then we shorten each ingredient description in order to have only useful information. We used the nltk library from python to delete all determinants, article, preposition, adjectives. With the verbs we created a list of techniques that we used in order to create new recipes. Additionally we create two list. A unit list, that regroups all the cooking measures with its respective abbreviations and a list of ingredients by performing web scraping on

"*http://www.bbc.co.uk/food/ingredients/by/letter/a*".

We then lemmatize all the ingredients from the recipes and from the ingredient list. The lemmatizer, changes the words to their root words, thus plurals and other form words were put in the same word format.

IV. Data Extraction & Analysis

A. Extraction ingredients

After the pre process we created a list of ingredients and their associated quantities. The main difficulty resides in extracting those quantities. Multiple units are given (cups, ounce, gallon, grams, etc) and they all have to be converted to grams. We made the approximation to always convert volume to grams using the density of water. We realized that this assumption can lead to major errors with non water ingredients. The real difficulty appears when we had to deal with unit-less ingredients. Typical examples are, 1 package of spaghetti, 3 apples etc. The value of one unit of each ingredient had to be known. We first though of web-scrapping it. We realized not many websites gave an exhaustive lists of those units. A first choice would be to use the data in "*http://www.eurofir.org/food-information/food-composition-databases-2/*". However the ingredients are hard to find and mainly are processed food. We looked for something more precise and related to our ingredient list. We found another website that gave us better results but still corresponds to only 300 ingredients. The rest had to be done manually. Even though every ingredients were analyzed, some problems remains. For example a "wrap" can appear as a single unit or as a package which makes it hard to put a single weight.

Having this informations we created a dictionary for each recipe containing their ingredients and the associated quantities. We end up with a total of 862 ingredients appearing among 46277 recipes.

B. Outliers

Many recipes ended up having enormous quantities of ingredients. We decided to delete them because most of the time they are due to misinterpretation of our algorithm or simply a bad recipe. In order to find them, we first analyzed a few distribution using boxplot:

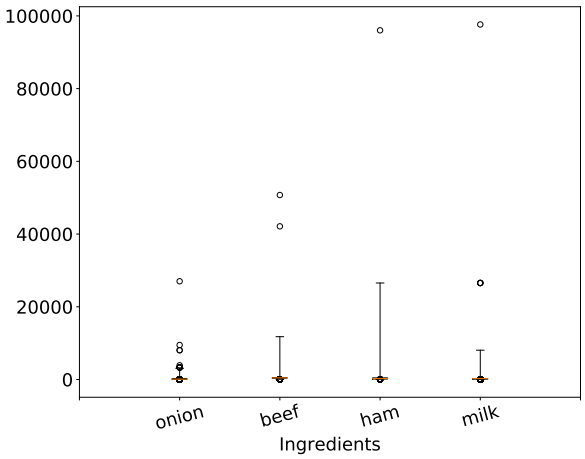

Figure 2: Outliers analysis

The top worst elements, appearing in highest quantities were coconut, milk, broccoli, ham, beef. Here is an overview of their distribution.

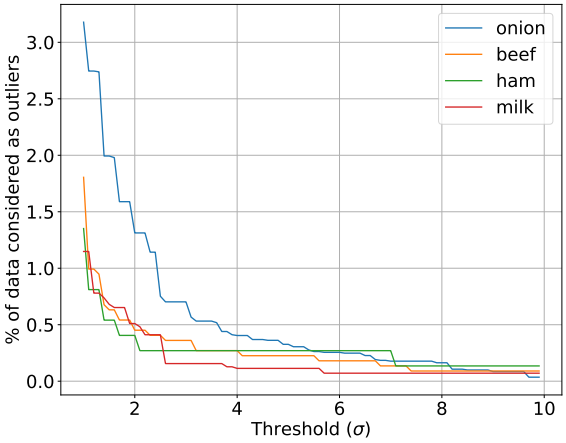

Figure 3: Percentage of outliers as function of the threshold $x_{outlier} > mean(x) + T\sigma$

We decided to keep 9 times the variance as a threshold limit. It seems that the ingredients have approximatively the same percentage of outliers using this threshold. So Furthermore we can see on fig. (3) that it is a reasonable limit even for the worst distribution. However, using this threshold some outliers remain, annexe example (4). Thus we reduced the threshold to 7 σ. After this process we end up with 45844 recipes.

C. Co-occurrence matrix

Once we got the data set M, we create the co occurrence matrix by transforming the data set matrix into a ingredient by ingredient matrix, recording which ingredients appeared together. To do so, we create the matrix X where each element X_{ij} = 1 if $M_{ij} > 0$ and $X_{ij} = 0$ if $M_{ij} = 0$. Then we just multiply XX^T and we got the co-occurrence matrix.

D. Ingredient quantity ratio

In order to create new recipes, we scaled the quantities of the ingredients according to the quantity of the main ingredient. We assumed that the following relation hold $\frac{E(x)}{E(y)} = \frac{x}{y}$. E(x) being the mean quantity of the main ingredient in the dataset, x the quantity of the main ingredient in the recipe to be created, E(y) the mean quantity of the ingredient to be scaled and y the quantity to be scaled of the ingredient that our algorithm adds to the recipe.

We decided to scrap the serving in order to improve the results. But finally we chose quantities over servings.

V. Results & Findings

Once we got our final matrix containing all the ingredients with the quantities for each recipe, we decide to perform some analysis in order to check if what we have been doing was reasonable. First we decide to plot the number of appearances of the top 20 ingredients. We were relieved to see in figure 4 that all the ingredients of the list, are of common use in the kitchen

and that it's normal to have them in the top 20. The salt appears in 20000 recipes which is about the half of our data set. Going a little bit in deep, we were surprise to see the onions above water or eggs. But as onions can be used in multiple ways it make sens that it appears more often than water or eggs.

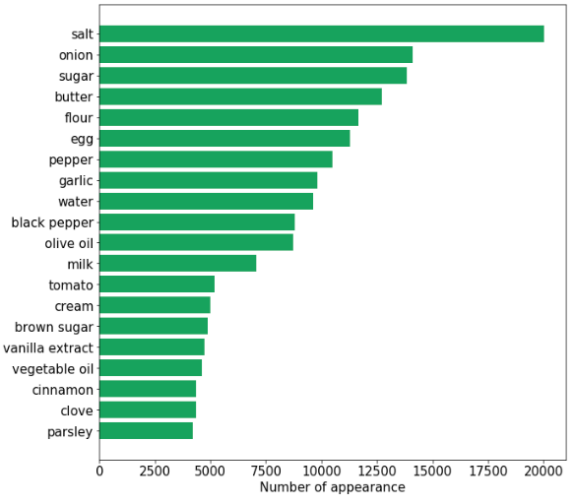

Figure 4: Number of appearance for each ingredient in dataset

We then wanted to see the distribution of each category of ingredients. This would allow us again to check for coherence and to check about our possibilities to create vegetarian, vegan, and healthy recipes. We expected that spices to aper the most with the category "others". Spices because it's very hard to cook something with taste without using spices, and the "others" because it's contains all the resting ingredients that are not in a precise category. The rest we expected to be quiet equilibrated. Figures 5 show us that we had a good intuition for the spices and for the other category. However the rest of the categories varies a lot. We expected to see more meat/fish in the recipes but here it's not the case. The reason may be that, it's unlikely to mix to different kind of mead/fish in the same recipe, while other categories as vegetables or sides, are more likely to be combined in a recipe. Additionally we found that there are 64.2% percentage of vegetarian recipe and 19.7% percentage of vegan recipe.

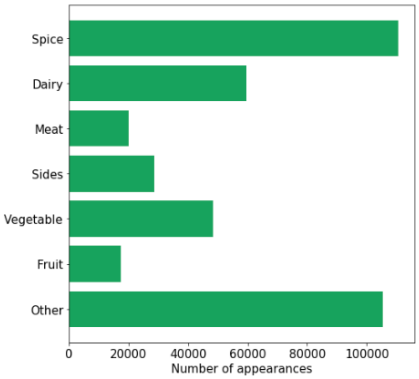

Figure 5: Number of appearance for each category of ingredient

To go further, we decided to plot the number of ingredients per recipe. This helped us to estimate the range of ingredients to use while creating new recipes. Figure 6 tell us that the number of ingredients per recipe follows a positive screwed normal distribution, with the mean at 7 to 8 ingredients per recipe. We believe it's a reasonable amount of ingredients for a recipe, knowing that mostly of the recipes are European. However we expected less recipes below 4 ingredients. Obviously there are some recipes like milkshakes that don't contains a lot of ingredients and that would explain a part of the number of recipes below 4 ingredients. However, the main reason is that inevitably ingredients were filtered in the preprocessing part. The lists filters are the main reasons. Indeed all ingredients with typos were automatically deleted as they where not identified as belonging to our ingredient list.

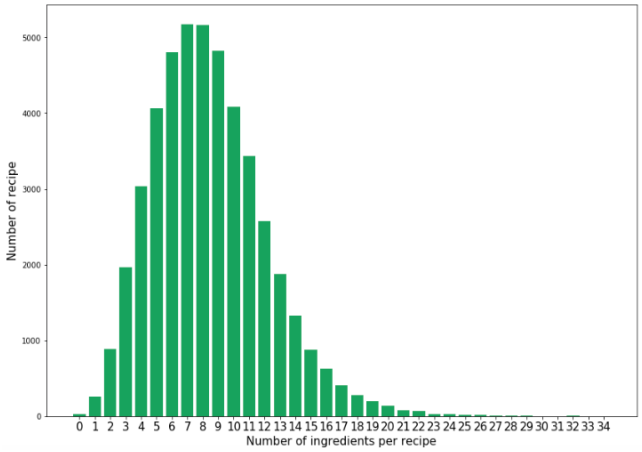

Figure 6: Number of recipes which have specific number of ingredients

VI. Recipe Creation

The co-occurrence matrix gives the number of time two ingredients appear together in the same recipe which gives an idea of the interconnection between ingredients. Indeed, if two ingredients happen to occur many times together, it could mean that these two ingredients are great together in a recipe and lead to a tasteful meal.

Therefore, the first idea was to construct proximity list for each ingredient based on the number of occurrence with others, thus for each ingredient a list of ingredients classified from the highest number of occurrence to the smallest. As a result, our 863 different ingredients have a list of 10 ingredients which it occurs the most together. For instance for *chicken* the list is [1.(onion 903), 2.(salt 709), 3.(pepper 597), 4.(garlic 586), 5.(water 500), 6.(black pepper 486), 7.(flour 374), 8.(butter 374), 9.(celery 362), 10.(olive oil 318)]

A. K-NN regression

However even if, the previous result seemed good and usable, using the k-NN regression on the co-occurrence matrix would allow to improve the list of proximity ingredients. Although, co-occurrence reflect how two ingredients are used together, a positive (or negative) association between two ingredients does not necessarily signify a direct or indirect interaction, as it can result simply from the ingredients having similar uses. For

example, it is possible that onions occur a lot with chicken because another ingredient is strongly related to chicken which could cause the occurrence to be high.

K-NN regression is a non-parametric method (unsupervised learning) used for classification and regression which consists of the k closest training examples in the feature space. So it is a Unsupervised learner for implementing neighbor searches based on a similarity measure (e.g., Hamming distance) $D_H = \sum_{i=1}^{k} |x_i - y_i|$ The result is different than before: *chicken* : [chicken stock, pea, rice, bean, sausage, shrimp, sage, pork, rosemary, noodle]

B. Special recipe creation

From there a simple algorithm create a recipe from an ingredients (e.g chicken) chosen by the user. It take the list of 10 for this ingredient and return 2 ingredients (e.g rosemary and pea) from that list which are picked randomly with probability of 1/10 and added to the recipe.Then the specific list for 2 these ingredients are added to the first list where the ingredients (rosemary and pea) previously picked are dropped. From there, 2 others ingredients are picked and the algorithm continues until the recipe is completed.

As a recipe would not be appropriate if it is composed from only spices or only meats, therefore, a number maximal (randomly from 1 to 2, except for meat where the number maximal is always set to 1) for each category is set. So the number of ingredients and ingredients per category for each recipe change constantly.

Besides, a technique of preparation and the quantity of each ingredient are added in the recipe.

Finally, a jupyter notebook interface is available (shown on the figure below) where the user is able to select a specific ingredient, the quantity of this ingredient as well as a restrictive condition on the recipe such as vegan, vegetarian of healthy recipe. It returns the recipe with all the ingredients and also, existing recipes with URL address from the dataset containing selected ingredient.

Figure 7: Example of recipe created

VII. Nutrient informations Prediction

We added an extra feature on the recipe prediction: the ability to predict the number of carbohydrate, protein and fat. In order to do that we needed to fit the values of the number of gram of carbohydrate, protein, fat for one gram of the ingredient. We used linear regression on the whole data set with slightly different specifications. But the error we minimized with stochastic gradient descent the following loss (called mean square logarithmic error):

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}[\log(\mathbf{wX_i}) - \log(Y)]^2 + \gamma \sum \mathbf{w}_i^2 \qquad (1)$$

γ regulates the weights penalty and was set between 1 and 0.01 depending on the case. We also forced the weights to be positive as no ingredient can have negative grams.

Even if we used a few major assumptions to infer the right quantities, we had pretty good results on the nutrient of each ingredient except a few anomalies: 0 fat for bechamel sauce.

VIII. Conclusion

Creating recipes is not an easy task, as there are a lot unsuspected difficulties to deal with. We believe we faced most of the difficulties in the right way. The key part of the project was the processing of all the recipes in order to get a matrix with the quantities of each ingredient in each recipe. We had to identify the outliers to reduce noise in the matrix. Outliers are due to bad recipes, but also due to our filtering that is not perfect. Also, we made an assumption which is that all ingredients have the same density than water. Unfortunately we couldn't find a website to scrap the exact density of each ingredient. This means that if a recipe contain 3 oz of butter, we will attribute the weight of 3 oz of water which is not very precise. All those approximations, made the matrix with the quantities not very accurate. Still, we were able to create a linear regressor that predicts carbo-calories quite accurately. The creation of recipes wasn't influenced by the quantity matrix, indeed we used the co occurrence matrix and apply k-NN regression on it, to find proximity within ingredients. The quantities of each ingredient in the new recipe were calculated using the ratio of the mean of each ingredient according to 1 serving. Our objective was achieved, however, we believe we will not be able to create the best food symphony as the majority of the recipes are European, and from USA. To create the best food symphony one need recipes from all around the word to find unsuspected association of ingredients.

IX. References

L. (2016, February 29). The Silicon Gourmet: training a neural network to generate cooking recipes. From http://aiweirdness.com/post/140219420017/the-silicon-gourmet-training-a-neural-network-to

Covucci, D. (2017, March 31). A scientist is trying to teach a neural network to cook-and the results are hilariously bad. From https://www.dailydot.com/unclick/neural-network-recipe-generator/

X. Annexe

1) Recipe 104: Raw

1/2 cup butter, softened|1 cup sugar|4 eggs|1 cup flour|1/2 teaspoon salt |1 (16 ounce) can chocolate syrup | 1 teaspoon vanilla|1/4 cup butter, softened|2 cups powdered sugar|2 tablespoons creme de menthe|6 ounces chocolate chips|1/4 cup butter

Recipe 104: Parenthesis Filter

1/2 cup butter, softened|1 cup sugar|4 eggs|1 cup flour|1/2 teaspoon salt |16 ounce chocolate syrup 1 teaspoon vanilla|1/4 cup butter, softened|2 cups powdered sugar|2 tablespoons creme de menthe|6 ounces chocolate chips|1/4 cup butter

2) Recipe 105: Raw

1 1/2 cups all-purpose flour | 1 teaspoon baking soda | 1 teaspoon baking powder | 1/2 teaspoon salt | 3 bananas, mashed | | 3/4 cup white sugar | 1 egg, lightly beaten | 1/3 cup butter, melted | 1/3 cup packed brown sugar | 2 tablespoons all-purpose flour | 1/8 teaspoon ground cinnamon | 1 tablespoon butter

3) Recipe 107: Raw

3 c fresh brussel sprounts | .25 c maple syrup salt to taste|pepper to taste

4) Recipe 45135: Remaining outlier

1412 ozs yellow hominy | 8 ozs corn |2tbsps lemon juice | 1 clove clove garlic|12 tsp ground cumin|14 tsp cayenne

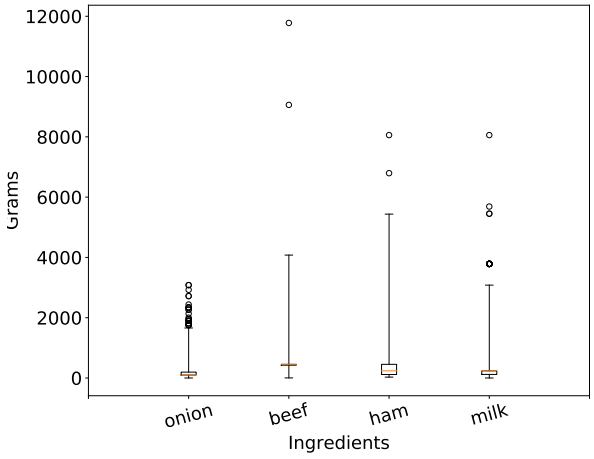

Figure 8: Distribution of the analyzed ingredient after the outliers threshold