# Clustering Recipes

Eli Rosmarin

May 15, 2021

**Abstract**

In this paper we explore how natural language processing and various clustering techniques can be used together to classify unlabelled recipe data. I outline the methodology in which I parse the text-based data, as well as how various parameters such as vocabulary size were optimized. We explore the shortcomings of K-means clustering which ultimately led to the development two novel graphical representation.

## 1 Introduction

Everybody loves food! It is only a natural extension of technology to attempt to make things we already love better. As a result, training machines to perform various tasks given recipe data has become a unique area of applied machine learning research. Researchers have used neural networks to predict and recommend food ingredient pairings [1], and K-nearest neighbors to create recipe recommendation systems [2]. Given the textual nature of recipe data, it is a great candidate for natural language processing tasks.

For this project I found a dataset which contains over 100,000 recipes scraped from the internet [3]. Each recipe contains a list of ingredients, instructions, and some of the recipes have associated images. I wanted to leverage the textual nature of the dataset and combine it with clustering methods explored in class. The versatility of clustering methods enable them to be applied to a wide array of data types to achieve various classification tasks. This lends well to natural language processing problems. For the purposes of this paper the focus is on the ingredients of recipes. As such, references to a recipe solely refer to the ingredients text, and not the instructions.

## 2 Problem

The average person has about 10,000 taste buds. We perceive taste in five ways: salty, sweet, sour, bitter, and umami. It is thanks to our taste buds and the way in which we perceive taste that we are able to differentiate food. It is the reason chicken tastes like chicken and a chocolate chip cookie tastes like a cookie. A given recipe contains any number of ingredients. Every ingredient in and of itself has a unique, and often complex flavor profile. Even basic ingredients stimulate our taste buds in multiple areas; olive oil is both sweet and bitter.

A computer, on the other hand has 0 taste buds, and the diverse flavors in a dish make classifying recipes much more challenging than simply asking ”is this salty?”.
The question at hand is can recipes be clustered based on a computational interpretation of their ingredients? Since a recipe is complex it is unwise to split data by the 5 pillars of taste, but rather makes more sense to attempt to cluster based on the part of a meal. Now other problems arise on how many “parts” of a meal are there. Do we expect there to be a difference between hors d'oeuvers and appetizers. Do we want different clusters for chicken, steak, and vegetables? The ultimate answer is we want to create something where the resulting clusters speak for themselves. Since this data is unlabelled we want to create some partitioning such that a rational human looking at the clusters would say “these groupings make sense!”.
The problem at hand is two-fold. The first part is a natural language processing task, in which we must develop a way to interpret the recipe data. The second part is a clustering task in which we can use the natural language processing in an attempt to find similarities between different recipes to create clusters.

## 2.1 Methodology

**Parse Ingredient Data**
The list of the ingredients for each recipe in the original dataset was represented as a list of strings.Each word in the list was parsed for clarity with the following steps:

1. Remove all non-alphabetic letters
2. Make all lowercase
3. Remove commons words using pythons NLTK stopwords [4]
4. Use Porter stemming algorithm [5] to remove common endings

Once every word in the list had been parsed, every word in the list was joined to create one string. The following is an example of what the parsing does:

> **Original ingredients for ”Slow Cooker Chicken and Dumplings”**:
> '4 skinless, boneless chicken breast halves', '2 tablespoons butter','2 (10.75 ounce) cans condensed cream of chicken soup', '1 onion, finely diced', '2 (10 ounce) packages refrigerated biscuit dough, torn into pieces'
>
> **Parsed ingredients for ”Slow Cooker Chicken and Dumplings**:
> 'skinless boneless chicken breast halv tablespoon butter ounc can condens cream chicken soup onion fine dice ounc packag refriger biscuit dough torn piec'

Note that the misspelled words in the parsed text such as “ounc” and“packag” are not errors but results of the Porter stemming algorithm. “packages”, “packaged” and other forms of the word “package” all result in “packag” with use of the algorithm. This allows our natural language processing model to capture more words using a smaller vocabulary.

**Create Bag-of-Words Representation**
Once every recipe in the dataset has been parsed and stored we can create a vocabulary. We can think of a vocabulary as a a vector of predefined size, $m$. The vocabulary for this project has been designed to contain both unigrams and bigrams. This allows common two-word terms or ingredients such as ‘condensed cream’ to be included (although it would be stored as ‘condens cream’ since the vocabulary is being built off of parsed data).
For each of the $n$ recipes in the dataset we can create a vector $v_i$ where $v_{(i,j)}$ gives us information on the $i^{th}$ recipes use of the $j^{th}$ word in the vocabulary. A simple information metric is as follows:

$$v_{(i,j)} = \begin{cases} 1 \text{ if recipe } i \text{ contains vocabulary word } j \\ 0 \text{ otherwise} \end{cases}$$

The limitation of this metric is that it simply tells us whether a word in the vocabulary exists in the recipe, but we also want to capture data on the importance of the word. The Term Frequency Inverse Document Frequency (TFIDF) is used instead. TFIDF is used to reflect how important a word is to a recipe in a collection of recipes.[6]

$$v_{(i,j)} = tf_{i,j} * log\frac{N}{df_j}$$
$$tf_{i,j} = \text{number of occurences of } j \text{ in } i$$
$$df_j = \text{number of recipes containing j}$$
$$N = \text{total number of recipes}$$

It is extremely unlikely that a given recipes ingredients will contain every word in the vocabulary, and the larger the vocabulary the more 0 elements will be in the vector. As such choosing the right vocabulary size is an important task. For preliminary purposes the vocabulary size was set to 500, but we will later explore modifications which were made.

**Graphical Representation (1)**

$$M_{(i_1,i_2)} = \begin{cases} \sum_{j=1}^{m}(1 \text{ if } v_{i_1} = 1 \text{ and } v_{i_2} = 1) \text{ if } i_1! = i_2 \\ 0 \text{ otherwise} \end{cases}$$

Here we are using the simple counting metric for values in v, not the TFIDF metric. While we inherently lose the information from the TFIDF metric by not using it, we gain information on the similarity of recipes. This matrix represents a graph where each node is a recipe and the weights of the edges between two nodes is equal to the total number of vocabulary words the two recipes have in common.

By counting the number of common vocabulary elements between two recipes we have eliminated the value of the information from TFIDF, but have gained value by creating a graphical representation of our data with which can be used to develop the Laplacian which we can then run spectral clustering on.

**Graphical Representation (2)**

For our second graphical representation we create a large sparse matrix, $M$. Here $TFIDF_{(i,j)}$ is equivalent to $v_{(i,j)}$ using TFIDF.

$$\underset{n\times n}{R} = \begin{bmatrix} 0 \end{bmatrix}$$

$$\underset{n\times m}{B} = \begin{bmatrix} TFIDF_{(1,1)} & TFIDF_{(1,2)} & \dots & TFIDF_{(1,m)} \\ TFIDF_{(2,1)} & TFIDF_{(2,2)} & \dots & TFIDF_{(2,m)} \\ \vdots & \vdots & \ddots & \vdots \\ TFIDF_{(n,1)} & TFIDF_{(n,2)} & \dots & TFIDF_{(n,m)} \end{bmatrix}$$

$$\underset{m\times m}{I} = \begin{bmatrix} 0 \end{bmatrix}$$

$$\underset{(n+m)\times(n+m)}{M} = \begin{bmatrix} R & B \\ B^T & I \end{bmatrix}$$

M is a sparse symmetric matrix which is a graphical representation of how different recipes are connected to similar ingredients. This matrix was used to develop the Laplacian which we can then run spectral clustering on.

## 3 Results

**K-means Clustering and PCA**

Using the initial bag of words representation, we can perform K-means clustering on the vectors $v$. Each word in the vocabulary is a feature, resulting in 500-dimensional space as this is our initial vocabulary size. Since distance loses meaning in high dimensions we perform principal components analysis to reduce the data to two dimensions. We separate the data into four clusters. four clusters was chosen given our original hypothesis that we believe that the clusters will have some correlation to different parts of a meal. 4 dishes would constitute a meal with appetizers, entrees, desserts, and a side dish.

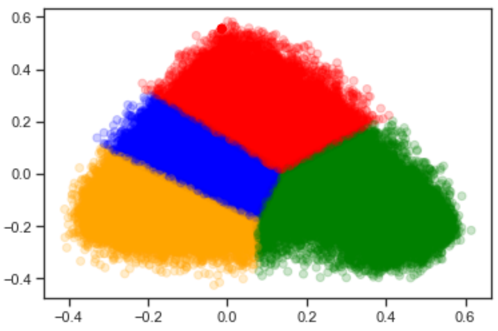

Figure 1: PCA Clusters

We can see graphically that the PCA clusters are not truly clusters, but appear more so as arbitrary partitions. Using PCA makes it so that our features no longer hold meaning in the way the original 500-dimension data did, and since the 500-dimension data was sparse less than 7% of the original data variance is captured in PCA. Due to all of these limitations the graphical representations were developed.

**Spectral Clustering - Graphical Representation (1)**
Given the density of the first graphical representation, we had to reduce the number of recipes in the graph to 100. Similar to the K-means clustering we assume four clusters, since there are no "natural" clusters in the graph. What was found was that that one of the four clusters was mostly dessert foods, one cluster was exclusively chicken dishes, and the other two were more or less a mixed bag. While the interpretability of this graphical representation is nice, its density does not make it an ideal candidate for spectral clustering, and the computational complexity deems it essentially useless when dealing with anything other than a small dataset. This led to the development of the second graphical representation.

**Spectral Clustering - Graphical Representation (2)**
Given the sparsity of the second graphical representation we can calculate the smallest eigenvalues and use the max eigengap heuristic to pick an optimal number of clusters.[7][8]

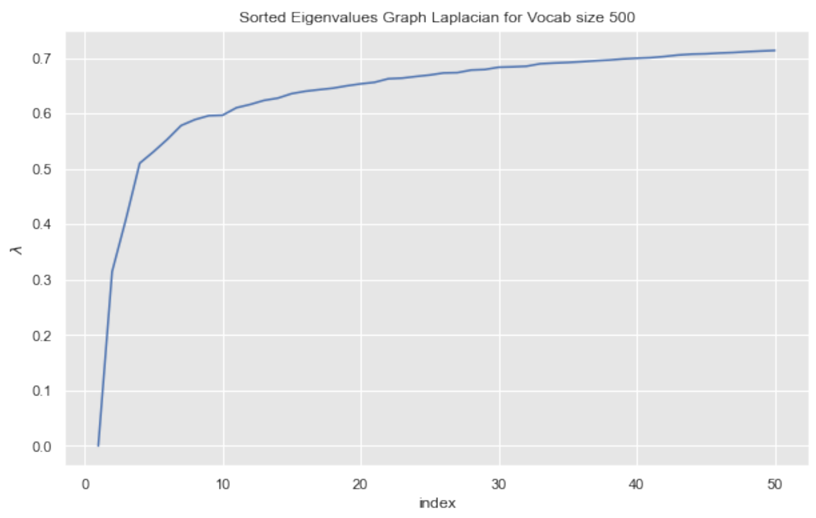

Figure 2: 500 word Vocabulary

With 100,000 recipes and a vocabulary of 500, the max eigengap occurs at index one (between the smallest and second smallest eigenvalue). If we were to use the eigengap heuristic here we would have one cluster, which we obviously don't want. We can still run spectral clustering here with four clusters. Looking at the clusters in this case there is one cluster that is very heavy on desserts and other sweet foods, similar to what happened when using the first graphical representation. The goal here is to programatically find an optimal number of clusters, which has not yet been achieved. In attempting to increase the max eigengap we can modify the vocabulary size.

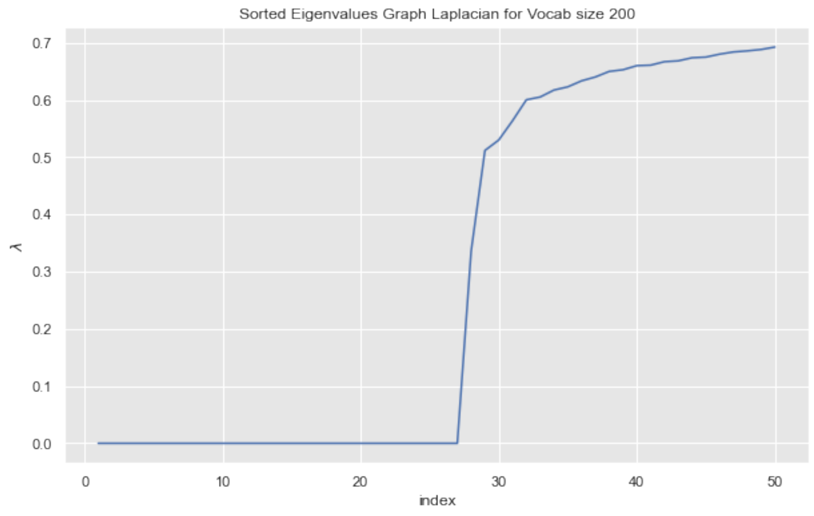

Figure 3: 200 word Vocabulary

Cutting the vocabulary down to 200 words, increases our max eigengap index to 27. Using 27 clusters yields interesting results. We end up with one cluster with almost all of the recipes, and 26 cluster with just one recipe. When looking at the 26 recipes with their own cluster, we find they're all very unique. They all have minimal ingredients, which results in them having no words in the vocabulary and thus being entirely disconnected from the rest of the graph. Take, for example, this recipe for "Leftover Pizza Rolls", which only has 3 ingredients: Leftover pizza, Leftover Chinese food, and Mozzarella. These ingredients (maybe with the exception of mozzarella) are definitely not ones you normally see in a recipe. Let's see what happens when we cut out these recipes. Through an iterative process of cutting recipes that end up being single nodes with no edges as the vocabulary decreases, we end up with a very small vocabulary of 32, and 99,820 recipes.

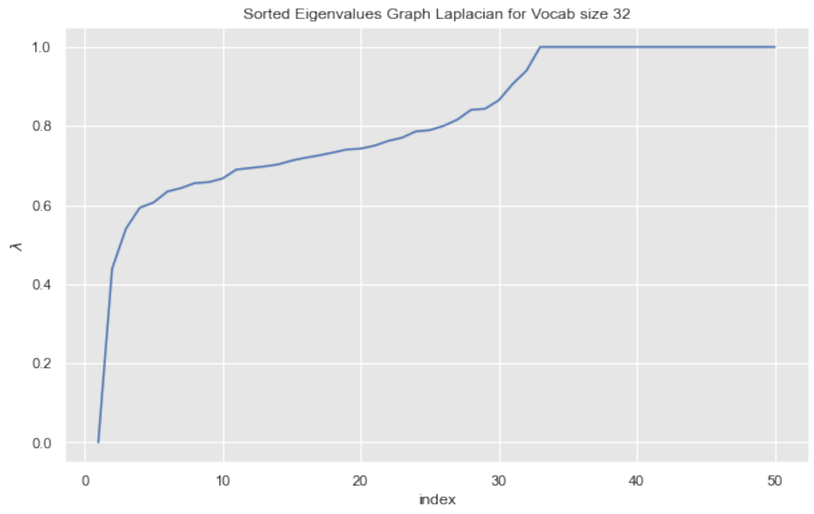

Figure 4: 32 word Vocabulary

even with a significantly decreased vocabulary size our eigengap heurestic remains one. In changing the vocabulary size to find an optimal number of clusters, the eigengap heurestic was unable to achieve its anticipated outcome. Even though the heurestic did not give us the results we were hoping for, running four clusters showed promise of classification capabilities for this graphical representation.

## 4 Discussion

This project was a lot of trial and error, and the results were not as glamorous as I had initially hoped for. A lot of the work which I did brings into light questions regarding how human logic and machine learning interact for unsupervised tasks on unlabelled data. Had the data been labelled, it is possible that my first attempt with K-means clustering created clusters that actually performed well. Whatever that labelling might be, if it even exists, I was unable to create a model which satisfied the clustering I had hoped to see, where it was divided by course. As outlined earlier in the problem section of the paper, as a human we are able to look at a recipe and say this is a dessert or this is an appetizer, which is a luxury that does not exist within a machine. With that being said, through my various experiments comes some valuable developments.

## 5 Conclusion

In this paper we established a framework for classifying recipes which combines natural language processing with clustering methods. I think there is a lot of room for work to be done in the future. The framework outlined here is a good starting point, and I hope that the various methods in this paper along with their successes and shortcomings can help others who might venture to solve similar tasks. In terms of future work I believe there are several key areas of improvement which could be made here. Firstly, I think labelling this data would be greatly beneficial. With over 100,000 recipes in the dataset this would be a fairly burdensome task, but could prove worthwhile in measuring model accuracy. I actively chose not to include instructions as part of the natural language processing task, because I thought that the vocabulary used in the instructions is fairly uniform across all types of food. With that being said, this provides a clear area which could be looked into. One could use the exact same framework outlined in this paper, but create a vocabulary using the recipe instructions. The recipe instructions will include the ingredients, but might miss out on some measurement data included in the ingredients list. If one is concerned about this they could explore using both the ingredient and instruction data.

For now I think it's fair to say that our 10,000 taste buds are still smarter than a computer.

## References

[1] D. Park, K. Kim, Y. Park, J. Shin, and J. Kang, "Kitchenette: Predicting and recommending food ingredient pairings using siamese neural networks," *CoRR*, vol. abs/1905.07261, 2019. [Online]. Available: http://arxiv.org/abs/1905.07261

[2] N. Bushra and M. Hasan, "Quicklycook: A user-friendly recipe recommender," in *Machine Learning and Metaheuristics Algorithms, and Applications*, S. M. Thampi, L. Trajkovic, K.-C. Li, S. Das, M. Wozniak, and S. Berretti, Eds. Singapore: Springer Singapore, 2020, pp. 245–254.

[3] R. Lee, "Recipe box," Mar 2017. [Online]. Available: https://eightportions.com/datasets/Recipes/

[4] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[5] P. Willett, "The porter stemming algorithm: then and now," *Program*, 2006.

[6] A. Rajaraman and J. D. Ullman, *Data Mining.* Cambridge University Press, 2011, p. 1–17.

[7] J. C. Orduz, "Getting started with spectral clustering." [Online]. Available: https://www.kdnuggets.com/2020/05/getting-started-spectral-clustering.html

[8] W. Kong, Z. Sun, C. Yang, G. Dai, and C. Sun, "Automatic spectral clustering based on eigengap and orthogonal eigenvector," *Tien Tzu Hsueh Pao/Acta Electronica Sinica*, vol. 38, no. 8, 2010.