Seungchan Lee, Daniel Gunther, Vince Hamill
CSCE 470 project

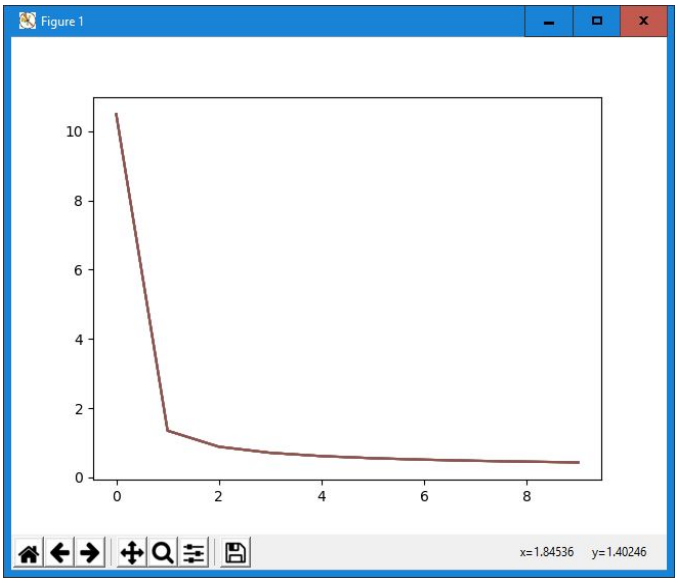
1. Project Motivation
   At first our plan was to cluster menus based on their text. We wanted users to use our program which would take in data from the menus of a bunch of restaurants and cluster restaurants and tell the users which restaurants might be favorable or disfavorable to their taste buds. We decided to switch our plan to cluster recipes of foods instead of menus instead as we found it difficult to use menus to cluster data together.

2. Overall Approach
   Our approach was to cluster recipes based on the name of the food and the ingredients used. In a recipe, typically the ingredients that appear at the front of the recipe are more essential or special parts of the recipe. Therefore, instead of using K-Means to cluster the recipes, we developed our own algorithm which would account for the order in which the ingredients appear.

3. Implementation Details
   Our algorithm consisted of several steps which are listed below.

   1. **Getting an unweighted vector representations of the data**
      Data is read from a file specified by the run configuration. To supply a data file, you simply pass it as an argument value when running the program. After the data is read from the given file, it is converted to a python dictionary, and the terms are normalized. Normalizing the terms simply consisted of converting all of the characters to lowercase, and removing whitespace and then beginning and end of the string. In addition to normalizing the terms, the ingredients are stored in a list, and all of the weights are assigned a value of one.

   2. **Converting the unweighted vectors to weighted vectors**
      After obtaining unweighted vector representations of the data, the terms are assigned a weight. Before assigning a weight to the terms, all of the unique ingredients are removed from the unweighted vectors. When assigning weights to the ingredients, we used a formula that would value the first ingredient very highly, and then weight the next ingredients less and less as shown in the following graph.



3. **<u>Choosing centroids</u>**
   Instead of randomly choosing centroids, we chose centroids to be as spread out as possible. To do this, we choose the first vector as a centroid. After choosing the first centroid, we choose the next centroids to be as far as possible from our current list of centroids.

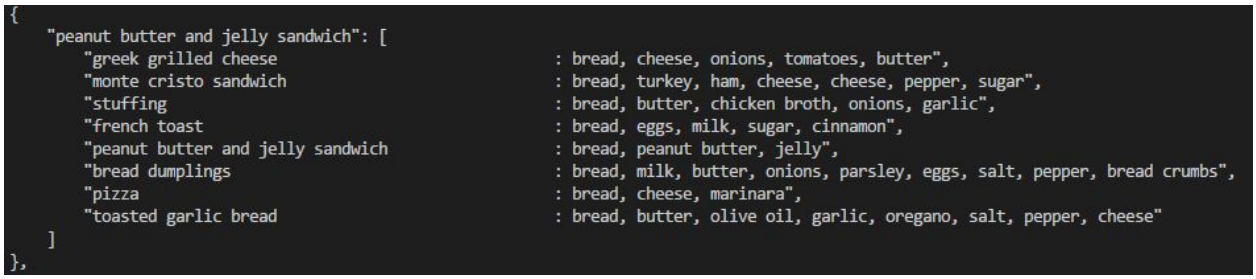
4. **<u>Assigning vectors to centroids</u>**
   After finding the centroids, we iterate through all of the vectors and find the centroid that is closest to the vector. We assign each vector to the cluster which corresponds to that centroid.

4. Dataset Crawled or Used
   For our dataset, we used a JSON file called food.json which listed the names of several food entrees as keys along with the set of ingredients which make them up. This dataset contains 236 items. The set of ingredients for each item gets treated as a bag of words. The program builds up an overall vocabulary vector for all items which we used to cluster the food entrees.

5. Evaluation Settings
   Unfortunately, no one in our group is an expert in evaluating benchmarks and therefore we do not have a gold standard for the data we have, so it is impossible for us to calculate a normalized mutual information score for how well our program clustered our data. We must take an eye test on the resulting clusters to see if our program worked properly or not, seeing if for example similar foods with mostly similar ingredients such as steak

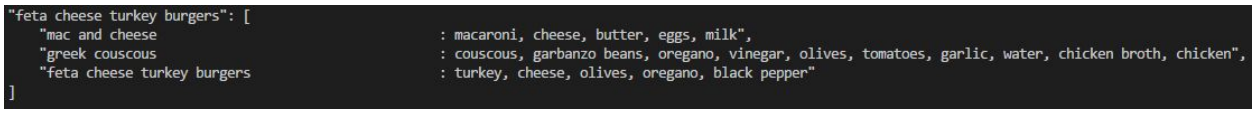
fajita tacos or crunchy tacos are in the same cluster or not.

6. Results



```
{
    "peanut butter and jelly sandwich": [
        "greek grilled cheese                          : bread, cheese, onions, tomatoes, butter",
        "monte cristo sandwich                         : bread, turkey, ham, cheese, cheese, pepper, sugar",
        "stuffing                                      : bread, butter, chicken broth, onions, garlic",
        "french toast                                  : bread, eggs, milk, sugar, cinnamon",
        "peanut butter and jelly sandwich              : bread, peanut butter, jelly",
        "bread dumplings                               : bread, milk, butter, onions, parsley, eggs, salt, pepper, bread crumbs",
        "pizza                                         : bread, cheese, marinara",
        "toasted garlic bread                          : bread, butter, olive oil, garlic, oregano, salt, pepper, cheese"
    ]
},
```

As shown in the picture above, our program has clustered many of the similar foods in our JSON file together. A look at the results of our program will show that most of the items in any of the clusters have similar first or second ingredients, which is our desire as if dishes with similar primary ingredients should be somewhat similar to others.



```
"feta cheese turkey burgers": [
    "mac and cheese                          : macaroni, cheese, butter, eggs, milk",
    "greek couscous                          : couscous, garbanzo beans, oregano, vinegar, olives, tomatoes, garlic, water, chicken broth, chicken",
    "feta cheese turkey burgers              : turkey, cheese, olives, oregano, black pepper"
]
```

In the picture above, we see that the three foods under centroid “feta cheese turkey burgers”, do not have any overlapping similar first or second ingredients. This is due to the few recipes in our data where their first or second ingredients that do not have any other overlapping first or second ingredients as other recipes, so our program groups them together.

7. Error Analysis

There were a few random outliers in our clusters that looks like they shouldn’t be in the same cluster, such as roasted okra and Greek tzatziki which does not have similar ingredients being the same cluster. These random outliers are due to the relatively small data set that we use which makes our program group recipes that use rarely used ingredients such as yogurt with other recipes. Another reason for possible outliers could be due to bad data. As our program relies heavily on the ordering of ingredients of our foods to cluster data, if a recipe was to have a non essential ingredient ahead of an essential one, that could also mess up the clustering.

8. Conclusion

As a whole, our project has performed much better than we have expected. If given more time, we may have been able to fix the minor persisting issues, but our program has achieved our initial goals and objectives we set when starting this project.

9. Contributions

    a. **Daniel Gunther (submitted the .zip file)**
    Implemented the clustering algorithm, worked on part 2 and 3 of the report, and contributed to the presentation slides.

    b. **Vince Hamill**
    Data collection/report/presentation

    c. **Seungchan Lee**
    Data collection/report/presentation