Enhancing Recipe Recommendations through Embedded Clustering Approaches

Annette Lopez Davila

July 12, 2023

1. Abstract

Food recommendation systems play a crucial role in assisting discovery of new recipes for users based on preference, ingredients, and health. In this paper, we present a recipe recommendation algorithm that leverages embedding with clustering & similarity analysis for ingredient-based input. This approach has been used for several recommendation systems recently with results that compete with state-of-the-art recommendation systems. We will use multi-modal data to understand further similarities between recipes and ingredients and conduct exploratory analysis of both the data and the model with further preprocessing and fine-tuning. The performance of our model will be compared to state-of-the-art recommendation systems using precision, recall, F1 score, and NDCG in future steps. Further, we will cross-examine the generated ingredient recipe list with existing recipes both in the dataset and beyond. Through integration with smart devices, this project aims to assist users in making informed decisions about what to cook.

2. Introduction to Recipe Recommendation Systems

The Internet currently has over 10,000 cooking websites, providing users with an overwhelming amount of recipes to search through. Creating a recommendation system can assist users with such filtering. Filtering features within recommendation systems can include nutritional values, type of cuisine, cooking processes, etc.[6]. Because there is no limit on ingredients used for recipes and few ratings for recipes exist, challenges for preference-based recommendation can include the cold-start and sparse data problems.

2.1 Background

As user access to smart devices and instant information retrieval continues to rise, the importance of diet preferences and simplified meal planning has also grown. Consequently, efficient recipe recommendation systems are necessary to cater to these evolving needs. Although extensive information can be found on the internet, it can be overwhelming for users to search for what they are looking for, let alone vocalize their preferences. Studies have shown that the excess of options on the internet leads to more frustration, making user-based recommendation systems a vital solution for simplifying decision-making processes with curated personalized content.

This paper seeks to address common research questions associated with food recommendation systems such as:

- Can embedding techniques effectively represent multimodal data such as ingredients and recipe images to capture a meaningful relationship with the recipe data?
- Does providing recipe instructions benefit or confuse the model's ability to capture meaningful relationships?
- Do clustering and similarity technologies enhance the understanding of dependencies in the recipe dataset leading to improved results?
- How well does the recommendation algorithm perform in generating diverse and relevant recipe suggestions based on user input ingredients?
- How well does this recommendation system perform in real-world scenarios, and how adaptable is it to diverse culinary preferences and user input?

2.2 Related Work and Current State-of-the-Art Research

Over the years, several recipe recommendation systems have been proposed based on different constraints. Food recommendation systems can be categorized into four categories: health related systems, ingredient-based systems, time-aware systems, and cluster-based systems. Health based systems incorporate nutritional factors into recommendation models, allowing users to select healthier eating habits. Ingredient-based recommendation systems focus on the content of recipes as well as user ratings and constraints regarding specific ingredients. Time-aware recommendation systems attempt to capture temporal information of user ratings, allowing the system to weigh the importance of older and newer user ratings to create a better preference prediction. Cluster-based models deploy grouping mechanisms for food items, where similar foods are identified and ratings for unknown foods can be predicted. This eliminates the cold-start problem of not having user preference initially. Our system will target ingredient-based content in the current phase with cluster-based methods and will delve deeper into expanding the model for user preference in the next phase.

2.3 Ingredient-Based Systems

Food recommendation algorithms tackle the information overload problem internet users face by using intelligent algorithms to make personalized food recommendations from a wide range of choices. All recommendation algorithms are divided into three search factors: attribute tagging, visual searches, and natural language searches. Attribute tagging involves applying automated food tagging and categorization to data, allowing users to filter search through different food attributes. Visual searches use computer vision to find and suggest foods that are most visually similar to the images they upload. Natural language searches involve customers searching for phrases to describe a particular recipe. When approaching a recommendation system, there are two basic approaches: collaborative filtering and content-based filtering, in which the former relies on historical interactions and the latter on item characteristics. Content-based filtering will rely on informative content descriptors to create a prediction. Collaborative filtering will correlate ratings across populations of users to the current new user.

Now that we have covered how recommendation systems are usually selected and categorized, we will explore proposed ingredient-based systems. Although recommendation systems have many applications within consumer industry, recipe recommendation systems are niche amongst the AI research community. Our model will focus on content-based filtering systems. In 2008, [7] proposed a model using a KNN classifier to make recommendations based on similar recipes. [8] utilized natural language processing techniques to parse recipe instructions and transform them into graphs showing the sequential flow. However, this did not consider ingredients nor user preferences. Authors of [6] propose a collaborative filtering approach called content-driven temporal-regularized matrix factorization (CTRMF) for recipe recommendation systems. This method aims to integrate diverse content information such as ingredients, categories, preparation methods, and nutritional faces into a matrix factorization model. This captures the latent correlations amongst recipe elements and exploits the temporal biases for future recommendations. In [9], the authors propose a recommendation model with a graph convolutional network, which allows multiple embedding layers and propagation mechanisms to capture the complex relationship between ingredients.

2.4 Cross-Domain Influence of Product Recommendation Systems

While recipe recommendation systems may be considered a niche application, other recommendation systems in various domains have garnered significant research due to their unsupervised nature. One such area is that of product recommendation systems. More specifically, we will look at fashion recommendation systems with multi-modal data. Despite their domain differences, both recommendation systems share similarities in their need for semantic understanding, feature extraction, focus on user preferences, similarity measurements, and context awareness.

Our proposed solution will consist of multi-modal embeddings. Embeddings play a crucial role in capturing visual and textual information in a lower-dimensional space. By reducing images to embeddings, similarities between items based on key features are facilitated. In [1], a Convolutional Neural Network (CNN) was used as an embedding function by extracting important features. We see this type of embedding function repeated across studies, such as in [2] and [3]. Siamese CNNs considering pairwise compatibilities and multi-modal embeddings combining text and image data are also common approaches to fashion embeddings [2].

Outfits, like ingredients in a recipe, are sets of arbitrary lengths which match contextually and are compatible with each other. Measuring the dot product between two items should reflect their compatibility [2]. Nearest neighbor approaches to outfit recommendation systems have been taken [1], but prove to have little semantic understanding. Visual Compatibility Learning such as CNNs and Siamese networks have captured the compatibility across fashion items in [4], but face the vanishing gradients issue. In order to overcome this, LSTMs have been proposed.

LSTMs regulate the use of memory while capturing sequential dependencies. Thus, by treating the set as a sequence, bidirectional LSTMs have been proposed with visual-semantic embedding [4]. This approach was built upon by [3], where a CNN serves as an embedding function, a bidirectional LSTM would learn sequential outfit combinations, a Visual-Semantic Embedding model would manage multimodal data, and a Style Embedded Autoencoder would learn to predict style vectors for each outfit. More recently [5], graph neural networks have been used with significant improvement (96 percent accuracy). Such research in different domains can assist us with finding a new and improved solution for recipe recommendations. Our proposed method will explore multimodal embeddings with cosine similarity matrices. Because we are using image data, K-means clustering may be necessary to preserve image features. K-means clustering was found to take patterns and designs on clothing imaging in a submission for Kaggle's H&M Personalized Fashion Recommendation Challenge. Within this approach, VGG-16 was used to extract image features.

3. Introduction to Multimodal Embedding, Similarity Analysis, & Clustering Methods

In the following sections, we will explore multimodal embedding techniques, similarity analysis methods such as cosine similarity, and clustering methods such as K-means Clustering and Spectral Clustering.

3.1 Image Embeddings

Image embedding is a technique used to represent images as feature representations in lower-dimensional space. Images tend to be high-dimensional vectors that often carry a lot of noise. By embedding images, the essence of an image can be captured by preserving meaningful features and important information. Image embeddings are often used in pre-processing steps for models to reduce computational expenses and making image retrieval, object recognition, and clustering more efficient. Several embedding techniques such as PCA, LDA, autoencoders, attention-based methods, and Siamese networks can be used. More recently, there has been a focus on utilizing deep learning to generate image embeddings. One such architecture, VGG-16, has been an effective model for feature extraction. Due to its straightforward and versatile architecture, it has become a popular technique. VGG-16 is a convolutional network compromising of 16 layers, 13 of which are convolutional layers. Because of its usage of small 3x3 convolutional filters, it delivers a more uniform representation of features across images. The model is considerably deep, enabling it to extract complex features. Using pre-trained weights from large datasets like ImageNet, it can be used for image embedding through transfer learning. Images are passed through the network, and the output from the convolutional layers is taken as an image embedding. This embedding can then be utilized for several machine learning tasks. Although the model offers a lot of benefits, it has its limitations. The most severe limitation is its high computational cost; it takes a lot of time and computational units to run. Limited memory and storage resources greatly affect the ability to apply this model efficiently. As with other neural networks, VGG-16 is prone to vanishing and exploding gradient problems. This can make it challenging to update weights effectively. While VGG-16 is often used as a preliminary approach to test machine learning applications, other more advanced architectures are being used such as ResNet, Inception, Vision Transformers, and DenseNet. Such models may perform better on embedding tasks.

3.2 Word Embeddings

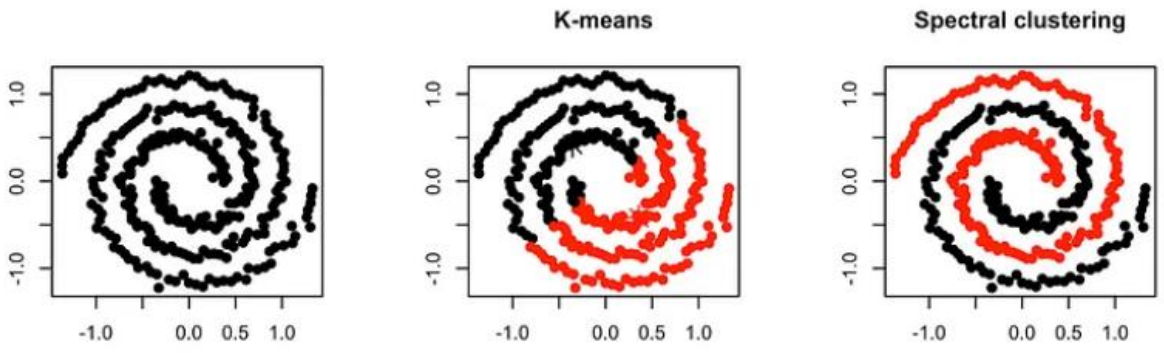
Word embeddings refer to vectorized representations of words, capturing semantic and syntactic relationships between words. Embeddings are different than one-hot encoded vectors, as they reduce the problem of sparsity and lack of relationships between words. Word embeddings map words to vectors where words with similar meanings are placed near each other. Although there are several word embedding techniques, one of the most popular methods is Word2Vect. The architecture is made of two models: Continuous Bag of Words and Skip-gram. One model predicts a target word based on the surrounding context while the other predicts context based on a target word. By using neural networks, embeddings are updated to maximize the probability of accurate prediction. Other word embedding methods are GloVe, ELMo, fastText, & BERT. Word2Vec remains one of the most popular methods due to its computational efficiency and capturing of semantic relationships. This is counterbalanced by its limitations in capturing complex embeddings, as the networks used in the models are shallow.

3.3 Similarity Analysis in Recommendation Systems

Similarity analysis methods refer to techniques which measure the similarity between two vectors. More specifically, cosine similarity is a method of calculating the cosine of the angle between vectors, representing similarity values between -1 and 1. 0 suggests vectors are orthogonal; -1 represents that vectors are the opposite, and 1 represents that the vectors are identical. Cosine similarity is very efficient and robust to vector magnitudes. This is often used in both collaborative and content based filtering, as it can be used to find users with similar tastes or recipes that are similar to each other.

3.4 Clustering Techniques

There are two approaches to clustering. The first approach uses compactness to cluster points while the second uses connectivity. Compact algorithms find points that lie close to each other and form centers. Closeness is measured by distance. Connectivity algorithms focus on points that are connected or next to each other. Even if the points lie near each other distance-wise, they may not be clustered together if they are not connected. K-means clustering is a compactness algorithm while Spectral Clustering is a connectivity algorithm.



K-means clustering is an unsupervised technique used to sort data points into distinct clusters based on their similarity. When applied to embeddings, K-means clustering can be useful for organization of the complex data representation. K-means clustering also offers several advantages with multi-modal data, as it facilitates the discovery of latent patterns. Combining both image and text data enables the identification of culinary themes that may reach beyond semantic understanding. K-means clustering is also a fairly interpretable algorithm that can provide similar and relevant data points, enhancing the number of similar recommendations provided to a user. Some limitations of K-means clustering depend on the nature of the clusters, such as the shape and number. As the data becomes more complex, non-convex clusters become harder for K-means to identify. This along with the sensitivity to outliers leads us to consider alternative methods.

Spectral clustering is used when dealing with complex data distributions and potentially non-convex clusters. Spectral clustering transforms the data into lower-dimensional space, where clusters become more separable. Because recommendation systems may have graph-like patterns within the data, spectral clustering can capture relationships between recipes in a more complex manner. There are two approaches to spectral clustering, one which uses similarity matrices and the other which uses K-Nearest Neighbors. With the similarity matrix approach, a similarity matrix is constructed to capture similarities between data points. The eigenvalues and eigenvectors of the Laplacian of the similarity matrix are dimensionally reduced and clustered. With the KNN approach, a KNN graph is constructed. The Laplacian is then used to obtain spectral embeddings, which are then clustered as before. It should be noted that spectral clustering is very computationally expensive due to the calculation of eigenvalues. In order to explore the convexity of the data, we shall attempt both clustering techniques.

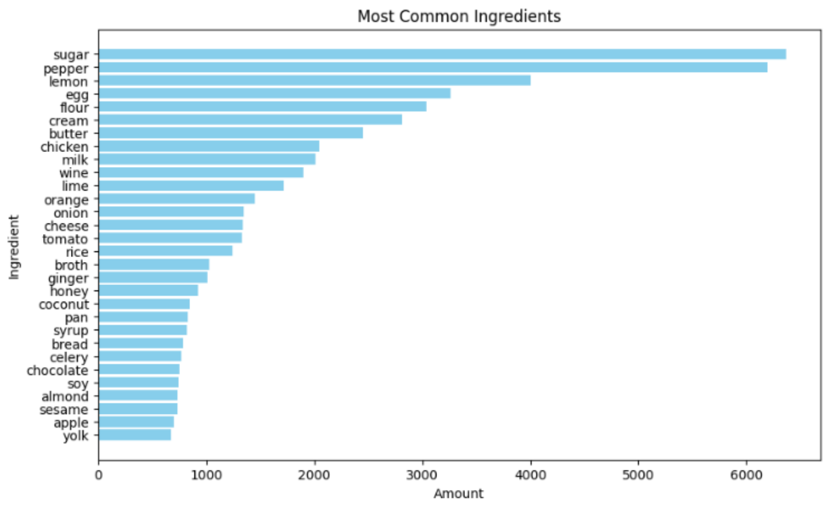
4. Data Collection, Analysis, & Initial Pre-Processing

The dataset used for this project can be found on Kaggle under the name “Food Ingredients and Recipes Dataset with Images”. It consists of a CSV file mapping the title of the food dish, ingredients, recipe instructions, the corresponding image name, and processed and cleaned ingredients as well as 13,582 images of each recipe. This dataset was chosen for its usability rating and size of 216 MB.

Within the dataset, 13,305 out of 13,496 titles were found to be unique. About 30 ingredients and instructions were also found to be repeated. Non unique image names, instructions, and cleaned ingredients were removed. Thirteen total null values were found in the dataset. Due to the relatively low numbers, these rows were removed.

Looking at the data table initially, pre-processing for the ingredient list was necessary. Each ingredient list contained a measurement unit. However, measurement across ingredients is not able to be converted, as some are liquids, some are in single units, and some are solids. There are two possible approaches to process this; either one can delete the metrics, predict the ingredient list necessary for a dish, and extrapolate the numbers in post-processing steps, or we can separate the ingredient category into ingredients by units. Due to time restrictions and simplicity of the model, the first option was chosen. While processing the images, it was found that a few rows did not contain matching images. These rows were left without an embedding and deleted.

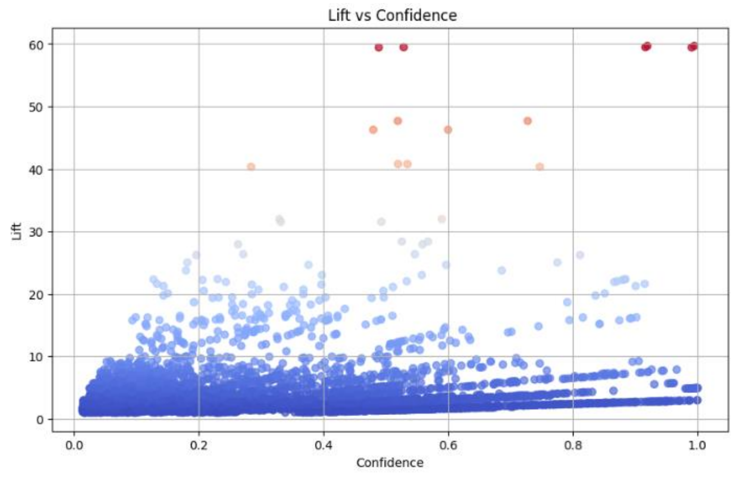
sugar: 6366 times	pepper: 6195 times
lemon: 4006 times	egg: 3266 times
flour: 3042 times	cream: 2817 times
butter: 2454 times	chicken: 2049 times
milk: 2008 times	wine: 1899 times
lime: 1722 times	orange: 1448 times
onion: 1348 times	cheese: 1342 times
tomato: 1327 times	rice: 1242 times
broth: 1031 times	ginger: 1013 times
honey: 925 times	coconut: 846 times
pan: 827 times	syrup: 818 times
bread: 784 times	celery: 770 times
chocolate: 754 times	soy: 743 times
almond: 739 times	sesame: 739 times
apple: 702 times	yolk: 678 times



Most Common Ingredients Further analysis of the dataset led us to find that there were 4,877 unique ingredients. The list of the most common ingredients can be found in the table. About 1,1882 ingredients were unique to only one recipe. The average common ingredient count is 1796.6, but only 10 ingredients are more commonly found than the average. This means that the majority of the dataset has unique ingredients rather than common ingredients, which can affect the complexity of the relationship between recipes. With less common ingredients found in most recipes, less complex relationships might be found. This may prove easier to cluster, but may affect the model's generalizability.

We also analyzed possible associations between ingredients that frequently appear together with an Apriori algorithm. The top association rules can be found in the below tables. The chart was primarily sorted by lift, which measures how likely a consequent ingredient is found when an antecedent ingredient is present. The lift is the measure of the support of the antecedent and the consequent divided by the support of the antecedent multiplied by the support of the consequent.

Antecedents	Consequents	Lift
(parmigiano)	(reggiano)	59.68
(reggiano)	(parmigiano)	59.68
(parmigiano, pepper)	(reggiano)	59.464
(reggiano)	(parmigiano, pepper)	59.464
(parmigiano)	(reggiano, pepper)	59.428
(reggiano, pepper)	(parmigiano)	59.428
(parchment)	(paper)	47.677
(paper)	(parchment)	47.677
(squash)	(butternut)	46.262
(butternut)	(squash)	46.262



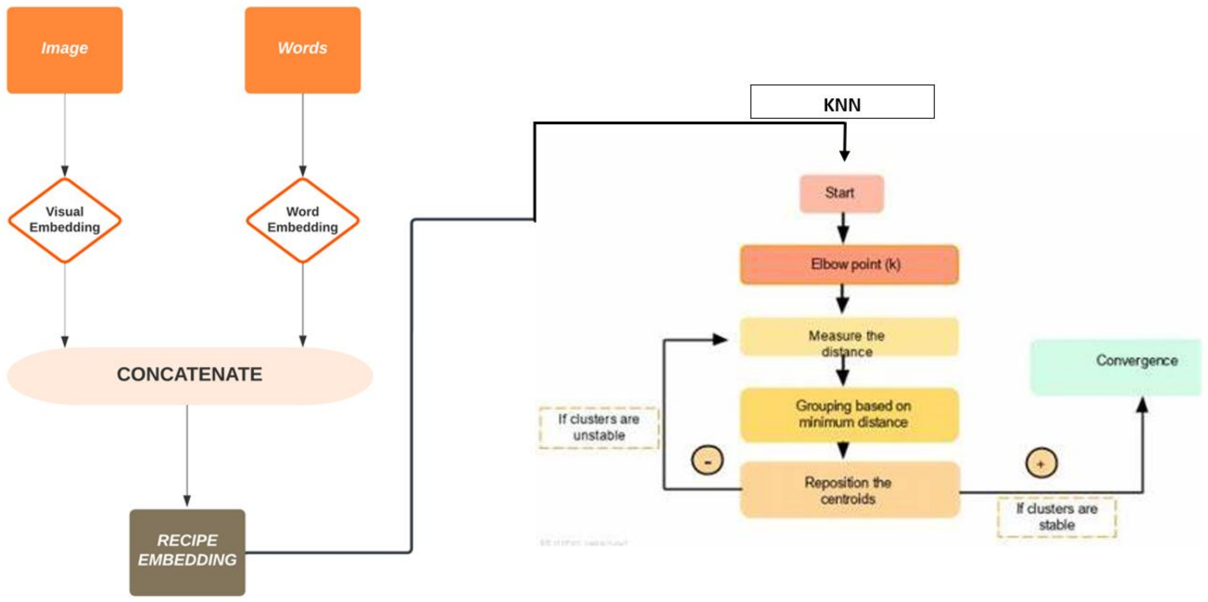
We can visualize this further across the dataset with a scatter plot representing association rules, their lift, and confidence values. We can note the majority of the points are clustered near the bottom of the graph, suggesting that the presence of one ingredient does not strongly impact a consequent ingredient. This may indicate that ingredients are commonly found independently of each other, with the exception of a few outliers marked in warmer tones. We also note that the confidence is spread across the graph, indicating that the strength of the association rules are varied. Analysis of lift and confidence underscores the possibility of diversity of ingredients within our dataset.

5. Methodology

The proposed pipeline for this project is delineated in the following steps:

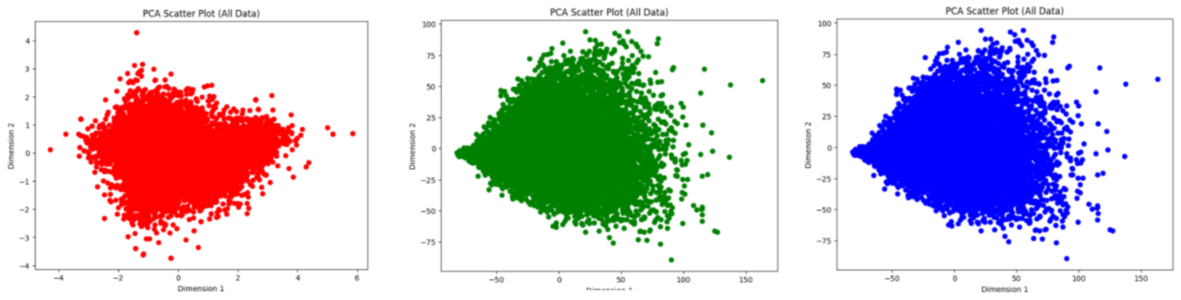
1. Data Pre-Processing
2. Image Embedding with a pre-trained VGG-16 on ImageNet
3. Ingredient Embedding with Word2Vec
4. Concatenation of Embeddings
5. Clustering
6. User Input Processing
7. Centroid Similarity with User Input
8. Recipe Filtering
9. Display of Results

Proposed Architecture



The proposed architecture involves embedding steps, clustering steps, and user similarity filtration. Embedding images involved minimal pre-processing steps, as the VGG-16 model extracts features. Word Embeddings required initial parsing, punctuation removal, tokenization, filtering of non-alphabetical and non-informative words, lowercasing, and lemmatization. Words were then passed into Word2Vec, which produced a vector size of 100 features per recipe row. In order to concatenate the word and image embeddings, the image embeddings were passed into a dense layer that converted the embedding from a (3,3,512) vector per row to a vector size of 100 per recipe row. Both vectors were stacked to create a concatenated embedding of shape (1,200) per recipe row.

5.1 Visualizations of Arrays

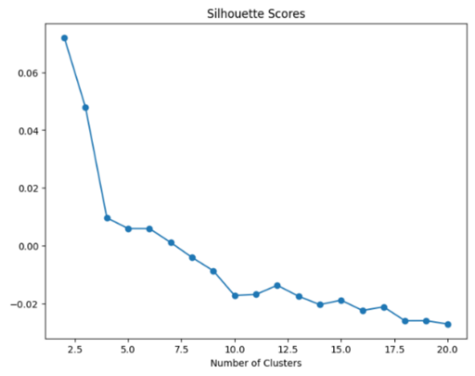


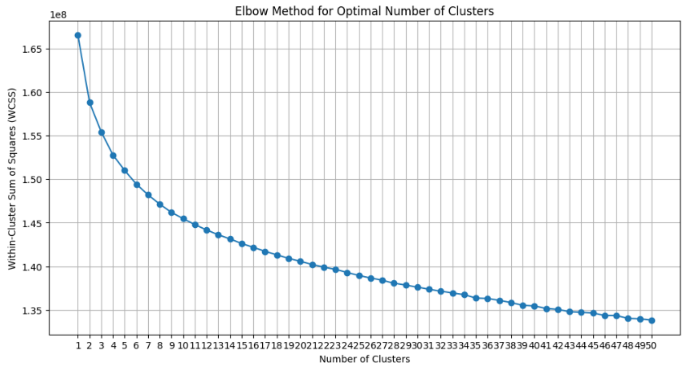
In the graph we can see the visual representations of the word embeddings (red), the image embeddings (green), and the combined embedding (blue) after dimensional reduction (n components=2). Based on the graph, it is possible that image embeddings dominate the data points, calling for possible standardization of the embedded vectors.

5.2 Algorithmic Experiments: Convexity

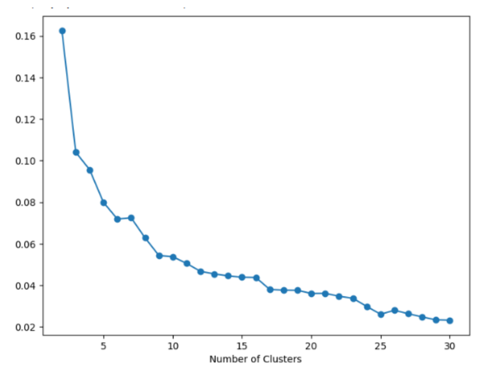
In the following section, we will explore the dependencies in the recipe data set by experimenting with a convex and non-convex clustering algorithm. Our experiments will shed light into the complexity of the relationship between ingredients. We will also measure the effects of standardization on the embeddings. Silhouette Scores and Within Cluster Sum of Squares (WCSS) will be used for comparison.

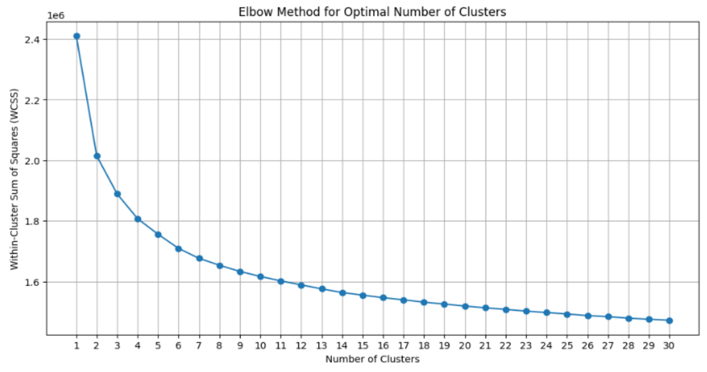
Non-Standardized K-Means Clustering



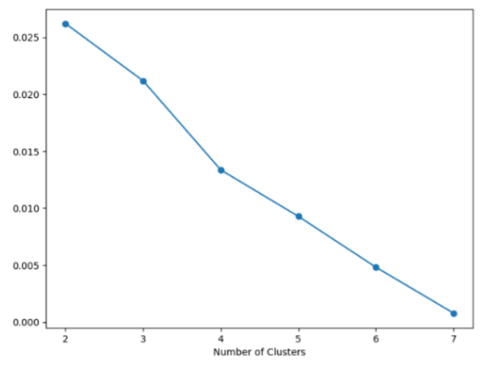


Standardized K-Means Clustering

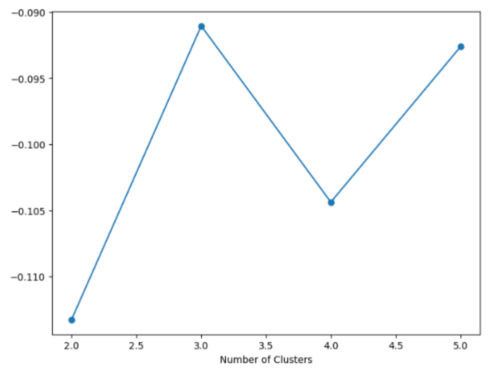




Similarity Matrix Spectral Clustering



KNN Spectral Clustering

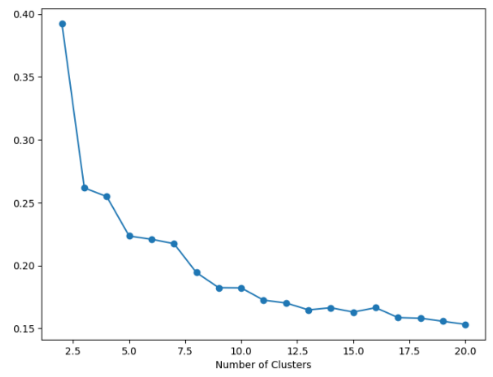


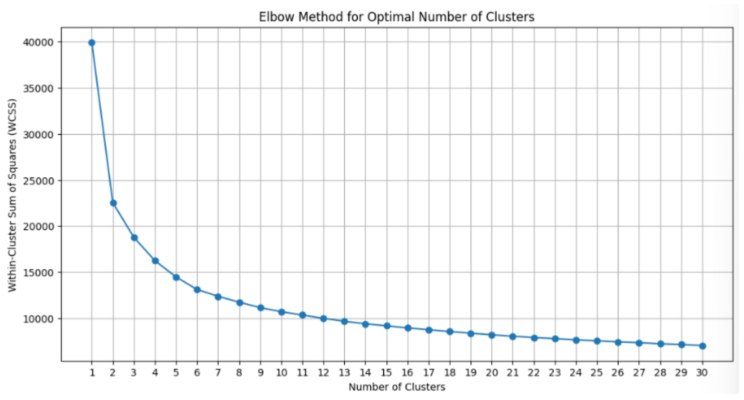
The graphs above show Silhouette and WCSS scores for K-Means and Spectral Clustering Algorithms. It can be noted that the ideal number of clusters for each algorithm is as follows: k=12 for Unsupervised K-means, k=10 for Supervised K-means, k= 3 for Similarity Matrix Spectral Clustering, & 2 for KNN Spectral Clustering. Silhouette scores for Spectral Clustering and Unstandardized K-means were rather low, with most staying near 0. Standardized K-means had higher silhouette scores, with the maximum staying at 0.38. It should be noted that although silhouette scores across all models were not high, it may not mean the model is ineffective, as silhouette scores describe how far apart clusters are from each other. Standardization was attempted with a KNN spectral clustering of 3 but was unable to be reproduced on more clusterings due to computational expenses and RAM issues. A significant increase in the silhouette score was seen, as it was 0.12 at 3 clusters. Standardized K-means clustering performed best on the data and thus will be used for the final model.

5.3 Algorithmic Experiments: Uni-Modal vs Multi-Modal Data

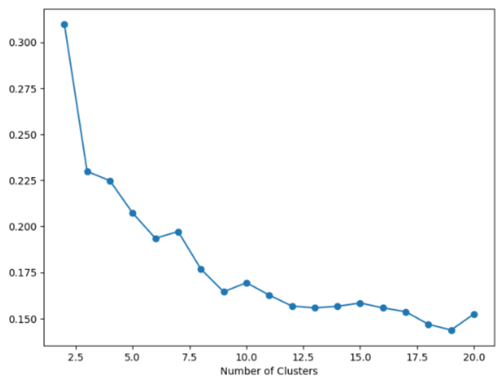
In the following section, we will explore whether features from image data will influence the model's ability to cluster. An experiment was run on clustering with only word embedding data and the other with combined data. The results for standardized and unstandardized embeddings can be found below:

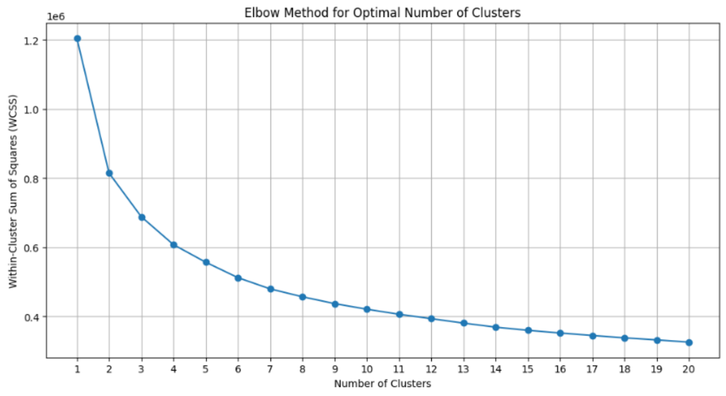
Unstandardized





Standardized



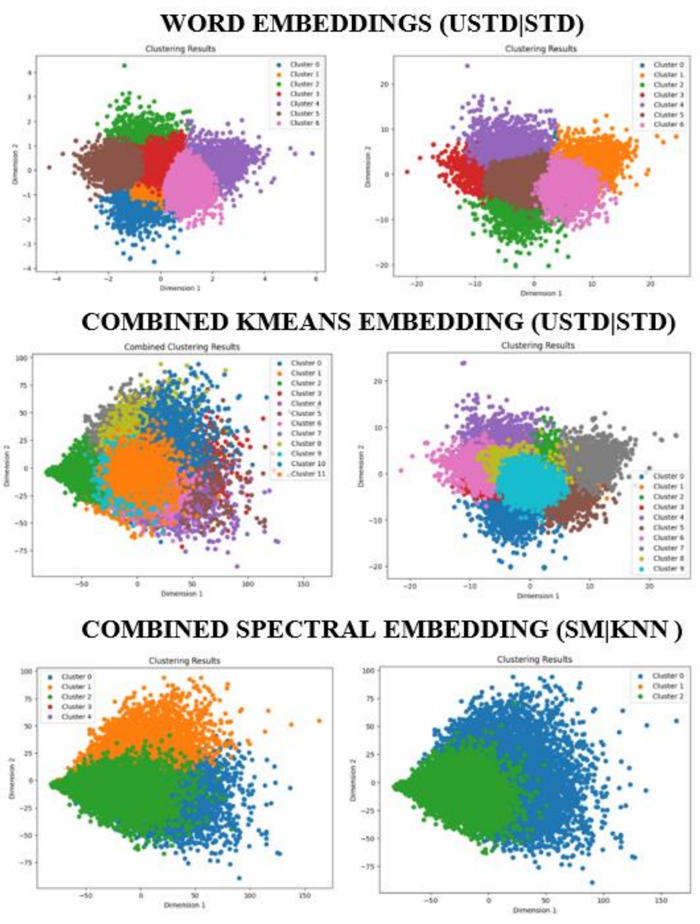


We can note that the effect of standardization is less pronounced with uni-modal data, and both unstandardized and standardized models have a lower within cluster sum of squares than the multi-modal data. The silhouette scores are also considerably higher in uni-modal data than in multi-modal data. This may mean that the feature extraction method chosen for images may not be sufficiently capturing significant information, or that image data does not enhance the model.

5.4 User Output Methods and Filtering

After choosing a clustering type, we can move on to applying the pipeline to user input. User input will need to be preprocessed and embedded in the same way that the training data was processed. In order to calculate the centroid most similar to user input, cosine similarity is calculated between the k cluster centroids and the input. Using the top three most relevant clusters, an output temporary data frame is created. This data frame can be filtered in any way the user desires. For example, if the user has an allergy, the data frame will exclude recipes with the allergen. For this particular phase of the project, only one constraint was tested. The chosen constraint allows the output to only be recipes with the inputted ingredients. The output of the model will have a random recommended recipe from the filtered data frame, the ingredients, the instructions, and the photo.

6. Results



6.1 2-Dimensional Clustering Visualizations

The following results are visualizations of the attempted models. We note that these visualizations are dimensionally reduced and may not accurately represent the data. From the image, it becomes clear that uni-modal embeddings have sharp and clear cluster boundaries while multi-modal embeddings do not.

6.2 User Output & Analysis

Although visual analysis, silhouette score, and WCSS were used to determine the quality of the clustering techniques, it is difficult to evaluate the successfulness of the model due to its unsupervised nature. Many methods of evaluation such as precision, recall, F1 score, and AUC all depend on user ratings and studies. We can only analyze to the extent of the quality of the clusters, initial input-output experiments, and user studies to understand the quality and relevance of the recommendations. Below are some sample outputs of the algorithm. At first glance, the algorithm performs almost ideally.

```
User Input: pineapple, onion
Recommended Recipe Title: Pineapple-Glazed Chicken with Jalapeño Salsa
Ingredients: pineapple,sugar,pineapple,pepper,onion,jalapeno,chicken breast
Instructions: Preheat oven to 400°F. Bring pineapple juice, brown sugar, and mustard to boil in small saucepan, stirring to dissolve sugar. Boil until glaze has thickened slightly, about 1 minute. Season with salt and pepper.
Mix pineapple, red pepper, cilantro, onion, and chiles in medium bowl. Season with salt and pepper.
Line baking sheet with foil. Place chicken on sheet and brush with glaze. Bake 15 minutes. Brush again with glaze, then broil until cooked through and golden, watching closely to avoid burning, about 5 minutes longer. Let rest 5 minutes.
Spoon salsa over chicken and serve.
```

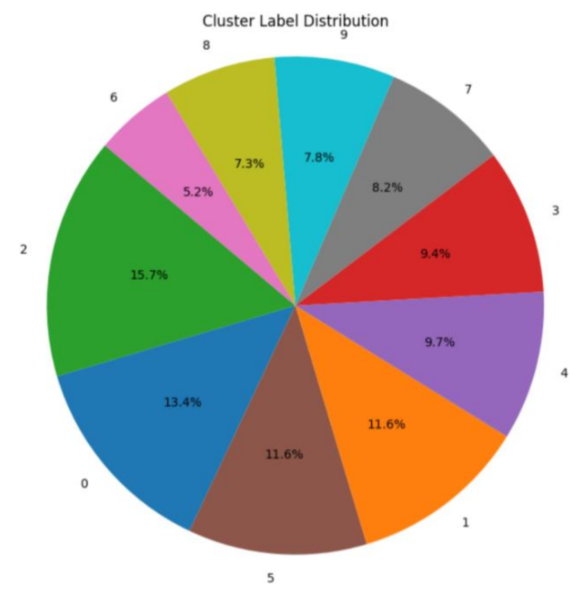
```
User Input: egg, steak
Recommended Recipe Title: Diner-Style Western Omelet
Ingredients: ham steak,onion,pepper,egg,milk,pepper,monterey jack cheese
Instructions: Heat oil in a medium (preferably 10") nonstick skillet over medium. Cook ham, onion, and bell pepper, stirring often, until softened and beginning to brown, 5–7 minutes.
Meanwhile, whisk eggs, milk, salt, and pepper in a medium bowl.
Reduce heat to medium-low and shake pan so that ham and vegetables form a single layer. Pour egg mixture over ham and vegetables, then sprinkle cheese over. Cook, tilting skillet and gently running a rubber spatula around edges to allow uncooked egg to flow underneath, until eggs
```

```
User Input: potato, steak
Recommended Recipe Title: 3-Ingredient Steak With Crispy Parmesan Potatoes
Ingredients: potato,parmesan,hanger steak,pepper
Instructions: Place potatoes in a medium pot; add cold water to cover by 1". Season with salt, bring to a boil, and cook until potatoes are fork-tender, 12-15 minutes. Transfer to a rimmed baking sheet; let cool slightly, then lightly crush with your palm.
Heat 3 Tbsp. oil in a large heavy skillet over medium-high. Reduce heat to medium-low and add half of the potatoes; season with 3/4 tsp. salt. Cook, turning once, until golden brown, 15-20 minutes. Transfer potatoes to a plate. Add 3 Tbsp. oil, 3/4 tsp. salt, and remaining potatoes t
Meanwhile, heat remaining 2 Tbsp. oil in a large skillet over medium-high. Season steak with salt and pepper and cook, turning occasionally, until deep brown and an instant-read thermometer registers 125°F for medium-rare, about 3 minutes per side.
Transfer steak to a cutting board. Thinly slice steak against the grain and serve with potatoes alongside.
```

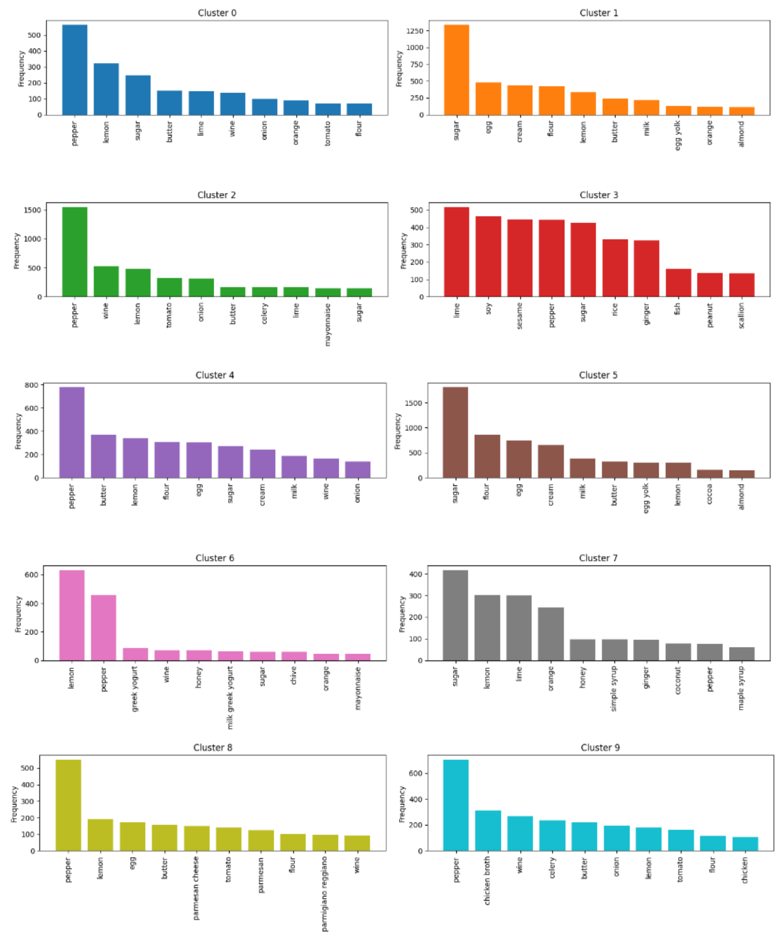
```
User Input: corn, potato
Recommended Recipe Title: Sweet Potato Fritters
Ingredients: potato,cornmeal,onion,pepper,deep
Instructions: Heat the oven to 200°F. Line an oven-proof plate or baking sheet with paper towels. Peel and grate the sweet potatoes and squeeze or press them dry if necessary; you want 3 packed cups; save the rest for later.
Combine the grated sweet potato, flour, cornmeal, onion, egg, and sprinkle of salt and pepper and mix well with a fork. If the mixture looks too liquid, add more flour, 1 tablespoon at a time. (You can make the batter ahead of time and refrigerate for up to a couple of hours before
Put 2 inches of oil in a large pot over medium heat. When the oil is hot, carefully drop spoonfuls of sweet potato into the pot. (Work in batches to avoid crowding the pot.)
Cook turning them with tongs or a slotted spoon as necessary so they brown on all sides, until they're cooked through, 5 to 7 minutes. Transfer the finished fritters to the paper towel-lined plate and put it in the oven to keep warm while you make the rest. Serve hot or at room temp
Drying Grated Vegetables: Put them into a strainer over a bowl or just squeeze them between your hands.
Heating Oil for Deep Frying: To use a thermometer for deep frying, clip it on the side of the pot and make sure the tiny hole that registers the temperature isn't touching the pot. If you don't have a thermometer, use a pinch of cornmeal or flour to test when the oil is ready for fr
Putting the Fritters in the Oil: To minimize splattering, hold one spoon close to the oil and scrape the batter into the oil with another spoon.
Recognizing Doneness: The fritters should be golden on the outside and soft but not wet in the middle. Check inside—you should still be able to tell the sweet potatoes were grated.
```

Let us now analyze the value counts of our standardized K-means clusters. For each cluster, we count the number of recipes that are found. The largest cluster, cluster 2, has 1893 data points while cluster 6, the smallest cluster, has 629. This range is relatively small, and the distribution of cluster sizes appears to be relatively even. The table below and graph show the distribution:

Cluster Number	Count
2	1893
0	1616
5	1404
1	1393
4	1170
3	1137
7	989
9	944
8	883
6	629



In order to improve interpretability of the clusters, we analyzed the top ten ingredients in each cluster. This can be found in the charts below.



6.3 Usability Experiments: Input Format

As seen above, initial input examples seemed to perform quite well but need further test cases to prove robustness and easy usability for customers.

6.3.1. Out of Domain Test

This test assesses how the model performs with ingredients that are not part of the data set. Due to the current constraints requiring output to contain all ingredients, the model should return nothing. When tested, the model returned a fail error. This output will need to be corrected in future versions indicating to users that there are no recipes with the inputted ingredients.

6.3.2. Unstructured Ingredient Parsing Test

This test assesses model performance when input ingredients are not separated by commas. This test produces errors as the model reads the entire input as one ingredient. Further parsing techniques or automated suggested corrections will be necessary to fix this error.

6.3.3. Single Ingredient Test

This test assesses model performance when user input includes only one ingredient. After several trial tests, the model was able to perform well with only one ingredient.

6.3.4. Tokenization Errors Test

This test assesses weather tokenization errors affect model performance. It was found that depending on the type of tokenization the model may or may not have issues. For example, when steak is input as an example ingredient, the model will qualify a recipe with the ingredient “ham steak”, an obvious tokenization error, as a potential candidate. Other tokenization errors such as missing pluralized endings can cause the model to incorrectly filter out recipes. This suggests better tokenization methods and processing are necessary for a more precise model.

6.3.5. Chronological Order Test

It is important to assess whether the order of user input matters to the result of the model. After performing this test, it was found that the output will remain the same regardless of the order. For example, if the user inputs “potato, steak”, their outputs will be the same as if they put in “steak, potato”.

6.3.6. Special Character Test

This test assesses whether special characters affect the output of the model. Due to the lack of further preprocessing of user input, extra characters affect the ingredient list. In order to fix this problem, all non-alphabetical characters must be deleted in the list.

6.3.7. Spelling Errors Test

This test assesses whether spelling errors affect the output of the model. Without automated spelling checks, the model is unable to understand similar ingredients that may be misspelled.

7. Conclusion

Our pipeline and experiments set out to answer whether embedding and clustering techniques effectively represent ingredient data well for recommendation purposes. In this study, we have analyzed whether multi-modal data captures meaningful features. Our experimentation leads us to conclude that further work will need to be done on the embedding methods, as uni-modal word data clustered more definitively than the embeddings with added image data. We also can conclude that the usage of image data makes standardization a requirement for the model to function accurately, as image data and word embeddings are scaled differently. Different models may be attempted and tested in future work to further determine the consequences of multi-modal data.

Through our analysis of clusters and similarity methods, we can conclude that clustering generally captures unseen relationships between ingredients. When looking at the top ten ingredients of each cluster, we can see certain cuisine types forming. For example, Cluster 3 contains the top ingredients of lime, soy, sesame, pepper, sugar, rice, ginger, fish, peanuts, and scallions. These ingredients are typically associated with Asian-inspired cuisines. Cluster 1 contains sugar, egg, cream, flour, lemon, butter, milk, egg yolk, orange, almond, otherwise known as typical pastry and dessert ingredients. We can also note that Cluster 6, with ingredients containing lemon, pepper, Greek yogurt, wine, honey, milk, sugar, chive, orange, mayonnaise, may represent Mediterranean cuisines. Similarly, Cluster 8 contains Italian cuisine ingredients, Cluster 9 contains French cuisine ingredients, Cluster 7 contains very sweet ingredients used for drinks, and Cluster 2 may be Western or American cuisines. Although we can see a divide in cuisine types, there are a few clusters that may be unnecessary or training data was unable to define them further. Cluster 0 is an atypical cluster with very broad ingredients. This cluster was rather large in value count, thus meaning that further training data is missing. Certain clusters, such as clusters 4 and 5, contain repeated ingredients from French and dessert cuisine types. This may signify that too many clusters were selected. Because the K clusters were selected with the elbow method, there may be some room for interpretation error. Considering the unimodal data embeddings selected 7 clusters as optimal and the multi-modal selected 10 clusters as optimal, it may mean that these 3 extra clusters that we are seeing may be unnecessary as a whole without further training data. Although we can spot cuisine patterns appearing from our clusters, we do not have specific cuisine classification in our training set. Using a dataset in the future containing cuisine types could help with accuracy metrics. From the given clusters, it may be that our training data was biased towards specific cuisine types, as there are several that are missing. Further studies with better data could improve the quality of the model and the diversity of output. In future studies, it would be interesting to include recipe instructions in the embeddings. This may also help creating a model that estimates cook times, but may also add beneficial information with the differing styles of cooking ingredients together. In conclusion, while this pipeline shows promise in its preliminary stages, further refinement and examination is necessary to enhance its robustness.

8. References

[1] M Sridevi et al. Personalized fashion recommender system with image-based neural etworks. https://iopscience.iop.org/article/10.1088/1757-899X/981/2/022073/pdf, 2020.

[2] Elaine M. Bettaney, Stephen R. Hardwick, Odysseas Zisimopoulos, and Benjamin Paul Chamberlain. Fashion outfit generation for e-commerce. arXiv preprint arXiv:1904.00741, 2019.

[3] Takuma Nakamura and Ryosuke Goto. Out-fit generation and style extraction via bidirec- tional lstm and autoencoder. arXiv preprint arXiv:1807.03133, 2018.

[4] Xintong Han, Zuxuan Wu, Yu-Gang Jiang, and Larry S. Davis. Learning fashion ompatibility with bidirectional lstms, 2017.

[5] Guillem Cucurull, Perouz Taslakian, and David Vazquez. Context-aware visual compatibility prediction. arXiv preprint arXiv:1902.03646v2, 2019.

[6] Lin, C.-J., Kuo, T.-T., & Lin, S.-D. A Content-Based Matrix Factorization Model for Recipe Recommendation. In Advances in Knowledge Discovery and Data Mining. 10.1007/978-3-319-06605-9_46

[7] Zhang Q., Hu, R., Namee, B., Delany, S.: Back to the future: Knowledge light case base cookery. Technical report, Technical report, Dublin Institute of Technology (2008)

[8] Wang, L., Li, Q., Li, N., Dong, G., Yang, Y.: Substructure similarity measurement in Chinese recipes. In: Proceeding of the 17th International Conference on World Wide Web

[9] Xiaoyan Gao, Fuli Feng, Heyan Huang, Xian-Ling Mao, Tian Lan, Zewen Chi. Food recommendation with graph convolutional network. Information Sciences, Volume 584, January 2022, Pages 170-183. DOI: https://doi.org/10.1016/j.ins.2021.10.040

Dataset: https://www.kaggle.com/datasets/pes12017000148/food-ingredients-and-recipe-dataset-with-image