Predicting Cuisines of Recipes

Abstract

The application of machine learning and data mining provides effective predictive models for cuisine classification. However, a more difficult task is to complete a partial recipe by adding missing ingredients. We approach this problem using recommendation systems based on collaborative filtering. In this project, we propose a cuisine classifier as well as a recommending system to complete partial recipes. We evaluate all models and selected Random Forest as the best classification algorithm, and KNN with z-score as the best collaborative filtering algorithm for the recommendation system. The rest of the algorithms are discussed in more detail and future work is proposed.

1 Introduction

Recommendation systems are highly desirable for many applications within the scientific community. The increased interest in improving recommendation systems comes from suggesting the use of collaborative filtering to improve recommender objects or items under a given task. In this study, we apply data mining and machine learning techniques to explore recommending ingredients and predicting cuisines. This interest in suggesting cuisines or completing incomplete recipes has greatly increased especially with the ease of searching for endless amounts of recipes online. This popularity motivates researchers to identify a cuisine given a recipe as well as completing an incomplete recipe. Therefore, this paper will analyze and implement a classifier and a recommendation system in this context.

There have been quite a few implementations of classifying cuisines based off a given recipe [5]. Previous techniques include Naive Bayes, Random Forest, and Linear Support Vector Machines. Each have their advantages and disadvantages. Naive Bayes provides a probabilistic model and is considered a simple approach to classifying cuisines. Random Forest classifiers provide many decision trees becoming one of the popular classifiers to use especially for data mining. Lastly, linear SVMs provide high efficiency in high dimensions and memory. We examine the above classifiers to observe which provides the best performance for classifying cuisines.

Various forms of recommendation systems are now a necessary aspect in people's everyday lives. This dependency in recommendation systems gives the ability to assist in minor or even major decisions based off predictions or certain evaluations. Collaborative filtering is becoming the replacement for these normal recommendation systems. This method uses known information about a group in order to make recommendations of unknown options [12]. This is done by comparing previous preferences in order to make a decision about a new preference. In determining the best model for the recommendation task, we experiment with model-based and memory-based collaborative filtering techniques due to each having their own advantages and disadvantages.

This paper will focus on predicting the type of a cuisine from a given list of ingredients for recipes. We will further explore techniques that can be implemented to make decisions in order to complete

partial recipes. Collaborative filtering will come to play in this aspect of the assignment. With these techniques in mind, this paper will discuss the metrics and performance in order to evaluate each method effectively. We structure the paper as the following: Section 2 provides details about the dataset and task. Section 3 discusses the exploratory data analysis techniques used to better visualize and interpret the given data. Section 4 will give information on the methodology including the classification technique chosen as well as the technique used for recommending partial recipes. We will provide and interpret our results in section 5. Lastly, section 6 will conclude our research and provide future work in this area.

2 Data preparation

This section provides information on our approach to familiarize and prepare the data for classification as well as recommending for partial recipes. The data contains ingredients for particular recipes in order to make cuisine classifications. There are a total of 4,235 recipes with 709 ingredients from 12 cuisines for the classification aspect of this research paper. Since we are given a very small dataset, we do not remove any ingredients and consider each ingredient to be an important item to consider in classification.

We take additional steps to prepare the data for collaborative filtering in order to recommend ingredients to partial recipes. We utilize Surprise [4] for building the recommendation system which requires the data to be formatted a certain way. Therefore, we convert our data by using the recipe ID and ingredient ID to get a specific score. Originally, our data shows each row representing a recipe and each column representing an ingredient. We rearrange this format so our new data will be formatted to be 3 columns: the first column contains the ID for the recipe; the second column contains the name of the ingredient; the third is the rating. We consider the rating to be 0 if ingredient is not in the recipe and 1 if the ingredient is present.

Our team considered combining similar ingredients that have different variations, such as various types of rice or beans. The initial reasoning behind this was to remove any form of variation that was not relevant. However, since we do not remove less frequently used ingredients within recipes since the dataset is small and concise, we use the same reasoning for simplification.

We evaluate the classifier using the mean classification accuracy score over the k-splits in K-Fold. As for evaluating the collaborative filtering algorithms, we use the common root mean squared error (RMSE) in the Surprise application.

3 Exploratory Analysis

In order to gain a better understanding of the structure of our data, we apply exploratory data analysis methods. This gives us a better insight on the relationships between the different ingredients as well as between the ingredients and the cuisines.

Cuisine	Chinese	English	French	German	Greek	Indian
Shortest recipe	**4**	**4**	5	6	7	10
Longest recipe	22	21	28	26	27	28
Mean	10.3	11.4	12.5	12.5	14.8	16.0
Cuisine	Italian	Japanese	Mexican	Moroccan	Spanish	Thai
Shortest recipe	9	10	12	12	13	14
Longest recipe	30	29	33	34	32	**42**
Mean	16.4	17.8	18.2	21.3	22.2	**22.7**

Table 1: Length of the longest and shortest recipes for each cuisine. The overall longest and shortest recipes are highlighted, as well as the largest mean recipe length.

We begin by analyzing the length of the recipes in each cuisine and the most common ingredients per cuisine and overall. First, we determine the number of ingredients for the shortest and longest recipes for each cuisine. Then, to better understand how the length of the recipes varied in each cuisine, we determine the mean recipe length across cuisines (Appendix C). The average recipe length , as well as the length of the shortest and longest recipes for each cuisine is presented in Table 1.

We also look into the usage of the ingredients. This allows us to determine what the most commonly used ingredients are (Appendix A), as well as the most characteristic ingredients for each cuisine (Appendix B). Overall, the most used ingredients are *garlic*, *onion*, *olive oil*, *salt* and *chicken*. Determining the most frequently used ingredients for each cuisine is, however, not very relevant, because ingredients like *garlic* are frequently used across all cuisines. Instead, we obtain the most characteristic ingredients, which are the ingredients that are particularly common in that cuisine considering their overall usage. For this, we obtain the average frequency of each ingredient across all the cuisines and subtract it to the raw frequencies to get an adjusted usage score. We then use this adjusted frequency to determine the most characteristic ingredient for each cuisine which are presented in Table 2.

Cuisine	Chinese	English	French	German	Greek	Indian
Ingredient	soy sauce	potato	butter	pork	oregano	turmeric
Cuisine	Italian	Japanese	Mexican	Moroccan	Spanish	Thai
Ingredient	parmesan cheese	rice wine	tortilla	cumin	sweet pepper	fish sauce

Table 2: Most characteristic ingredient for each cuisine.

In order to gain insight on the relationship between different ingredients, we compute the Pearson's Correlation Coefficient, ρ, between all pairs of ingredients. To identify linear relationships, we select all pairs of ingredients that have a $\rho \in [-1, -0.7[\cup]0.7, 1]$, as we reason that -0.7 and 0.7 are good thresholds. There were no pairs with a negative linear relationship ($\rho \in [-1, -0.7[$) and only 5 pairs with a positive linear relationship ($\rho \in]0.7, 1]$). These 5 pairs are presented in Appendix D, along with the corresponding coefficient.

Finally, in hopes to better visualize the data, we perform dimensionality reduction on the data set using PCA. The number of features reduced from $d = 709$ to $k = 175$ as that allows for a cumulative explained variance of 90%. The data with reduced dimensionality is presented in Figure 1, where each dot in the 2-D plane represents a recipe. We can observe that most recipes from Asian cuisines (Indian, Japanese and Thai) appear more on the right of the plot, whereas recipes from European cuisines (English, French, German, Greek, Italian and Spanish) appear more on the left. Therefore, we see an apparent relationship between recipes from European cuisines as well as between recipes from Asian cuisines.

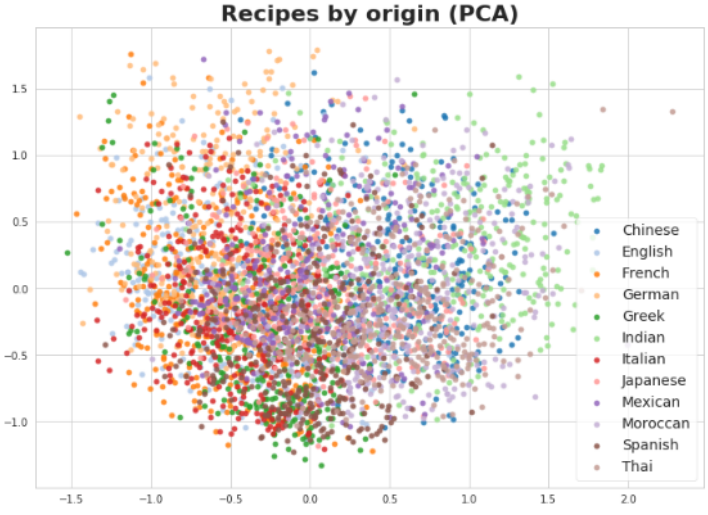

Figure 1: Dimensionality reduction with PCA ($k = 175$).

4 Methodology

This section will describe all experiments performed to discover the best performing classifier for predicting cuisines as well as recommending ingredients to incomplete recipes. All of the experiments for the classifier and collaborative filtering recommendation system use methods for estimating the generalization performance. We use cross-validation for both the classifier and the recommending systems, with 5 folds for the classifier and 3 for the recommending system. The recommending system is tested using Surprise's cross validation method, which uses the Root Mean Squared Error as a performance metric. This regression method is used in a classification task due to the fact that it is a highly accepted evaluation measure for recommendation systems of the predicted rating and the actual true rating [10]. This proves to be a good measure of accuracy while not assuming the purpose of the recommendation system, even if there are more variables involved in making decisions to recommend an ingredient [7].

4.1 Classifier

The first task involves building machine learning models that predict the type of cuisine for a given recipe. We consider three well-known algorithms: a Random Forest classifier, a Support Vector Machine (SVM), and a Naive Bayes Classifier. We use these classifiers not only because they are widely used but also because they have been used in a similar task as ours [6]. This allows us to evaluate each separately and decide on the best model to make the predictions. Naive Bayes is chosen due to its simplicity. Although Random Forest does not always guarantee high performance, as shown in the research mentioned above, it is excellent at recognizing unfamiliar patterns within a dataset and will be considered [6]. Random Forest is ensemble learning. This means there is not only a single instance of a trained model, but there are many instances, and the predictions are averaged between all model predictions [2]. We also implement a Support Vector Machine due to its efficiency and ability to handle nonlinear data. Support Vector Machines attempt to separate classes inside the dataset by maximising a margin, only keeping count of the data points that are closest to the margins, generally referred to as the support vectors. Usually, Random Forest algorithms are more suited for multi-class problems, as is this one, and Support Vector Machines are to binary classifications. This means in order to make predictions, the classes had to be previously one-hot encoded.

4.2 Collaborative Filtering

Secondly, we evaluate memory-based approaches versus model-based for collaborative filtering. We consider the ratings to be 1 if the item is in the recipe and 0 if it is not present. Memory-based filtering is one of the collaborative filtering techniques that is simple and effective [12]. We use the item-based K-Nearest Neighbours and some of its variants as our memory-based algorithms. Essentially this algorithm predicts the rating of a given ingredient by considering the average (weighted by similarity) rating of its k nearest neighbours, which are the most similar ingredients to it. There are disadvantages to keep in mind with memory-based techniques, such as the need to compute a similarity matrix that contains all the pair-wise similarities among ingredients [12]. This is the most computationally expensive component of the algorithm.

Typical model-based algorithms involve the use of clustering, Markov Decision Processes (MDP), and SVD decomposition. We focus on SVD-based algorithms which are based on factorizing the ratings matrix. Specifically, the decomposition is described by the following equation 1:

$$S = U\Sigma V^T \tag{1}$$

where Σ is a pxp diagonal matrix containing the p largest singular values, U is the pxp matrix containing the corresponding left-singular vectors, and V is an nxn orthonormal matrix containing the right-singular vectors. [3].

One possible limitation of this approach is the potential loss of information due to the dimensionality reduction inherent in the decomposition. However, it has been noted that this technique is useful when dealing with highly sparse data [1].

In our research, we compare variations of K-Nearest Neighbors for the memory-based and SVD as the model-based approach. We do not consider other variations of SVD, such as SVDpp, due to our data

not relying on implicit rankings. We use the Surprise framework, a Python based recommender system to implement both collaborative filtering techniques [4]. All the techniques used are item-based, meaning we compare similarity metrics between items based on ratings.

Specifically, we use the Surprise memory-based implementations for the different K-Nearest Neighbor(KNN) variants: KNNBasic, KNNWithMeans, KNNWithZScore, and KNNWithBaseline. KNNBasic implements the base KNN algorithm for collaborative filtering. KNNWithMeans takes into account the mean ratings of each recipe. KNNwithZScore normalises the ratings using the z-score from for each recipe. And, lastly, KNNWithBaseline adds a bias term to each rating. Given the computational cost of each algorithm, we use the default values for most of the hyperparamaters of each model to perform our model selection (see Section 5). Specifically, for the KNN variants use the following number of neighbours: $k = 40$. We evaluate different similarity metrics as discussed in more detail in Section 4.3. For the SVD decomposition we use 100 factors ($p = 100$).

4.3 Similarity Measures

Finally, we wanted to evaluate the effect of using distinct similarity measures on collaborative filtering. We consider the cosine similarity, the Pearson correlation coefficient and the mean squared difference (msd).

4.3.1 Cosine similarity

The first similarity measured we test is the cosine as proposed by Sarwar, 2001 [9]. Since the tests are ingredient-based, instead of recipe-based, we use the formula shown in Equation 2. The similarities can either be calculated between recipes or between ingredients, and the choice has a massive impact on the results of the predictive algorithms. In this similarity, the two ingredients are considered to be data points in a m-dimensional ingredient space. Their similarity is calculated by the cosine of the angle between those points. In the equation, "R" stands for the recipe, "i" and "j" are the two ingredients being compared, "r" is the ranking of that ingredient for that recipe, and "$\cdot$" stands for the dot product of the two data points.

$$cosine\_sim(i,j) = \frac{\sum_{u \in R_{ij}} r_{ui} \cdot r_{uj}}{\sqrt{\sum_{u \in R_{ij}} r_{ui}^2} \cdot \sqrt{\sum_{u \in R_{ij}} r_{uj}^2}} \tag{2}$$

4.3.2 Pearson correlation coefficient

The second similarity measure, the Pearson correlation coefficient is a modified version of the cosine similarity, where the values are mean-centered. It was chosen as proposed by Spartus et al, 2016 [11], where they demonstrate that the cosine and Pearson similarities outperform other existing similarities. The formula for this method can be seen in Equation 3.

$$pearson\_sim(i,j) = \frac{\sum_{u \in R_{ij}} (r_{ui} - \mu_i) \cdot (r_{uj} - \mu_j)}{\sqrt{\sum_{u \in R_{ij}} (r_{ui} - \mu_i)^2} \cdot \sqrt{\sum_{u \in R_{ij}} (r_{uj} - \mu_j)^2}} \tag{3}$$

4.3.3 Mean Squared Difference

Finally, a third, less common method is tested, which is the Mean Squared Difference (MSD), as recommended by [8]. It shows worse performance in datasets where users did not have many ratings. However, since all our recipes have available ratings for all the ingredients (0 if the ingredient was not present, and 1 otherwise), this problem should not affect the performance and we hence decide to test it. Equation 4 shows how this similarity metric is calculated.

$$msd\_sim(i,j) = \frac{1}{(\frac{1}{|U_{ij}|} \cdot \sum_{u \in U_{ij}} (r_{ui} - r_{uj})^2) + 1} \tag{4}$$

5 Discussion

In this section, we explore multiple possible techniques on both the classification and recommendation tasks in order to find optimal performance. In the recommendation task, we conduct a more exhaustive search by also experimenting with different similarity measures for the memory-based techniques and by removing the biases in the model-based technique. We use the mean performance from the cross-validation to choose the best model for each task. These models are then evaluated on a test set.

5.1 Classification models

For the classification task, we experiment with three models: Naive Bayes, which serves as a baseline, Random Forest and Support Vector Machine. The results from the cross-validation are presented in Table 3. Since Naive Bayes is a very simple model and our data has a large number of features ($d = 706$), the poor performance achieved by the model is expected. For the other two models, we expected the performance to be considerably better. Since Random Forest achieves the best results, we evaluate this model on the test set and achieve an accuracy of **73.8%**.

Algorithm	Validation
Naive Bayes	19.2%
Random Forest	**73.9%**
Support Vector Machine	71.8%

Table 3: Mean classification accuracy on cross-validation for different models. The best validation performance was achieved by the Random Forest model (highlighted in bold).

As mentioned previously in Section 3, we see worse performance errors within Asian cuisines and European cuisines. For instance, the Chinese cuisine is sometimes misclassified with Indian, Japanese or Thai, while the Greek cuisine recipes are sometimes misclassified with Spanish or Italian, in accordance to the results previously found using PCA.

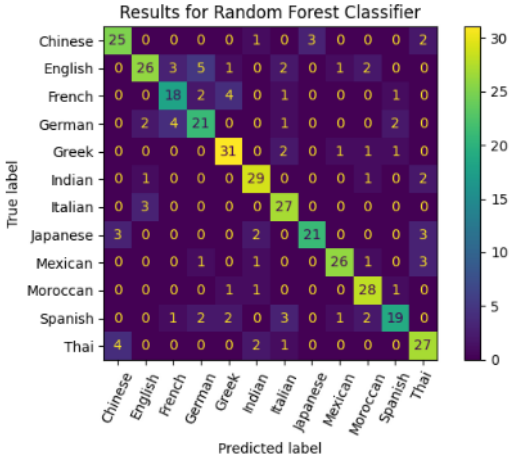

Figure 2: Confusion matrix for the test set results of the best performing classifier, Random Forest.

5.2 Recommendation models

For the recommendation task, we evaluate different collaborative filtering techniques and similarity measures. The results of the cross-validation are presented in Table 4. The performance of the KNN-based techniques is different for each similarity and, in general, using the mean squared difference yields better results. For the model-based techniques, using baselines (or biases) results

in a smaller mean RMSE. Overall, KNNWithZScore achieves the best performance, with a mean RMSE of $\mathbf{0.1090 \pm 0.0003}$.

Type	Model	Method	Similarity	Biased	RMSE
memory-based	KNNBasic	-	cosine	-	0.1797 ± 0.0014
memory-based	KNNBasic	-	msd	-	0.1221 ± 0.0008
memory-based	KNNBasic	-	pearson	-	0.1457 ± 0.0013
memory-based	KNNWithBaseline	-	cosine	-	0.1457 ± 0.0008
memory-based	KNNWithBaseline	-	msd	-	0.1135 ± 0.0003
memory-based	KNNWithBaseline	-	pearson	-	0.1245 ± 0.0002
memory-based	KNNWithMeans	-	cosine	-	0.1462 ± 0.0006
memory-based	KNNWithMeans	-	msd	-	0.1135 ± 0.0004
memory-based	KNNWithMeans	-	pearson	-	0.1253 ± 0.0011
memory-based	KNNWithZScore	-	cosine	-	0.1122 ± 0.0003
memory-based	KNNWithZScore	-	msd	-	0.1146 ± 0.0002
memory-based	KNNWithZScore	-	pearson	-	$\mathbf{0.1090 \pm 0.0003}$
model-based	SVD	-	-	True	0.1139 ± 0.0007
model-based	SVD	-	-	False	0.1174 ± 0.0005

Table 4: Mean RMSE and standard deviation on cross-validation for the different models with different parameters. The best validation performance was achieved by KNNWithZScore model using Pearson's correlation coefficient as the similarity measure (highlighted in bold).

Finally, we evaluate the KNNWithZScore, which is chosen as the best algorithm according to the cross-validation scores. We perform a grid search with cross-validation to find the best value for the number of neighbours parameter, which we find to be $k = 40$. We use a held-out test set consisting of ratings that are not used for the cross-validation procedure outlined above. To choose the held out ratings, first we randomly choose 5% of the recipes in the data set. Then, we remove all the ratings of the ingredients that were not used in the recipe, and randomly choose an ingredient that was used in the recipe as a hidden ingredient and held its rating out for the test set. We consider two settings: one where we compute the RMSE on all the test ratings and another where we compute it only if the rating belongs to an ingredient that is present on the recipe. The RMSE in the first case is 0.0665 and in the second 0.8062. This large difference is due to the fact that the majority of the ratings in the dataset are zero and this biases the model towards lower ratings. Therefore, when we evaluate only on ingredients whose rating is 1, as is the case of the ingredients present in a recipe, the model consistently underestimates the rating, leading to a poor RMSE.

We consider a different form of evaluation where we predict the hidden ingredient rather than comparing the predicted ratings with the test ones. In this case, for each recipe, we predict the ratings for all held-out ingredients from the test set and choose the one with the highest rating as the recommended one. We then compare the recommended ingredient with the hidden ingredient for each recipe and compute the accuracy obtained. Our model selects the correct ingredient among the held-out in 7% of the test recipes. Note that in this instance, the model had to recommend an ingredient among around 700 that were not present in the recipe and so this low accuracy is expected. Below, we show some example partial recipes and the recommendation of the model in Table 5.

6 Conclusion

This paper evaluates and analyzes the prediction of cuisines as well as recommending ingredients to partial recipes. We consider multiple classifiers in order to gain maximum classification performance. We find the highest performance is achieved with a Random Forest Classifier. As for the recommendation system, we explore using memory-based and model-based collaborative filtering approaches. The best technique for collaborative filtering is KNNWithZScore with the default parameter values. Future work can be done to improve our investigations. One should consider applying dimensionality reduction to the dataset before classification. Further research can be done to test more model-based techniques for the recommendation system.

Example 1	
Partial recipe	broth, brown sugar, chili paste, cornstarch, egg, ketchup, olive oil, salt, soy sauce, sugar, vinegar
Recommended ingredient	chicken
Hidden ingredient	chicken
Example 2	
Partial recipe	baking powder, brandy, chicken, egg, flour, ginger, ice water, salt
Recommended ingredient	soy sauce
Hidden ingredient	soy sauce
Example 3	
Partial recipe	cilantro, egg, garlic, ginger, green onion, lemon grass, lime, turkey, water
Recommended ingredient	soy sauce
Hidden ingredient	chile pepper
Example 4	
Partial recipe	basil, chile pepper, green bean, lemon, quinoa, shallot, soybean
Recommended ingredient	chicken
Hidden ingredient	tofu

Table 5: Example correct and incorrect recommendations made by the model.

References

[1] Dheeraj Bokde, Sheetal Girase **and** Debajyoti Mukhopadhyay. "Matrix factorization model in collaborative filtering algorithms: A survey". **in**: *Procedia Computer Science* 49 (2015), **pages** 136–146.

[2] Leo Breiman. "Random forests". **in**: *Machine learning* 45.1 (2001), **pages** 5–32.

[3] Michael Gutmann **and** Arno Onken. "Data Mining and Exploration". **in**: *The University of Edinburgh* (2021), **pages** 25–27.

[4] Nicolas Hug. "Surprise: A Python library for recommender systems". **in**: *Journal of Open Source Software* 5.52 (2020), **page** 2174. DOI: 10.21105/joss.02174. URL: https://doi.org/10.21105/joss.02174.

[5] S. Jayaraman, T. Choudhury **and** P. Kumar. "Analysis of classification models based on cuisine prediction using machine learning". **in**: *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)*. 2017, **pages** 1485–1490. DOI: 10.1109/SmartTechCon.2017.8358611.

[6] Shobhna Jayaraman, Tanupriya Choudhury **and** Praveen Kumar. "Analysis of classification models based on cuisine prediction using machine learning". **in**: *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)*. IEEE. 2017, **pages** 1485–1490.

[7] Sean M McNee, John Riedl **and** Joseph A Konstan. "Being accurate is not enough: how accuracy metrics have hurt recommender systems". **in**: *CHI'06 extended abstracts on Human factors in computing systems*. 2006, **pages** 1097–1101.

[8] JL Sanchez **andothers**. "Choice of metrics used in collaborative filtering and their impact on recommender systems". **in**: *2008 2nd IEEE International Conference on Digital Ecosystems and Technologies*. IEEE. 2008, **pages** 432–436.

[9] Badrul Sarwar **andothers**. "Item-based collaborative filtering recommendation algorithms". **in**: *Proceedings of the 10th international conference on World Wide Web*. 2001, **pages** 285–295.

[10] Guy Shani **and** Asela Gunawardana. "Evaluating recommendation systems". **in**: *Recommender systems handbook*. Springer, 2011, **pages** 257–297.

[11] Ellen Spertus, Mehran Sahami **and** Orkut Buyukkokten. "Evaluating similarity measures: a large-scale study in the orkut social network". **in**: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 2016, **pages** 678–684.

[12] Xiaoyuan Su **and** Taghi M. Khoshgoftaar. "A Survey of Collaborative Filtering Techniques". **in**: *Adv. in Artif. Intell.* 2009 (**january** 2009). ISSN: 1687-7470. DOI: 10.1155/2009/421425, URL: https://doi.org/10.1155/2009/421425

Appendices

A Top 10 ingredients overall.

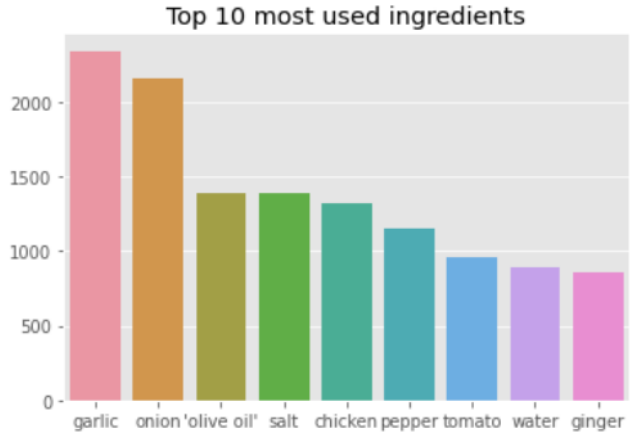

Figure 3: Top 10 ingredients overall

B Top 10 ingredients per cuisine.

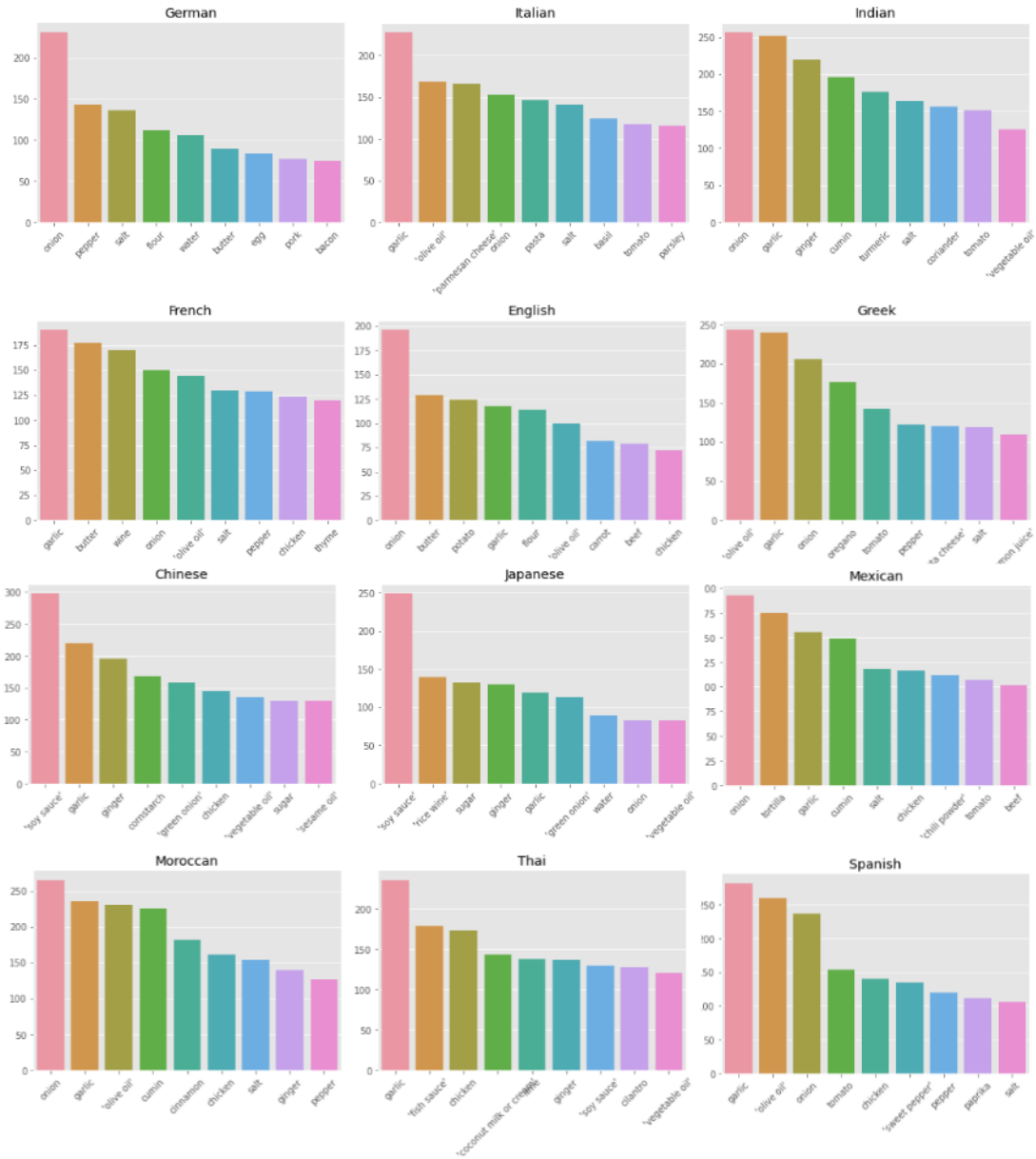

Figure 4: Top 10 ingredients per cuisine

C Mean ingredient count per cuisine

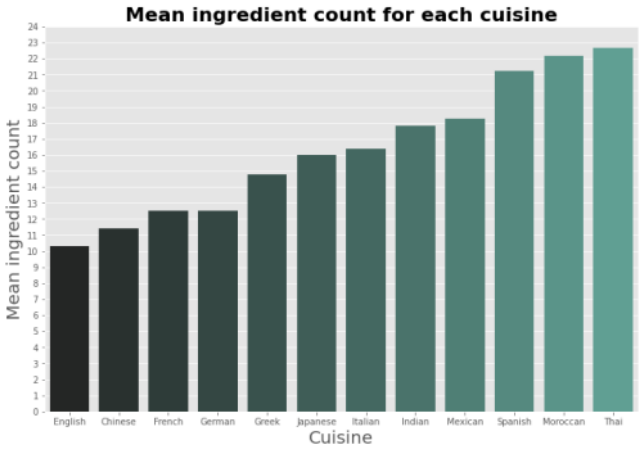

Figure 5: Mean ingredient count per cuisine

D High correlation ingredient pairs

Ingredient 1	Ingredient 2	Pearson's correlation coefficient
Sofrito sauce	Sazon goya	0.816
Coconut oil	Almond butter	0.707
Kamaboko	Burdock root	0.707
Sea cucumber	Chinese mushroom	0.707
Gumbo	French style green bean	0.707

Table 6: Ingredient pairs with the highest Pearson's correlation coefficient.