Serenity

Aditya Sahai, Rachel Black, Samuel Yusuf, Ryan LaFleur

Introduction

Serenity is a sophisticated recipe recommendation system with the ability to provide aspiring chefs recipe results based on the ingredients they have readily on hand. This is not your average recipe suggestion website; Serenity fulfills the conventional requirement of providing recipes, while incorporating advanced algorithms to enhance the quality of results.

Problem definition

Serenity's goal is to aid a user's meal preparation by suggesting recipes that balance the number and the rarity of the additional ingredients required.

Survey

Serenity incorporates multiple research areas, each providing relevant insight. Unfortunately, only a few of the research papers will be relevant to Serenity going forward. We discuss a few literatures below:

[2] presents a recipe recommendation that is based on their personal preferences. However, the method requires that we already have user's data, which we wouldn't have in this iteration of our system. Papers [12] and [13] explore algorithms for measuring similarity. [13] suggests clustering query results to existing recipe clusters, but was unable to boast any significant improvements. Yet [13]'s similarity rankings are much more computationally expensive, so we will instead incorporate ideas from [1].

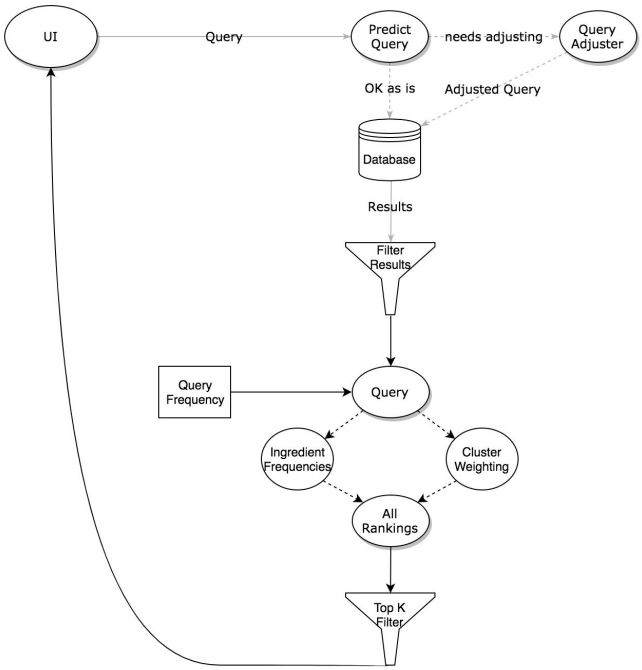
[5] discusses how to use new ingredient combinations to develop new recipes and this can be used in a later part of our project to suggest dishes by substituting ingredients. We do not expect to use the neural networks used in [9] due to the amount of data needed for such techniques. We may add an additional feature that returns results based on user preferences, but do not plan to use the filters mentioned in [10] since we have already decided to use an individual's preferences.

We felt it was important to explore different user interface options, however, decided to use the standard keyboard and mouse[11] since the vast majority of potential users are familiar with these. [7] talks about how D3 is different from other visualization tools as it integrates directly with the web page and rewards prior knowledge of web technologies and this aligns with the visualization requirements of this project. [3] emphasizes the importance of proper arrangement of elements on a user screen.

We did a comparative study between the machine learning libraries Scikit-learn [6] and Caret [8] and decided to pick Scikit-learn for our project because of the ubiquity of Python. To build a robust model using machine learning techniques, we initially explored the classification methods presented in [4] but eventually decided to use Scikit-Learn for regression.

Proposed methods & Innovations

Innovation	**Impact**
Database	Innovative use of JSON Blob datatype for faster overall database use
Predictive Query Model	Innovative model to predict the number of results before querying the database
Ranking Algorithm	Innovative exploration and implementation QF-IDF for ranking recipes. An extension of this would include quantity based ranking as well.



Above: Diagram of Serenity Architecture

Existing systems do not check if a query will be fruitful before querying. We have built a multilayer perceptron regressor which predicts the number of recipes that will be returned by the database based on the input query of ingredients. This list of ingredients is the input to the regressor. If the regression predicts that too few results would be returned, then the least frequently occurring ingredient is removed from the query and the regression is consulted again. This process is repeated until the regression says that the returned set of recipes will be rich enough for our purposes (10 in our case). This improves our runtime as we avoid re-querying the database multiple times. Predictive querying becomes useful especially as the database scales from thousands of recipes up to hundreds of thousands of recipes.

This machine learning model is one of our most innovative features. We experimented with four different models: k-nearest neighbors, linear regression, random regression forest, and neural networks. We generated data by creating random queries and querying the database and intersections of ingredients. The relationship may be nonlinear, hence k-nearest neighbors, random forests, and neural networks might do well. As discussed in the Experiments section, the neural network was the most accurate and was therefore the model we ultimately used.

Serenity needs an efficient database implementation to work well. The source of our recipe data is Spoonacular. It returns recipe data in JSON format (figure below). The JSON has ingredients and instructions fields which are of variable lengths.

Similarly, each ingredient could be involved in multiple recipes. In terms of database concepts, these fields are multivalued fields. To achieve highest normal form in a standard relational schema implementation would mean setting up multiple tables leading to high query complexity. There is also the additional cost of cleaning the incoming JSON such that all the tables could be updated.

During our research we came across a serendipitous new data field supported by some of the newer databases like PostgresDB called json and jsonb. These fields allow us to store JSON objects directly into the database and also has SQL extensions to query/alter JSON objects directly. For example, the following query in PostgreSQL adds an element to the json object array “recipes” for ingredient id 10311529, instead of painfully implementing it in a programming language (‘||’ is the concatenation operator).

```
 UPDATE "ingredients" SET "recipes" = "recipes"::jsonb || '[248732]'::jsonb WHERE "id"
= 10311529;
```

This makes the implementation of our database much easier. We chose to use PostgreSQL for our database which is an open source relational database. To summarize, we have the following advantages with this approach,

- Easier to store data
- Simple to implement
- Data stored in its native form without too many changes
- Fast data query
- Fast data update
- Since, most modern programming languages support json objects as dictionaries, querying and using these json objects becomes much easier

Recipes

id	isVeg	isVegan	isGlutenFree	Is Dairy Free	title	Cooking time	Ready Time	Price Per Serving	servings	Popularity score	image (json)	Dish type (json)	Instructions (json)	Ingredients (json)

Ingredients

id	name	type	Image (json)	Recipes (array)

Figure: Schema Diagram

We used Spoonacular.com’s API to gather over twenty-thousand recipes for our database. Due to cost and time constraints we did not gather the largest quantity of data possible, but rather an amount that led to the construction of an effective prototype. Our

data consisted of 15 attributes which are listed in the above table. In order for our ranking algorithms to work correctly we had to clean the data in OpenRefine. In OpenRefine we converted all the different units and quantities for ingredients to a standard unit - ounces. For instance, if there was a unit for which there was no official conversion to ounces e.g. “packages” we query Spoonacular.com’s API for the conversion in real time.

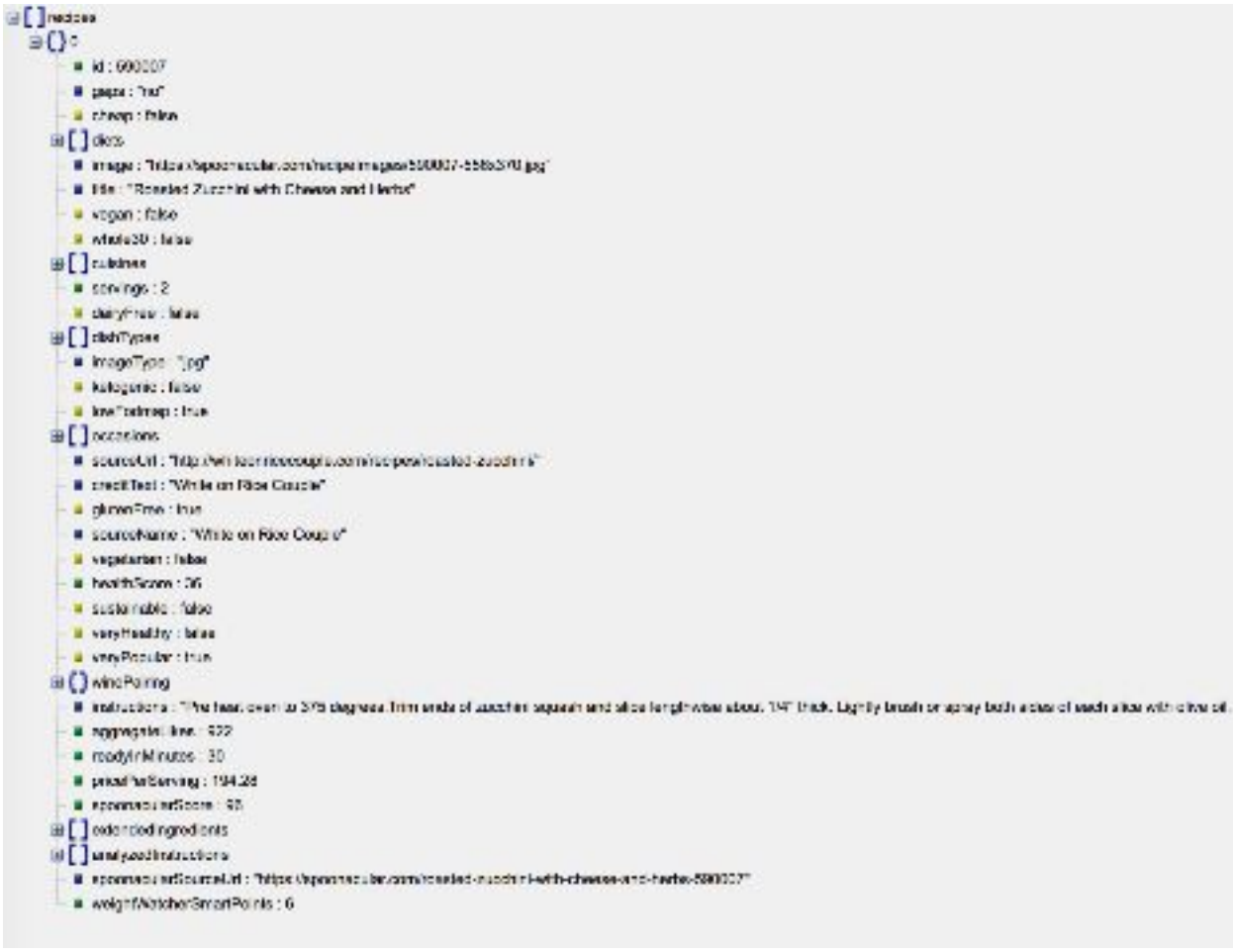
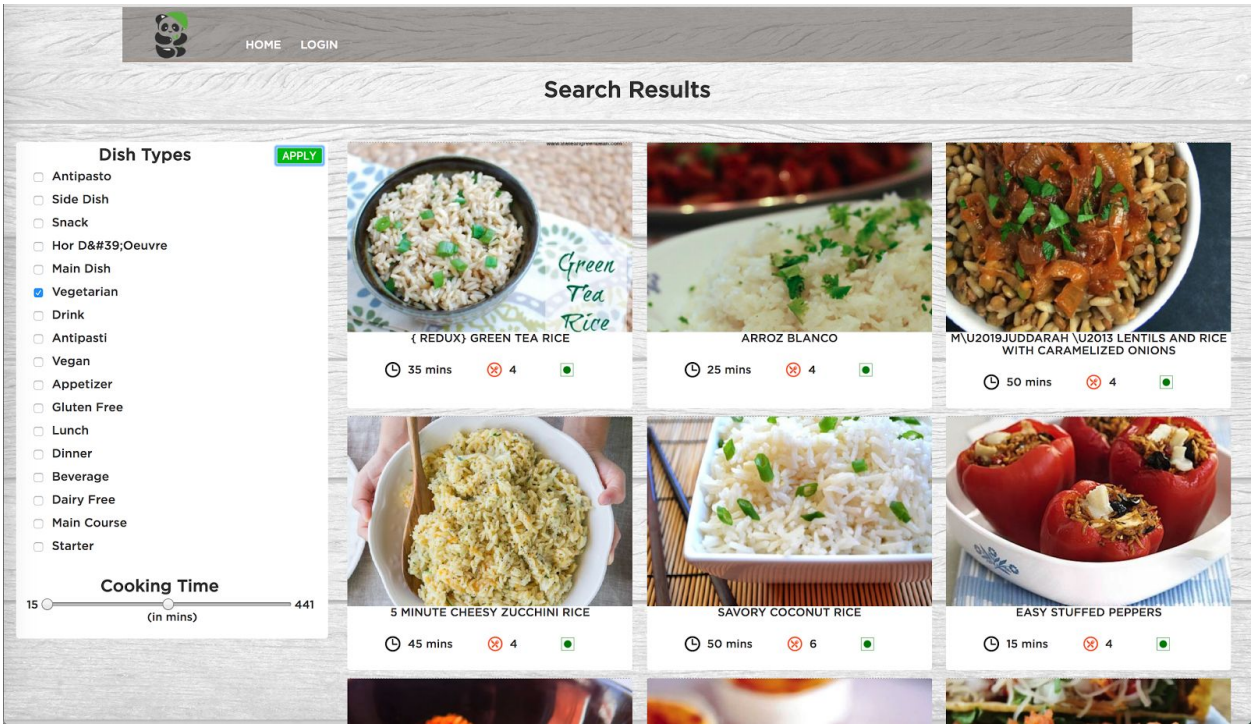


Figure: Sample JSON Recipe object.

Another key challenge for Serenity is for it to be able to rank results from the database. A query-recipe pair has two types of ingredients, those which are not common between them and those that are. The ingredients which are uncommon are not affected by the quantity specified by the user, and simply incur a ranking penalty proportional to the frequency of the ingredient in the database and the frequency with which the ingredient is queried by the user (QF-IDF[1]). Unique to our system, if the ingredient is common to both, then the weight is scaled by the quantity of the ingredient used, such that if the user has half the necessary amount, then the ranking penalty will

only be half. We also account for synonyms so that the frequency of an ingredient is equal to the sum of all the frequencies of its synonyms. These methods effectively rank by displaying recipes that use the most ingredients the user provided and the least amount of additional ingredients. Furthermore, if additional ingredients are needed they are typically common ingredients so that the user is more likely to already have these in their kitchen.



Above: A picture of the Serenity search results screen for the query of rice and chilli flakes. Each picture of the recipe can be clicked to expand the full recipe details including ingredients, cooking time, instructions, etc. The left panel has been used to filter so only vegetarian recipes taking 15 to about 200 minutes to prepare are displayed.

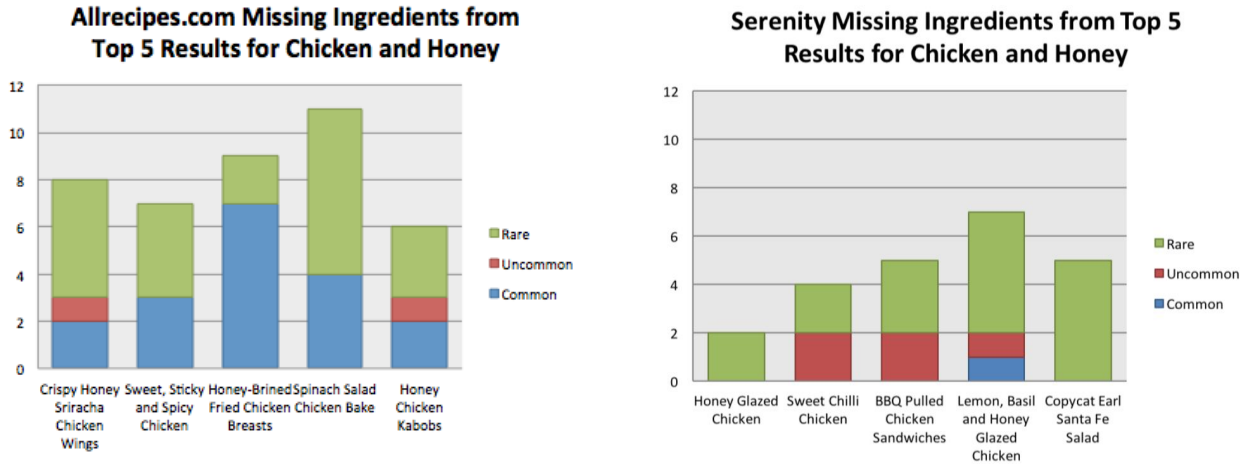
The top ranked results are then displayed on the user interface. A user can click on any of the recipes and a box describing the ingredients and instructions will then appear. Also, on the results page are two filtering mechanisms. The first has checkboxes to filter out any recipes that do not match the dish type. The second is a slide bar to hide recipes which take longer than desired to prepare.

Experiments / Evaluation

Our testbed incorporates a combination of quantitative and qualitative assessments of our ranking mechanism. Quantitatively, after each pass of ranking we

calculate how many ingredients requested by the user are missing, how many ingredients from each recipe are not in the query, a breakdown of the rarity of the missing ingredients, and a generated score value. A purely quantitative approach fails to capture the intricacies of the problem. The system should be recommending recipes which meet the user's expectations in order to be successful. We employ qualitative evaluations to confirm the system is behaving well.

Qualitatively, we can observe whether the dishes returned are the recipes we are expecting to see. The following experiment highlights how qualitative assessment can add to our evaluation. The following two charts detail results from our algorithm and Allrecipes.com for the query "chicken breast" and "honey". The top result is a dish of chicken garnished with honey, exactly what we expected. Also, notice that our top results are typically missing far fewer ingredients than those recommended by Allrecipes.com. It is also interesting to observe that the top suggested recipes from our competitor dance around the expected results. It is not until the fifth result that we receive a result which is strictly a honey garnished chicken dish. Serenity thus gives result which are more to the point than Allrecipes.com, while simultaneously balancing the number of additional ingredients needed. We hypothesize that Allrecipes.com uses an algorithm that heavily weights the rating of a dish that trades off on how well the recipe matches the user's query.



Above: Visual representations of the number of missing ingredients from the search results of Allrecipes.com (left) and Serenity (right)

As seen in the above charts, we broke down the membership of missing ingredients into rarity classes. This was done by using k-means and clustering on the frequency of the ingredients. However, after manual inspection of the clusters, the vast majority of ingredients were labeled as "rare"; there was practically no difference

between the most common rare ingredients and the uncommon and common categories. Therefore, the relative rarities of the missing ingredients is rather devoid of meaning.

In the progress report, we proposed an alternative method for generating the ranking score. In lieu of using the frequencies of an ingredient and its synonyms, we suggested using the ingredients' rarity clusters instead. As discussed above, these clusters proved to be rather devoid of information. They also proved to be less robust to the ranking problem, leading us to opt for pure frequencies as our method.

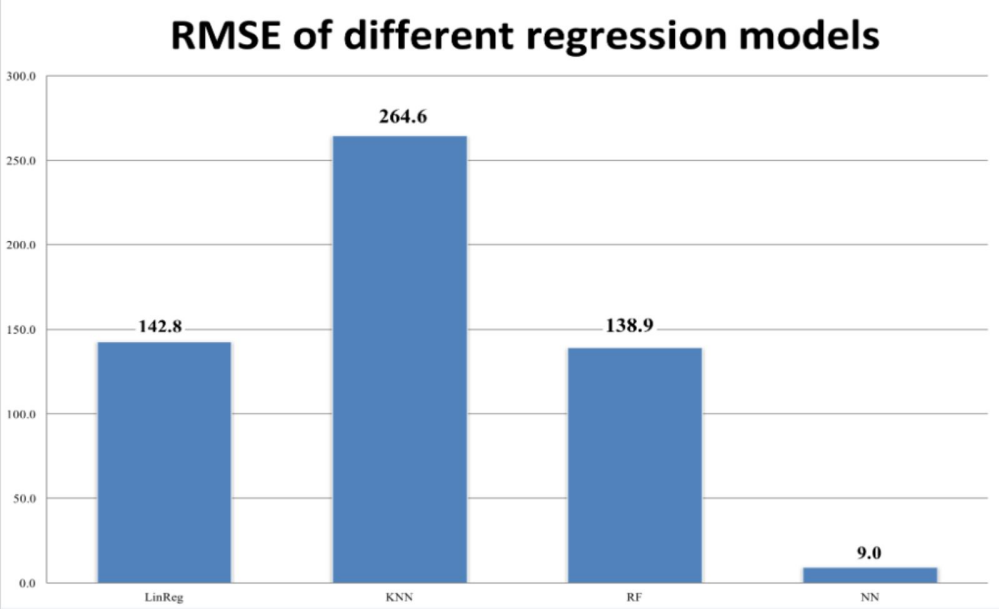
Continuing on in the vein of quantitative evaluations, we used these types of methods to evaluate the effectiveness of our quantity ranking mechanism. To show the impact of the mechanism, we ran three queries. The first query was for chicken without using quantity weighting, the second queried for chicken and honey without quantity weighting, and the third queried for chicken and honey assuming an unlimited supply of chicken while having only an eighth tablespoon of honey. The results of the first two queries had no overlap but both were made entirely of chicken dishes, just with differing garnishes. However, when honey is queried for but in a very small quantity, the top result is the same as the top result from the honey query, but the rest come from the first query. This shows that having some of the ingredient causes the recipe using the honey to be ranked higher, but not having enough of it causes most of the query results to resemble those of the query only containing chicken. The results of these queries can be viewed in the following table.

Chicken (No Quantity)	Chicken, Honey (No Quantity)	Chicken, Honey (Quantity)
Chicken Marsala	Honey Glazed Chicken	Honey Glazed Chicken
Honey Glazed Chicken	Sweet Chilli Chicken	Chicken Marsala
Grilled Chicken Calzones	BBQ Pulled Chicken Sandwiches	Grilled Chicken Calzones
Classic Roasted Chicken	Lemon, Basil and Honey Glazed Chicken	Classic Roasted Chicken
Bacon Wrapped Guacamole Stuffed Chicken	Copycat Earl Santa Fe Salad	Bacon Wrapped Guacamole Stuffed Chicken
BBQ Chicken with Bacon and Cheddar	Grilled Chicken with Chilli and Sesame Seeds	BBQ Pulled Chicken Sandwiches
Slow Cooker Buffalo Chicken Sandwiches	Baked Honey Lemon Chicken	Slow Cooker Buffalo Chicken Sandwiches
Poached Chicken	Chicken Piccata	Poached Chicken
Oven Baked Chicken Strips	Caramelized Chicken Breasts	Oven Baked Chicken Strips

Above: Visual results for the quantity experiment. The far left column shows top results from a query for chicken breast without taking quantity into account. The middle column shows top results for a query for chicken breast and honey disregarding the quantity. Lastly the far right is a query for chicken and honey using unlimited chicken but only one eighth tablespoon of honey.

To evaluate the predictive querying model, we measure the root mean squared error of several models to compare their effectivenesses. Candidate models included linear regression, random forests, k-nearest neighbors and neural networks. As seen in the following chart, the RMSE of the neural network was substantially lower. This is likely due to the nonlinear relationship and the sparsity of the data. With sufficient training samples, the neural network was the best balance of predictive efficiency and accuracy.

In the progress report, we reported nearly flawless results by our linear regressor. However, we discovered a bug in the code that resulted in the previous models being trained with the target value as a feature, thus leading to near perfect linear regression. The results reported here reflect the corrected models.



Above: Root mean square errors (RMSE) of out-of-sample training data points for linear regression (first), K-nearest neighbors (second), random forest (third), neural network (fourth) models

Conclusion & Discussion

Discussion

Our team feels that there are aspects that could be improved on going forward in Serenity version 2.0. With more time, we would improve our machine learning algorithms by exploring, in detail, converting the regression problem to a classification problem. It was also revealed upon inspection that the model was trained on class imbalanced data. This does not particularly matter in a regression problem, but it suggests that a classification model could be improved by taking actions to counteract class distribution effects. Furthermore, we will do some more data cleaning so that we would have a conversion table between the different quantity units, allowing us to not make queries to the Spoonacular.com conversion API when rare units are used in a recipe.

We would also scale up our application by adding more recipes to the database from the Spoonacular API. We had limited access to the API due to a subscription fee, so the number of recipes we could get was limited from around three hundred thousand down to twenty thousand. On the user experience side, we would include the ability for multiple users to access the site and update the UI design accordingly.

Conclusion

An especially relevant lesson we learned while working on Serenity is the effect large scale data has on every mechanism of a project. It is important to optimize approaches

so that only a portion of the data has to be calculated at a time, otherwise the responsiveness of the application to a user can be compromised. We also have grown to appreciate the role machine learning can play in large data system by elegantly consolidating information from a massive amount of data into a compact model. We will carry the lessons learned about data preprocessing, management, and computation as we move forward in our academic and professional careers.

Distribution of team member effort

All team members contributed a similar amount of effort.

APPENDIX

Citations

[1] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis, “Automated Ranking of Database Query Results,” Proc. Conf. Innovative Data Systems Research (CIDR), 2003.

[2] Ueda Mayumi., Takahata Mari., and Nakajima Shinsuke. “User’s Food Preference Extraction for Personalized Cooking Recipe Recommendation.”

[3] Bahadur, Surya, et al. “User Interface Design With Visualization Techniques.” *IJREAS*, vol. 2, no. 6, June 2012, pp. 51–62.

[4] Kotsiantis, S B. “Supervised Machine Learning: A Review of Classification Techniques .” *Informatica*, 2007, pp. 249–268.

[5] Ahn, Y. Y., Ahnert, S. E., Bagrow, J. P., & Barabási, A. L. (2011). Flavor network and the principles of food pairing. *Scientific reports*, *1*.

[6] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(Oct), 2825-2830.

[7] Bostock, M., Ogievetsky, V., & Heer, J. (2011). D^3 data-driven documents. *IEEE transactions on visualization and computer graphics*, *17*(12), 2301-2309.

[8] Kuhn, M. (2015). A Short Introduction to the caret Package. *R Found Stat Comput*, 1-10.

[9] Teng, C., Lin, Y., & Adamic, L. A. (2012). Recipe Recommendation Using Ingredient Networks. *Proceedings of the 3rd Annual ACM Web Science Conference on - WebSci 12*. doi:10.1145/2380718.2380757

[10] Patil, S., & Ansari, M. B. (2015). User Profile Based Personalized Research Paper Recommendation System Using Top - K Query. *International Journal of Emerging Technology and Advanced Engineering, 5*(9). Retrieved October 15, 2017, from http://www.ijetae.com/files/Volume5Issue9/IJETAE_0915_35.pdf

[11] El-Bakry, H. M., Riad, A. M., Abu-Elsoud, M., Mohamed, S., Hassan, A. E., Kandel, M. S., & Mastorakis, N. (2010, February). Adaptive user interface for web applications. In Recent Advances in Business Administration: Proceedings of the 4th WSEAS International Conference on Business Administration (ICBA'10) (pp. 20-22).

[12] Jones, K. S. (2004). A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation, 60(5), 493-502.

[13] Tombros, A., Villa, R., & Rijsbergen, C. V. (2002). The effectiveness of query-specific hierarchic clustering in information retrieval. Information Processing & Management, 38(4), 559-582.