FooDIS - Food.com Data, Information and Search

ANTÓNIO BEZERRA (UP201806854), GONÇALO ALVES (UP201806451), and PEDRO SEIXAS (UP201806227)

This work deals with the retrievement, refinement, analysis and querying of datasets regarding the website **Food.com**. The end goal is to create a search system capable of answering non-trivial queries that are not feasible through the website. To achieve this, the data sources are explored in order to extract information about all the recipes and reviews. This data is then refined through a process described within this paper and finally, it's analyzed through exploratory techniques. Next, through the use of Solr, we define a collection and its respective documents and describe their indexation. With this, the previously identified retrieval tasks are answered and evaluated according to their precision and recall. Finally, using Solr's plugins and machine learning techniques, we develop the search system to answer the remaining retrieval tasks and implement a user-friendly interface using Web technologies.

Additional Key Words and Phrases: datasets, food, reviews, recipes, information processing, information retrieval, solr, indexing, statistical analysis

ACM Reference Format:
António Bezerra (up201806854), Gonçalo Alves (up201806451), and Pedro Seixas (up201806227). 2021. FooDIS - Food.com Data, Information and Search. 1, 1 (January 2021), 14 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Food is a necessity to us as humans. What started as simple as hunting and gathering developed side by side with humanity itself, being refined by the discovery of fire and preservation methods. Food is also a great way to identify cultures and to connect friends and family over a dinner table. As such, it made perfect sense for us to take a deep dive into this timeless topic, especially since the pandemic has increased the amount of cooking and baking people do in their homes (social media was booming with content related to this topic [1]). Hence, the goal of this paper is to describe a search system that can help every level of chef obtain a recipe in an easier and faster way. This paper is split into three parts. The first, Data Preparation (2) , starts by describing the datasets and their refinement. Their quality is also assessed through exploratory techniques and some modifications are then performed, to obtain a final version of the dataset. This exploratory analysis allows us to have a better understanding of its values, distributions and patterns. Next, a conceptual model is presented to represent the final version of our dataset. This model depicts all the entities present in our domain, as well as the associations between them. Finally, some retrieval tasks are identified in order to represent some of the possibilities our final product is expected to have. The second part, Information Retrieval (3), starts with a description of our collection and their documents, as well as their indexation. Then, some of our previously identified retrieval tasks are then answered and the results are evaluated, reflecting the quality of our results.

Authors' address: António Bezerra (up201806854), up201806854@up.pt; Gonçalo Alves (up201806451), up201806451@up.pt; Pedro Seixas (up201806227), up201806227@up.pt.

2 DATA PREPARATION

Before finding a dataset, we started by defining what information we wanted to retrieve. Since food is such a broad topic, encompassing such things as diets, nutritional information, types of food (fruit, meat, dairy, ...), we felt that the best way to try and condense that much information would be with recipes. Finding sources on recipe information (i.e., ingredients, cooking time, nutritional information) is not difficult, as many sources already collect these properties, since this is often what people are looking for. As such, we came upon a Kaggle competition[2], where data, such as recipes and reviews, from the website Food.com[3] was collected. This was a good source since the dataset was easy to understand and the dataset was being actively maintained, as we can see from the usability score. After a brief analysis of the datasets, we concluded that both these datasets would result in a challenging but interesting project, not only by their size but by the information they contained. They have shown to be two very complete datasets, especially the recipes' dataset which had information not only about ingredients or instructions but also about calories and macro-nutrients.

Technical details:

- recipes
  - Size: 704.21 MB
  - Columns: 28 columns (16 textual, 8 numerical,2 dates)
  - Rows: 523 thousand
- reviews
  - Size: 496.1 MB
  - Columns: 8 columns (2 textual, 1 numerical ,2 dates)
  - Rows: 1.40 million

2.1 Data Pipeline

After this preliminary analysis, the first step was to extract users, categories, keywords, ingredients, and images into their separate tables. Besides being good practice from a database standpoint, this also allowed us to more easily explore the data and eliminate redundancy from our data. The next step was the cleaning of the data. To achieve this, we removed duplicate values, irrelevant columns and some outliers. Since a user can be a recipe and review author, when extracting the users from the original files, various duplicates were found and then promptly removed. The ingredients' quantities were removed since there were inconsistencies and because in the instructions the user can have a pretty good notion of how much of each ingredient. The outliers removed concerned themselves with some absurd caloric values (>50.000), and as such, it made no sense for them to belong in our dataset. After this, we proceeded to normalize some columns, such as dates or times, as the original file had them with special encodings. Finally, we thought it best to create a new column with the URL to the webpage of the corresponding recipe. The pipeline depicted below, Figure 1, summarises the described process.

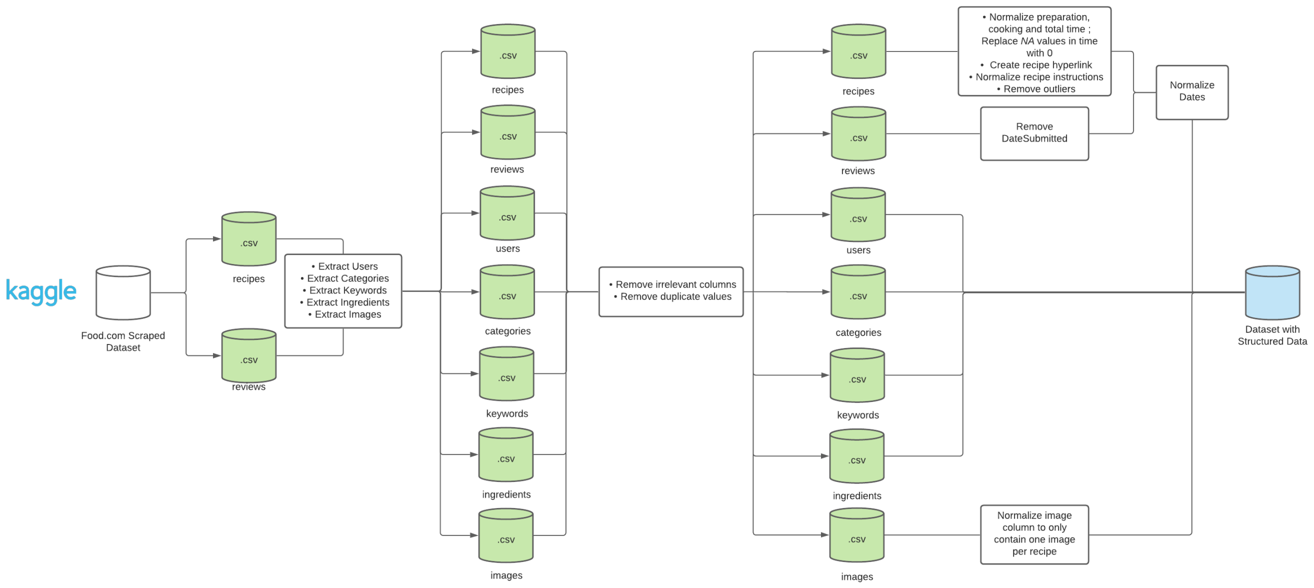

kaggle
Food.com Scraped Dataset
.csv
recipes
reviews
• Extract Users
• Extract Categories
• Extract Keywords
• Extract Ingredients
• Extract Images
users
categories
keywords
ingredients
images
• Remove irrelevant columns
• Remove duplicate values
• Normalize preparation, cooking and total time ; Replace NA values in time with 0
• Create recipe hyperlink
• Normalize recipe instructions
• Remove outliers
Remove DateSubmitted
Normalize Dates
Normalize image column to only contain one image per recipe
Dataset with Structured Data

Fig. 1. Final Pipeline

2.2 Dataset Characterization

Our dataset can be mainly divided into two parts, the **recipes** and their **reviews**. To understand data patterns and assess if they are semantically expected, we made some exploratory analysis.

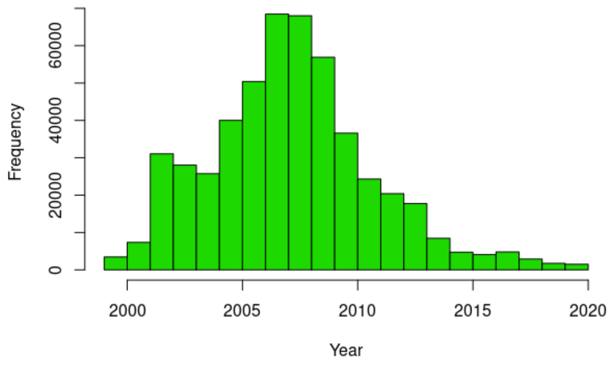


Fig. 2. Recipes Histogram

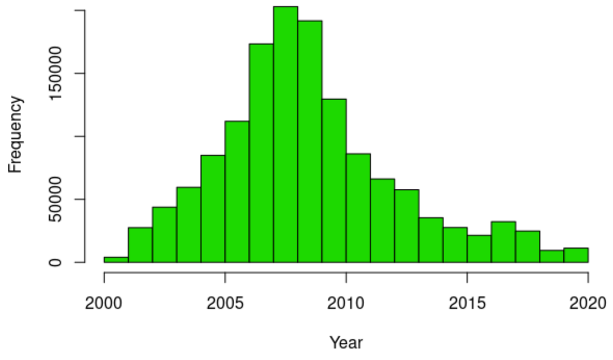


Fig. 3. Review Histogram

As for the **recipes**, our dataset has more than 500 thousand entries, published in more than 20 years. Looking at Figure 2 we can see that the number of recipes uploaded to **Food.com** peaked 14 years ago, in 2007, with more than 60 thousand recipes being uploaded in a single year. This was due to the website's popularity at that time. After that, we can see a significant decrease in the number of recipes uploaded to the website, being currently on an all-time low, according to this dataset. This doesn't mean that the website has no current active users since there can be many recipes uploaded to the website that are not being scrapped by the creator of the dataset.

When talking about **reviews**, this dataset contains over 1.4 million reviews. Figure 3 looks very similar to the **recipes**' one (2), even though the frequency values are a lot higher which is expected since the reviews are scrapped from the recipe's page and the recipes can have tens or even thousands of reviews.

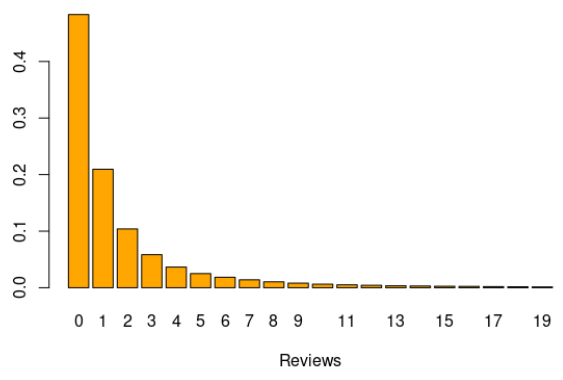


Fig. 4. Reviews per Recipe Histogram With 0

As we can see from Figure 4, more than 50% of the recipes don't have any reviews. To get this graph, we needed to filter the recipes, as some had more than a thousand reviews, being undeniable outliers. Without those recipes, the resulting graph shows a clear exponential decrease in the percentage of recipes, as their number of reviews increase. To get a better understanding of how was the exponential pattern, the recipes that did not have any reviews were removed, temporarily, to create a graph shown in Figure 5.

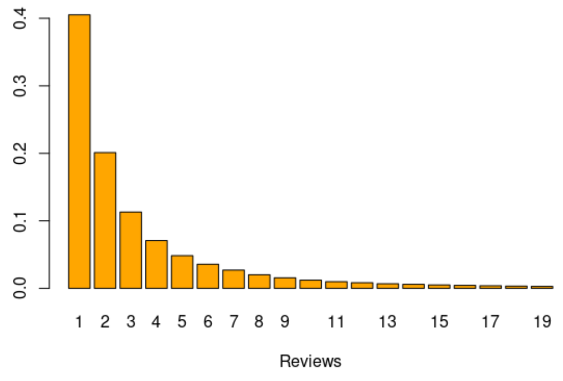


Fig. 5. Reviews per Recipe Histogram Without 0

Even without the recipes that did not have any review, we could see the exponential decrease. This was also verified if we filtered out the recipes with 1, 2, 3, 4 and so on, which resulted expectedly.

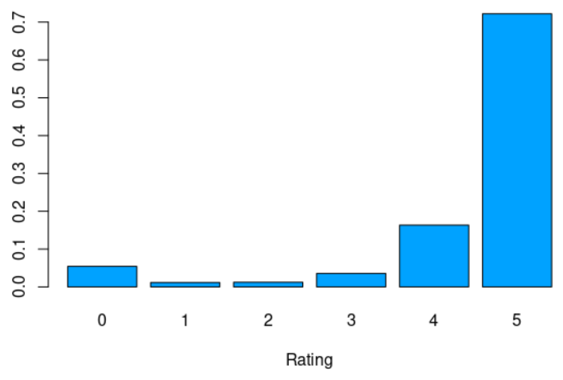


Fig. 6. Rating Histogram

After analysing the number of reviews, we thought it was interesting to find out what was the rating of these reviews to see if they are mostly positive or negative. With the graph in Figure 6, we can see that the number of positive reviews is significantly higher than the negative ones. One curious statistic is the fact that there are almost no reviews with 1, 2 or 3 stars, indicating that the users tend to use the extremes when reviewing, either positively or negatively.

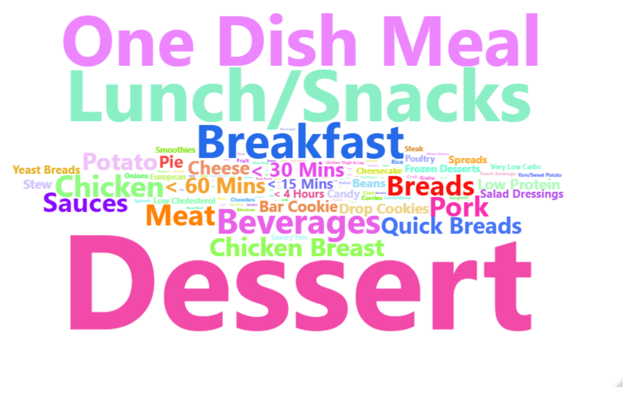


Fig. 7. Categories Word Cloud

After all the analysis in the reviews dataset, we found out that it would be interesting to know, since most of the reviews are positive, which categories of recipes were the most popular. For that, we made a word cloud (a great type of graph to measure the frequencies of certain words). In this word cloud, 7, we have some categories that are more frequent than the others, such as "Dessert" or "Lunch/Snacks". This statistic has just confirmed the thoughts at the start of this project, that there would be a lot of recipes about cakes and other traditional dishes. There are over 300 categories in this dataset, so this word cloud can not demonstrate the full list of categories.

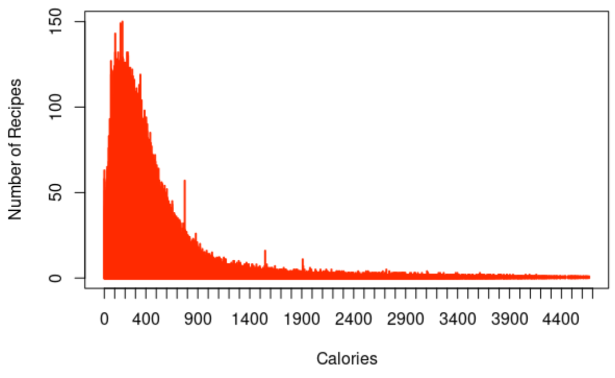


Fig. 8. Calories Per Recipe

With the categories in mind, another statistic that we could study would be the total calories of the recipes. As two of the main categories were "One Dish Meal" and "Lunch/Snacks" we already expected to have the highest frequency at around 300 calories, which is the normal calorie count for a complete meal and the graph in Figure 8 is demonstrating just that. The higher calorie recipes are mainly desserts or recipes with a high number of portions.

2.3 Data Domain Conceptual Model

The conceptual data model describing the collected data can be found below:

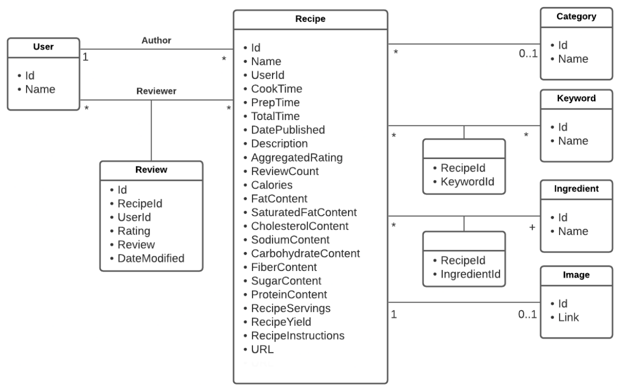


Fig. 9. Data Conceptual Model

The **Recipe** class is in the centre, as it is the main focus of our dataset. A recipe has an Author, can have a Category and an Image. It can also contain several Reviews, Keywords and Ingredients.

Recipe has the following attributes:

- **Id** : Unique recipe ID;
- **Name** : Recipe name;
- **UserId** : Recipe's author ID;
- **CookTime** : Time it takes to bake the recipe, in seconds;
- **PrepTime** : Time it takes to prepare the recipe, in seconds;
- **TotalTime** : Time it takes to prepare and bake the recipe, in seconds;
- **DatePublished** : Date that the recipe was published in the website;
- **Description** : Description of the recipe;
- **AggregatedRating** : Review ratings mean;
- **ReviewCount** : Number of reviews;
- **Calories** : Number of calories;
- **FatContent** : Fat in the recipe, in grams;
- **SaturatedFatContent** : Saturated Fat in the recipe, in grams;
- **CholesterolContent** : Cholesterol in the recipe, in miligrams;
- **SodiumContent** : Sodium in the recipe, in miligrams;
- **CarbohydrateContent** : Carbohydrate in the recipe, in grams;
- **FiberContent** : Fiber in the recipe, in grams;
- **SugarContent** : Sugar in the recipe, in grams;
- **ProteinContent** : Protein in the recipe, in grams;
- **RecipeServings** : Number of servings in the recipe;
- **RecipeYield** : Recipe yield;
- **RecipeInstructions** : Instructions to follow the recipe;
- **URL** : Food.com url;

Each **Review** is an association class between an author and a recipe and has the following attributes:

- **Id** : Unique review ID;
- **RecipeId** : Recipe's ID;
- **UserId** : User's author ID;
- **Rating** : Rating given in the review, between 0 and 5;
- **Review** : The actual review text;
- **DateModified** : Date that the review was made, or last modified;

2.4 Search Tasks

With the finalized dataset, different types of information can be queried. Some interesting retrieval tasks identified are:

- What desserts can I make?
- What can I make in a short time (less than 1h)?
- What can I make with chicken and lime?
- What are the highest-rated recipes?

3 INFORMATION RETRIEVAL

Information Retrieval is the process of retrieving a set of documents over a collection that best suit a given information need. The most common retrieval task is when a user specifies their information need through a query that is matched against all the documents and matching results are ordered by how relevant the system deemed them. The evaluation of the results is done in two systems: the first only performs matching on key fields, the second tunes the weights, using the eDisMax Solr query parser [4], by boosting more relevant search fields. These boosts are chosen using some previous knowledge about the query and the dataset in general, giving a value for a field in accordance to its relevance to the query. These values are a result of a trial and error phase, where multiple combinations of values were used for the same query, with the combination containing the best results the one that was used for analysis. The top 10 results for each query will be analyzed via their Recall, Precision at 10 and Average Precision values. Precision at n measures the precision level at a lower recall level (n < collection size). Because Precision at n is a measure that is easily influenced by the number of relevant documents, Average Precision will be used to complement it. Average Precision is defined as an average of the precision values obtained for the set of top n documents existing each time a new relevant document is retrieved. Since it does not depend on recall, it is considered a more stable measure. Recall, on the other hand, measures the fraction of retrieved relevant documents, i.e., what proportion of all the relevant documents has been retrieved. Calculating recall accurately requires comparing the set of retrieved documents with the entire collection, which is impossible in many cases, therefore it is possible to evaluate recall over a subset of the whole collection, using pooling.[5]

3.1 Collection and Indexing Process

A recipe defines a document in our system. As there will be no other type of documents, a single collection is to be queried upon. To achieve this, we utilized the recommended tool, Solr[6]. Previously, we had described the way we transformed our two data sources into a more database-like structure. However, this structure is not appropriate to use with the selected tool. As such, we had to adapt our previously created dataset to create the single required collection. The changes made were:

- Join of recipes, users, ingredients and keywords - This eliminated the necessity of doing a join in query time, severely reducing the search time of our queries;
- Join with reviews - This way, we can easily access the recipe it is associated with, again eliminating the necessity of a join in query time. To achieve this we needed to concatenate the reviews' text with its author and rating so that we could have a string with all the necessary information;
- Added suffix string "T00:00:00Z" to dates - This is the format of a date in Solr;

As per the indexation of our document, not every field is expected to be searched by a user. Therefore, our first task was to identify which fields would be relevant to the user of our search system. These fields were the recipe's name, description, category, keywords, ingredients, author name, nutritional information, cooking time, time of posting and its reviews. For textual data, such as keywords or ingredients, a default Solr type, *text_en*, was used which already had the needed filters. These filters were:

- Stop word filter, discarding tokens that are on a given stop words list, such as "a" or "an";
- Lower case filter, preventing non-matches due to casing mismatch;
- English Possessive Filter, removing singular possessives (trailing 's) from words;
- Porter Stem Filter, removing the most common morphological and inflexional endings from words in English, like "jumping" to "jump";

At query time, an additional filter is added to the query, the synonym graph filter, which maps single or multi-token synonyms, producing a fully correct graph output. This is useful when searching terms that have synonyms capable of returning better results, such as "oven" and "bake", which ultimately point to the same thing.

We also used a custom type for the recipe's Name and Category fields. Since our goal might eventually be to build a user-friendly interface for our search system, having the ability to search for parts of words is useful for returning partial results. To achieve this we used an EdgeNGram filter, that creates tokens for parts of the word starting at the beginning. Table 1 specifies the added type.

The schema was defined using Solr's Schema (REST) API. The documents were indexed using Solr's Post tool by providing an input JSON array file. Table 2 specifies all fields used in the schema.

Name	Tokenizer	Filters
title	StandardTokenizerFactory	ASCIIFoldingFilterFactory EnglishMinimalStemFilterFactory LowerCaseFilterFactory EdgeNGramFilterFactory

Table 1. Custom field types

Field	Type	Indexed
Name	title	Yes
Date	date	Yes
Description	text_en	Yes
Images	uiString	No
Category	title	Yes
Keywords	text_en	Yes
Ingredients	text_en	Yes
AggregatedRating	float	Yes
ReviewCount	integer	Yes
Calories	float	Yes
Servings	integer	Yes
Yield	text_en	Yes
Instructions	text_en	Yes
URL	uiString	No
AuthorName	text_en	Yes
Review	text_en	Yes

Table 2. Document schema fields

3.2 Retrieval Process

This section will explore some features of Solr while answering some of the proposed queries. For each search task the information need, the corresponding query, the top 10 results, their relevance and performance are presented. As our original datasets have about half a million recipes and a million reviews, we will work with a smaller subset of 100 recipes and their respective reviews.

3.2.1 *Information Need 1 – Person that has chicken in the fridge and wants to search for recipes that use it.*

With this information need, we want to find all the recipes that contain chicken. A perfect system would only retrieve the recipes that contain some part of the chicken (breast, thighs, ...), however, some recipes may contain chicken broth or recommend accompanying with chicken, which might make the query harder. The chosen query is CHICKEN. For search boosting, documents where "chicken" appears in the title will have their score boosted relative to those where the search term appears only in the category or the instructions, and these will have their score boosted relative to those documents where "chicken" appears only in the description, keywords and ingredients. The configuration and results for this query are presented below in Tables 3 and 4, respectively.

	Regular	Boosted
Query	chicken	chicken
Parameters	qf: Name Description Category Keywords Ingredients Instructions	qf: Name^10 Description Category^2 Keywords Ingredients Instructions^2

Table 3. 'chicken' query parameter configuration

	Regular		Boosted	
Rank	Recipe	R	Recipe	R
1	Chicken Cacciatore	Y	Chicken Cacciatore	Y
2	Awesome Garlic Wings	Y	Hot Chicken Sandwiches	Y
3	Tom Kha Gai, Thai Coconut Chicken Soup!	Y	Grilled Chicken Fettuccine	Y
4	Grilled Chicken Fettuccine	Y	Baked Rice and Chicken	Y
5	Feta Topped Broiled Chicken	Y	Feta Topped Broiled Chicken	Y
6	Chicken Cutlet Parmesan With Tomato Sauce	Y	Chicken Tomato Basil Soup	Y
7	Awesome Chicken Noodle Soup	Y	Elegant Stuffed Chicken Breast	Y
8	Thai Satay Chicken Skewers	Y	Awesome Chicken Noodle Soup	Y
9	Hot Chicken Sandwiches	Y	Thai Satay Chicken Skewers	Y
10	Baked Rice and Chicken	Y	Chicken Cutlet Parmesan With Tomato Sauce	Y
Recall	1		1 ($\Delta = 0$)	
P@10	1		1 ($\Delta = 0$)	
AP	1		1 ($\Delta = 0$)	

Table 4. 'chicken' query top 10 results

In total, 30 recipes were retrieved. Both approaches achieved a Recall of 1, precision at 10 (P@10) of 1 and average precision (AP) of 1.

3.2.2 *Information Need 2 – Person that has just bought a new oven and wants to use it, but only has 1 hour to make something.*

For this information need, we want to find all the recipes that use an oven and have a total time (preparation + cooking) of 1 hour or less. The chosen query is OVEN. For search boosting, documents where "oven" appears in the keywords will have their score boosted relative to those where the search term appears only in the instructions, and these will have their score boosted relative to documents where "oven" appears only in the description. We also added a range filter, to limit results to the desired total time. Besides field weights, this query also takes advantage of an entry in our synonyms file that was used in the Boosted system. We define "bake" as a synonym for "oven" since it is a term frequently used in recipes that use ovens. The configuration and results for this query are presented below in Tables 5 and 6, respectively.

	Regular	Boosted
Query	oven	oven
Parameters	qf: Description Keywords Instructions fq: TotalTime:[* TO 3600]	qf: Description Keywords^5 Instructions^2 fq: TotalTime:[* TO 3600]

Table 5. 'oven' query parameter configuration

	Regular		Boosted	
Rank	Recipe	R	Recipe	R
1	Pizzcuit	Y	Jim Dandies	Y
2	Jim Dandies	Y	Breakfast Egg Nests	Y
3	Elegant Stuffed Chicken Breast	Y	Quick and Easy Beer Bread	Y
4	Breakfast Egg Nests	Y	Buffalo Chips (Cookies)	Y
5	Quick and Easy Beer Bread	Y	Mexican Cornbread	Y
6	GAL Muffins	Y	My Favourite Sponge Topping for Fruit	Y
7	Buffalo Chips (Cookies)	Y	Elegant Stuffed Chicken Breast	Y
8	Jim's Sweet Nuts	Y	Witches Fingers (Cookies)	Y
9	It's a Spicy-A Meatball	Y	Jalapeno &amp; Shrimp Poppers	Y
10	Chewy Chocolate Chip Cookies (Ww)	Y	Jim's Sweet Nuts	Y
Recall	0.74		1 ($\Delta = 0.26$)	
P@10	1		1 ($\Delta = 0$)	
AP	1		1 ($\Delta = 0$)	

Table 6. 'oven' query top 10 results

In total, 17 recipes were retrieved by the regular system and 23 were retrieved by the boosted one. This difference in search results is largely due to the synonyms file, which widened the search in the Boosted system. Both approaches achieved a P@10 and AP of 1 while having a Recall of 0.74 in the regular system and 1 in the boosted system.

3.2.3 *Information Need 3 – Get recipes that only need small bowls, not medium or big ones.*

For this information need, we want to find all the recipes that can be made using a small bowl. The query chosen is SMALL BOWL. For search boosting documents where the terms "small" and "bowl" appear together, with a max token distance of 4, and in the instructions will have their score boosted relative to those where "small bowl" appears only in the keywords. The configuration and results for this query are presented below in Tables 7 and 8, respectively.

	Regular	Boosted
Query	small bowl	small bowl
Parameters	qf: Name Description Instructions	qf: Name Description Instructions pf: Instructions^5 ps: 4

Table 7. 'small bowl' query parameter configuration

In total, 12 recipes were retrieved in each system. In the regular system, ranks 1, 4, 8 and 9 were not relevant, while in the boosted system, ranks 5, 8 and 9 were not relevant. The regular system achieved a P@10 of 0.6 and an AP of 0.62, while the boosted system achieved a P@10 of 0.7 and an AP of 0.91. Both approaches achieved a Recall of 1.

	Regular		Boosted	
Rank	Recipe	R	Recipe	R
1	Pear Gingerbread	N	Dill Pickle Soup	Y
2	Potato Soup & Smoked Salmon Relish With Crispy Brie Crouton	Y	Delicious, Easy, Restaurant-Quality Eggplant	Y
3	Dill Pickle Soup	Y	Feta Topped Broiled Chicken	Y
4	Deep-Dish Peach Pie	N	Breakfast Egg Nests	Y
5	Strawberries and Cream Salad Dressing	Y	GAL Muffins	N
6	Delicious, Easy, Restaurant-Quality Eggplant	Y	Grilled Spice-Rubbed Salmon	Y
7	Feta Topped Broiled Chicken	Y	Thai Satay Chicken Skewers	Y
8	GAL Muffins	N	Chewy Chocolate Chip Cookies (Ww)	N
9	Chewy Chocolate Chip Cookies (Ww)	N	Deep-Dish Peach Pie	N
10	Breakfast Egg Nests	Y	Strawberries and Cream Salad Dressing	Y
Recall	1		1 ($\Delta = 0$)	
P@10	0.6		0.7 ($\Delta = 0.1$)	
AP	0.62		0.91 ($\Delta = 0.29$)	

Table 8. 'small bowl' query top 10 results

3.2.4 *Information Need 4 – Recipes that have reviews saying that it was easy to make.*

For this information need, we want to find all the recipes that have a review stating that they can be easily made by a cook. The chosen query is EASY TO MAKE. For search boosting, documents where the terms "easy", "to" and "make" appear together, with a max token distance of 3, will have their score boosted relative to those where those terms simply appear in the reviews. The configuration and results for this query are presented below in Tables 9 and 10, respectively.

In total, 37 recipes were retrieved by each system. Both systems retrieved only 1 irrelevant recipe, which was in rank 2 in the regular system and rank 5 in the boosted system. With only one recipe not being relevant, both systems achieved a P@10 of 0.9, but as the boosted system retrieved the irrelevant one with a lower rank, its average precision was higher than the regular system, 0.93 compared to 0.84. Both approaches achieved a Recall of 1.

	Regular	Boosted
Query	easy to make	easy to make
Parameters	qf: Reviews	qf: Reviews pf: Reviews^5 ps: 3

Table 9. 'easy to make' query parameter configuration

3.2.5 *Information Need 5 – Get Peter J's recipes published between 2008 and 2012.*

For this information need, we want to find all the recipes that were authored by Peter J between the years 2008 and 2015. The pooled subset did not have relevant numbers of recipes by the same author. Since retrieving all recipes authored by someone is a feasible task, we opted to use the **full dataset**. The chosen query is PETER J. For search boosting documents where tokens "peter" and "j" appears in the recipe's author with a max token distance of 1 will have their score boosted. The configuration and results for this query are presented below in Tables 11 and 12, respectively.

In total, both systems returned 793 recipes. Both approaches achieved a Recall of 1, P@10 of 1 and an AP of 1 since they both retrieved the relevant documents at first.

3.3 Evaluation

Overall, the results were very satisfactory. Average precision was always above 0.91 in the boosted system and Recall was always 1, meaning our top results were relevant and all relevant documents were retrieved. The regular system had a MAP (Mean Average Precision [7]) of 0.892 while the boosted system had a MAP of 0.968. With

	Regular		Boosted	
Rank	Recipe	R	Recipe	R
1	Dill Pickle Soup	Y	Dill Pickle Soup	Y
2	Pale Tigers Butter	N	Shipwreck Ground Beef Stew for the Crock Pot	Y
3	Shipwreck Ground Beef Stew for the Crock Pot	Y	Tater Tot Casserole	Y
4	Tater Tot Casserole	Y	Chicken Cacciatore	Y
5	Chicken Cutlet Parmesan With Tomato Sauce	Y	Pale Tigers Butter	N
6	Delicious, Easy, Restaurant-Quality Eggplant	Y	Cottage Cheese-Banana Breakfast Delite	Y
7	Chicken Cacciatore	Y	Chicken Cutlet Parmesan With Tomato Sauce	Y
8	Jalapeno &amp; Shrimp Poppers	Y	Delicious, Easy, Restaurant-Quality Eggplant	Y
9	Alli's Renegade Cocktail Sauce	Y	Jalapeno &amp; Shrimp Poppers	Y
10	Cottage Cheese-Banana Breakfast Delite	Y	Alli's Renegade Cocktail Sauce	Y
Recall	1		1 ($\Delta = 0$)	
P@10	0.9		0.9 ($\Delta = 0$)	
AP	0.84		0.93 ($\Delta = 0.09$)	

Table 10. 'easy to make' query top 10 results

	Regular	Boosted
Query	peter j	peter j
Parameters	qf: AuthorName fq: Date:[2008-01-01T00:00:00Z TO 2012-01-01T00:00:00Z]	qf: AuthorName fq: Date:[2008-01-01T00:00:00Z TO 2012-01-01T00:00:00Z] pf: AuthorName^20 ps: 1

Table 11. 'peter j' query parameter configuration

these results, we can confirm that the boosted system improved metrics when compared to the regular one, meaning our optimizations had the desired effect. Query 3 showed the most significant differences, improving AP by 0.29 and P@10 by 0.10. In the cases where AP and P@10 were the same between the systems, we observed that the boosted system returned subjectively better results first. However, our analytical methods could not objectively measure that improvement.

	Regular		Boosted	
Ranked	Recipe	R	Recipe	R
1	Cheese and Onion Stuffed Sausages	Y	Cheese and Onion Stuffed Sausages	Y
2	Creamy Chicken Pizza Topping	Y	Creamy Chicken Pizza Topping	Y
3	Prosciutto Breakfast Bake	Y	Prosciutto Breakfast Bake	Y
4	Lamb's Fry and Bacon	Y	Lamb's Fry and Bacon	Y
5	Thai Dried Squid (Calamari)	Y	Thai Dried Squid (Calamari)	Y
6	Smoked Oysters and Bacon	Y	Smoked Oysters and Bacon	Y
7	Red Currant Sauce	Y	Red Currant Sauce	Y
8	Healthy Greek Style Chicken Tacos	Y	Healthy Greek Style Chicken Tacos	Y
9	Slow Roast Leg of Lamb	Y	Slow Roast Leg of Lamb	Y
10	Quick Steamed Scallops	Y	Quick Steamed Scallops	Y
Recall	1		1 ($\Delta = 0$)	
P@10	1		1 ($\Delta = 0$)	
AP	1		1 ($\Delta = 0$)	

Table 12. 'peter j' query top 10 results

Concerning the precision-recall curve, queries 1, 2 and 5 had similar curves, with the precision at 1.0. This was observed in both the regular and boosted systems, as can be seen in Figure 10. The other two curves had significant changes between the regular and the boosted systems that can be seen in Figure 11 and Figure 12, for queries 3 and 4, respectively.

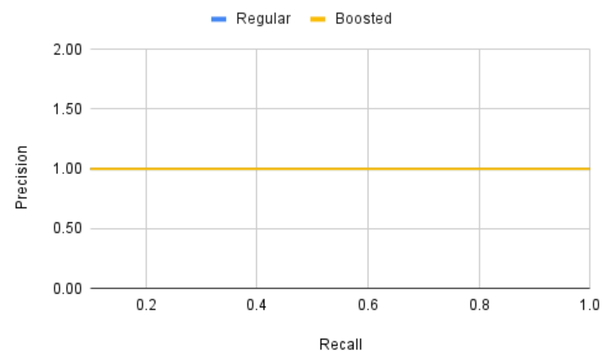


Fig. 10. Precision-Recall Curve for queries 1, 2 and 5

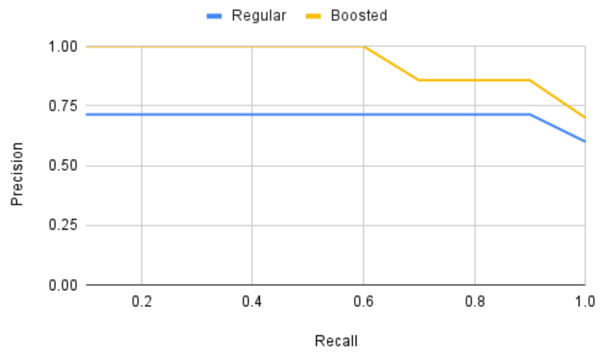


Fig. 11. Precision-Recall Curve for "small bowls"

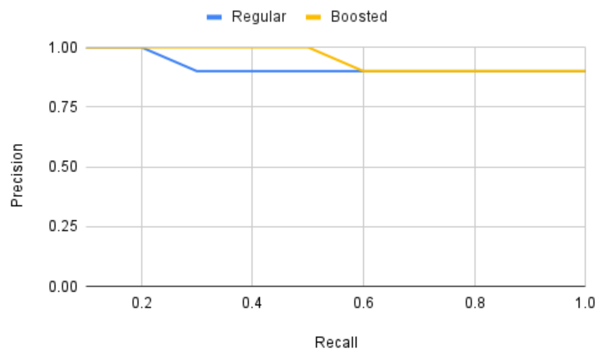


Fig. 12. Precision-Recall Curve for "easy to make"

In both these graphs, we can see that the boosted system outperforms the regular one in terms of precision for low recall values. This means that the first documents returned by the boosted system are more relevant than the ones returned by the regular one. Therefore, our boosts are having the desired effect of pushing more relevant results to the top.

4 SEARCH SYSTEM

4.1 User Interface

For an easier interaction with the user, a user interface was developed using React[8], Node.js[9] and Express[10].

React was used to implement the front-end of both the search page, Annex A, and the individual recipe page, Annex B.

On the main page, the user can search for recipes and sort the results by relevance, date and calories. The relevance sort is made with a re-ranking technique made available by Solr, where the top recipes received from the query are re-ranked with a trained model, to retrieve the most relevant recipes first. This is explained in more detail in Section 4.3. Additionally, the user can filter the recipes by their rating, time, category and ingredients.

For these filters, Solr's faceting feature [11] was used. Searchers are presented with the indexed terms, along with numerical counts of how many matching documents were found for each term, which makes it easy for users to explore search results, narrowing in on exactly the results they are looking for. To achieve this, some changes had to be made to the schema. As both the Category and the Ingredients field were using a tokenizer, faceting would return counts for every indexed token, which was not intended. To only get the counts for the full value, two new fields had to be added to the schema:

- Category_Facet: A field of type string where its contents are copied from the Category field;
- Ingredients_Facet: A multi-valued field of type string where its contents are copied from the Ingredients field;

With these two new fields, faceting could be implemented in a user-friendly way, making the search interface a lot more robust.

On the recipe page, the user can see the full details of the recipe: instructions, ingredients, nutritional information and reviews.

Since Solr does not implement CORS, we were not able to make direct requests from the React app to Solr. Instead, a proxy server was implemented using Node.js and Express, which is responsible for receiving front-end requests and forwarding them to Solr. With this proxy, we can successfully make requests to Solr. The proxy server also reduced the front-end's complexity, since Solr uses a somewhat not user-friendly syntax for its queries.

4.2 Search System Improvements

As demonstrated in the previous section, Section 3, the search system using boosts to make some fields more relevant had better results than the regular system. This is because some fields are more susceptible to search by the regular user, e.g. the recipe's title or its category. On the other hand, some fields like the reviews are less likely to have higher relevancy for the search.

With this in mind, we found it useful to apply the boosts on a more general level, instead of applying them on a query level. Using boosts on a query level, even though the results might be better since each query has its boosts adapted to it, is not feasible to achieve on a large scale search system, as the user has an infinite amount of possible queries, and there is no way to achieve the best possible boost combination for each one of those cases.

To prevent this, the same boosts are applied in every query. These boosts, present in Table 13 were achieved using the general knowledge about recipes, which had a great impact on the ordering of the fields by their relevance. After ordering the fields, the boosts' values were attained using a trial and error methodology.

Field	Boost
Name	5
Category	2
Ingredients	2
Keywords	2
Description	1
Instructions	1
Reviews	0.5
AuthorName	0.2

Table 13. System Boosts

Even though it does not give the optimal results, the system can successfully retrieve the relevant documents at a higher rank than the non-relevant documents, which is sufficient since the system is using a re-ranking model (Section 4.3), where the top documents are re-ranked. As long as the relevant documents are among the top documents, not necessarily being the first ones, the re-ranking model should bring them up to the user. This approach has given the best results when compared to the previous ones, as demonstrated in Section 4.4.

4.3 Learning To Rank

Learning to Rank, LTR [12], is a plugin in Solr that allows for the configuration of machine-learned ranking models in Solr. Re-Ranking allows you to run a simple query for matching documents and then re-rank the top N documents using the scores from a different, more complex query. In information retrieval systems, Learning to Rank is used to re-rank the top N retrieved documents using trained machine learning models. The hope is that such sophisticated models can make more nuanced ranking decisions than standard ranking functions like TF-IDF or BM25.

4.3.1 Configuration.

Solr's LTR plugin is a built-in plugin that comes disabled by default. To enable it, the core configuration file must be updated. Following the LTR Configuration section, from [12], the plugin becomes enabled and it is ready to be used. For some features, Section 4.3.3, the number of items of a multi-valued field was used. To get that count, while maintaining the original dataset, Solr's Update Request Processors [13] were used. Using these processors, the number of values of multi-valued fields would be added to a new field when indexing the dataset, resulting in the same dataset but with more information that is used to improve search results.

4.3.2 Training Data.

To train a machine learning model, training data is needed. For ranking problems, the data must provide information about the relevance of a set of query results. This relevance classification may be based upon human judgment, by having human actors classify a set of results. This can be expensive, especially at scale, but provides good quality data. Alternatively, basing the relevance score on analytical parameters, like user click data or time spent viewing a page, makes it possible to gather more data cheaply. However, this data may be incomplete and noisy [14]. Since the developed system was not deployed in a production environment, using user data was not possible. Therefore, we opted to use manual classification to build our training data-set.

Five queries were chosen as a starting point. We opted for different queries than those used previously as we thought these could provide more meaningful relevance data. These were:

- "Apple pie"
- "Chicken" in the African category
- "Easy bread"
- "Pasta bolognese"
- "Oatmeal"

Next, each query was run on the base Solr search with the boosts specified in table 14, recording the top 100 results. These were then analysed and classified on a scale from 0 to 5 according to the following criteria:

(0) A document that does not match the query.
(1) A document that vaguely matches the query is very incomplete (missing important fields, like instructions) and has no reviews. Or has very negative reviews.
(2) A document that partially matches the query is incomplete and has no reviews. Or a document with negative/mixed reviews.
(3) A document that matches the query semantically is reasonably complete (may miss more than two fields) and has at least one positive review.
(4) A document that perfectly or almost perfectly matches the query semantically is complete or missing just one of the fields and has a good number of positive reviews (5 to 20).
(5) A document that perfectly matches the query semantically is complete (the recipe has a full ingredient list, steps and cooking time/nutritional information) and has a lot of positive reviews (more than 20).

This allowed us to generate 462 data points (one of the queries had only 62 results). In future work, it would be relevant to gather more data to feed the models. A good solution for this is crowd-sourcing this data. A popular example of crowd-sourcing AI training data is Google's reCAPTCHA [15] which uses data collected by its human-verification challenges to improve computer vision algorithms. It is conceivable to add a crowd-sourcing mechanism to our user interface, making use of gamification to incentivize people to contribute. The analytic based methods mentioned previously could also be used to complement this data.

4.3.3 Features.

A ranking model computes the scores used to re-rank documents using features as inputs. A feature is a value, that represents some quantity or quality of the document being scored or of the query for which documents are being scored. Selecting these features is an important part of building a machine learning model. We had limited time to do extensive experimental feature engineering, so we selected features that the literature recommended [16] and based on our knowledge and intuition about the data. Table 14 represents the list of features used to build and train our model. Essentially, there are three categories of features: **field match** features that match the query with a specific field; **count** features that count the number of entries in a given field; **attribute** features, that are related to a given attribute of the document, like its recency. Note that ranking models expect features to be normalized, so these values are not raw but are scaled appropriately.

4.3.4 Model.

Solr's LTR implementation supports three kinds of model: Linear, Multiple Additive Tree and Neural Network. We opted to work with linear models for their simplicity and higher availability of resources. At first, we experimented with the Scikit Learn Python library, using

Feature	Description	Weight
queryMatchName	Score of only matching the query with the Name field	-0.61264533
queryMatchDescription	Score of only matching the query with the Description field	-0.016012382
queryMatchCategory	Score of only matching the query with the Category field	0.40750793
queryMatchIngredients	Score of only matching the query with the Ingredients field	0.10536964
queryMatchKeywords	Score of only matching the query with the Keywords field	0.23444816
queryMatchInstructions	Score of only matching the query with the Instructions field	-0.030611534
queryMatchReviews	Score of only matching the query with the Reviews field	0.29199001
queryMatchAuthorName	Score of only matching the query with the AuthorName field	-0.88715649
recency	Measure of how recent the recipe is	-0.43097624
rating	Recipe's Rating	0.84916848
reviewCount	Number of reviews	1.3468372
keywordCount	Number of keywords	1.6347141
ingredientCount	Number of ingredients	1.6268826
instructionCount	Number of instructions	0.045574259
imageCount	Number of images	-5.1624022
originalScore	The original score returned by Solr's boosted ranking	0.29625478

Table 14. LTR Features and Model Weights

the LinearSVR algorithm. [17] However, the initial results we got were not very impressive, possibly due to the algorithm being based on a point-wise approach, analysing only one document at a time and trying to infer what its relevance is based on its features. We then turned to the Ranking SVM [18] algorithm, which uses a pairwise approach, looking at document pairs and predicting which one will be the most relevant to the query, given its features. The final model for our LTR system was calculated using this algorithm. [19] The weights calculated by the algorithm can be seen in Table 14.

4.4 Evaluation

For evaluation measures, in addition to Average Precision and Recall, it was necessary to find a metric that took into account *how* relevant a result was, not just *if* it was relevant. After some research, we opted for the Expected Reciprocal Rank Evaluation Metric (ERR). Alternatives like Normalized Discounted Cumulative Gain (NDCG) could also provide this information, however, ERR is based on the cascade user model. [20]

The cascade model assumes a user scans through ranked search results in order, and for each document, evaluates whether the document satisfies the query. If it does, the user stops the search. This assumes that a single document can satisfy a search, which falls short for some applications but is a reasonably good assumption for a search engine like ours - people will tend to choose only one recipe to make. To make a cascade model complete, we need to model how likely it is that a given document will satisfy a given user query. This is done by assigning each query-document pair a grade that is later translated into probabilities of the document satisfying the search. [21]

Expected reciprocal rank is the prediction of the reciprocal of the position of a result at which a user stops. In other words, it is the prediction of a value $1/s$, where s is the position where a user stops. It is calculated using the following formula, where K is the total documents retrieved and $p(q, d_k)$ is the probability of a document k satisfying the search query q:

$$ERR = \sum_{k=1}^{K} \frac{1}{k} \cdot p(q, d_k) \cdot \prod_{i=1}^{k-1} (1 - p(q, d_i))$$

To evaluate the improvements of the LTR model, we evaluated a new set of testing queries. These are queries on which the model was not trained on, therefore giving us a better idea of the performance improvements that LTR will give us. Once the queries were selected, we analysed the results obtained using the two systems: the Regular system, with boosts; and the Enhanced system, with boosts and re-ranking the top 100 results. For each query, the top 10 results returned by each system were classified, to be able to calculate the desired evaluation metrics. Table 15 shows the test queries and the results obtained by each system, ranked according to the criteria in Section 4.3.2. ERR, Average Precision and Recall values were calculated for each query.

As we can see in Table 15, the Enhanced system presented significant improvements over the Base system, concerning ERR. The average ERR of the Base system is 0.54 and the Boosted system is 0.84. This means that we expect that, on average, a user will be satisfied by the second result of the search engine (1/0.54=1.85), when using the base system) and the first result (1/0.84=1.19), with the enhanced system - a great result given the size of our experiment. The Boosted system improves this metric, even with the very small amount of data that we trained it on (462 data points in a universe of half a million recipes). This shows that LTR can be a very powerful addition to a search system. To further improve results it would be necessary to add more training data and also explore more models and features. LTR also provides extra functionality that was not explored in this project but can improve result quality further, such as the possibility of adapting results to a given user profile, by providing user information to the model. [22]

Query	System	Ranks										AP	Recall	ERR
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**			
pumpkin pie	**Regular**	3	4	3	2	2	2	2	2	3	2	1.0	1.0	0.46
	Enhanced	4	3	3	3	4	5	5	3	3	5	1.0	1.0	0.62 ($\Delta = 0.16$)
vegetable curry	**Regular**	4	2	3	3	4	3	4	2	3	1	1.0	1.0	0.59
	Enhanced	4	3	3	4	3	3	3	3	3	4	1.0	1.0	0.62 ($\Delta = 0.03$)
mojito	**Regular**	4	4	5	2	2	1	2	2	1	2	1.0	1.0	0.69
	Enhanced	5	4	5	3	4	4	4	4	2	5	1.0	1.0	0.98 ($\Delta = 0.29$)
noodles	**Regular**	4	2	2	2	2	3	4	5	3	2	1.0	1.0	0.58
	Enhanced	5	3	4	3	5	4	3	4	4	4	1.0	1.0	0.98 ($\Delta = 0.40$)
beef stew	**Regular**	3	2	2	3	4	1	2	2	2	2	1.0	1.0	0.37
	Enhanced	5	3	5	3	4	3	3	3	5	4	1.0	1.0	0.98 ($\Delta = 0.61$)

Table 15. Results comparing System with Boosts and System with Boosts and LR

5 CONCLUSION

This paper addressed three distinct milestones in the construction of a system capable of answering complex queries about food recipes data. The retrieved data was cleaned, refined and conceptually described to represent the main classes and associations between them. An exploratory analysis showed patterns associated with the source of the data but also with the data itself. From this point, it was possible to identify information retrieval tasks that were later answered, using Solr and our schema. Results were positive for what would be expected by a user and were consistent with existing literature. Then, the system was reanalyzed from a new perspective. using machine learning techniques, to develop a fully-fledged search system. With the use of a plugin for Solr, it was possible to significantly improve the results of the retrieval tasks. Additionally, a user interface was designed. Further improvements to the machine learning model, training data, and to the user interface could be made in posterior iterations of this project, refining the results already obtained.

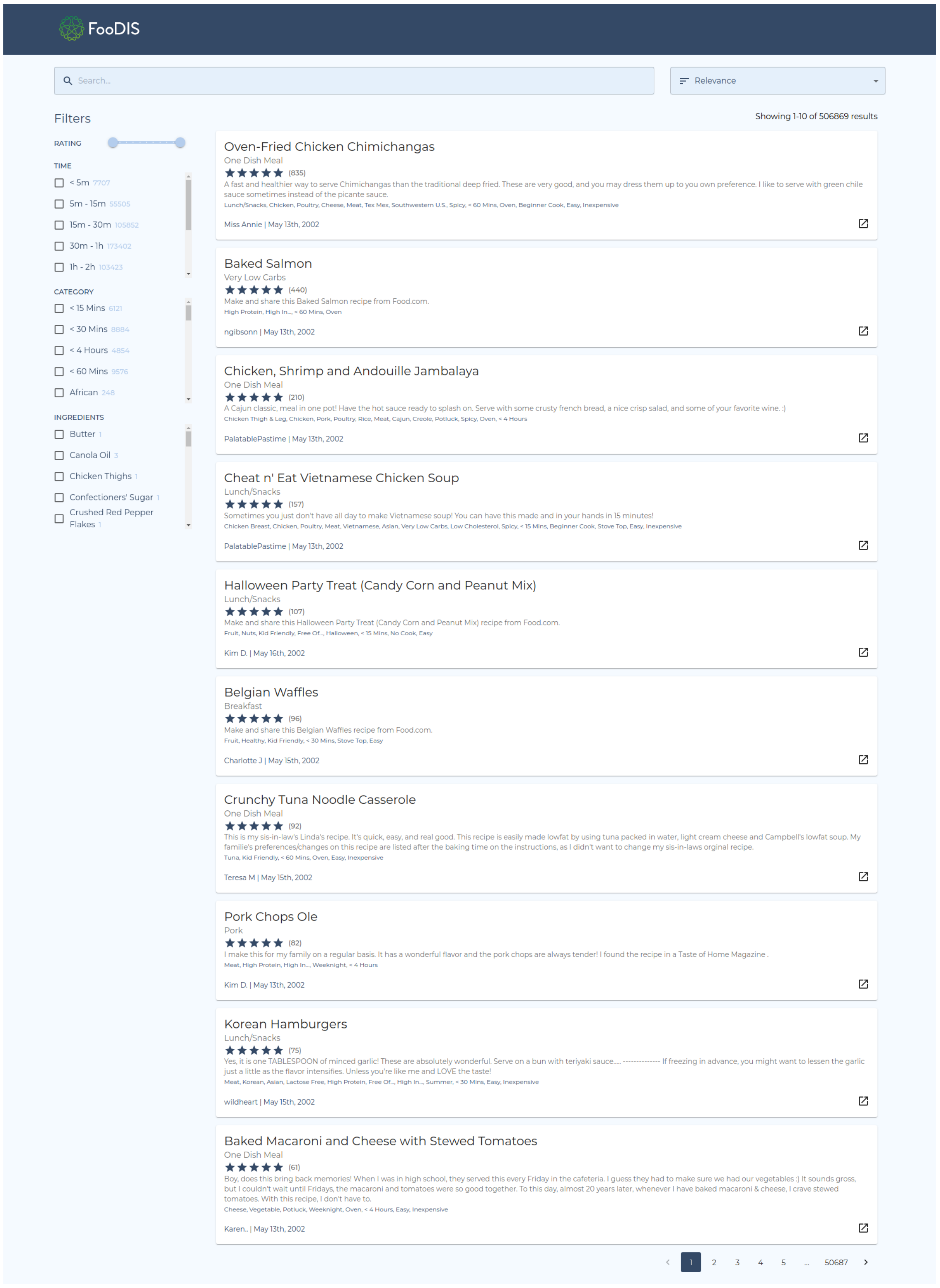
6 CITATIONS AND BIBLIOGRAPHIES

The following section serves to reference our citations and to list the materials used in this project.

REFERENCES

[1] M. E. Shoup. (2021, Mar.) Home baking continues in 2021 giving rise to comfort and wellness trends. [Online]. Available: https://www.foodnavigator-usa.com/Article/2021/03/22/Home-baking-continues-in-2021-giving-rise-to-comfort-and-wellness-trends
[2] irkaal. (2020) Food.com - recipes and reviews. [Online]. Available: https://www.kaggle.com/irkaal/foodcom-recipes-and-reviews
[3] Discovery. (2021) Food.com. [Online]. Available: https://www.food.com/?ref=nav
[4] A. S. Foundation. (2021, Nov.) The extended dismax (edismax) query parser. [Online]. Available: https://solr.apache.org/guide/8_10/the-extended-dismax-query-parser.html
[5] S. Nunes. (2021, Nov.) Search results evaluation. [Online]. Available: https://git.fe.up.pt/pri/tutorials/-/tree/main/06-evaluation
[6] T. A. S. Foundation. (2021) Solr. [Online]. Available: https://solr.apache.org/
[7] Wikipedia. (2021, Dec.) Evaluation measures (information retrieval). [Online]. Available: https://en.wikipedia.org/wiki/Evaluation_measures_(information_retrieval)#Mean_average_precision
[8] I. Meta Platforms. (2022, Nov.) React. [Online]. Available: https://reactjs.org/
[9] O. Foundation. (2022, Nov.) Node.js. [Online]. Available: https://nodejs.org/en/
[10] ——. (2017, Nov.) Express.js. [Online]. Available: https://expressjs.com/
[11] A. S. Foundation. (2021, Sep.) Faceting. [Online]. Available: https://solr.apache.org/guide/8_10/faceting.html
[12] ——. (2021, Nov.) Ltr. [Online]. Available: https://solr.apache.org/guide/8_11/learning-to-rank.html
[13] ——. (2021, Sep.) Update request processors. [Online]. Available: https://solr.apache.org/guide/8_10/update-request-processors.html
[14] ——. (2017, Jul.) Ltr example. [Online]. Available: https://github.com/apache/lucene-solr/tree/releases/lucene-solr/8.10.0/solr/contrib/ltr/example
[15] Google. recaptcha: Easy on humans, hard on bots. [Online]. Available: https://www.google.com/recaptcha/intro/invisible.html?ref=producthunt#creation-of-value
[16] T. Qin and T. Liu, "Introducing LETOR 4.0 datasets," *CoRR*, vol. abs/1306.2597, 2013. [Online]. Available: http://arxiv.org/abs/1306.2597
[17] airalcorn2. (2020, Aug.) From zero to learning to rank in apache solr. [Online]. Available: https://github.com/airalcorn2/Solr-LTR
[18] D. o. C. S. Thorsten Joachims, Cornell University. (2009, Mar.) Support vector machine for ranking. [Online]. Available: https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html
[19] M. Wambua. (2018, Aug.) Build yourself a mini search engine. [Online]. Available: https://www.cs.toronto.edu/~muuo/blog/build-yourself-a-mini-search-engine/
[20] C. Burges, 01 2010. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/MSR-TR-2010-82.pdf
[21] Breckbaldwin. (2010, Mar.) Chapelle, metzler, zhang, grinspan (2009) expected reciprocal rank for graded relevance. [Online]. Available: https://lingpipe-blog.com/2010/03/09/chapelle-metzler-zhang-grinspan-2009-expected-reciprocal-rank-for-graded-relevance/
[22] M. Nilsson and D. Ceccarelli. (2015, Nov.) Learning to rank in solr. [Online]. Available: https://www.youtube.com/watch?v=M7BKwJoh96s

A SEARCH PAGE



Filters

RATING

TIME

- < 5m 7707
- 5m - 15m 55505
- 15m - 30m 105852
- 30m - 1h 173402
- 1h - 2h 103423

CATEGORY

- < 15 Mins 6121
- < 30 Mins 8884
- < 4 Hours 4854
- < 60 Mins 9576
- African 248

INGREDIENTS

- Butter 1
- Canola Oil 3
- Chicken Thighs 1
- Confectioners' Sugar 1
- Crushed Red Pepper Flakes 1

Showing 1-10 of 506869 results

Oven-Fried Chicken Chimichangas
One Dish Meal
★★★★★ (835)
A fast and healthier way to serve Chimichangas than the traditional deep fried. These are very good, and you may dress them up to you own preference. I like to serve with green chile sauce sometimes instead of the picante sauce.
Lunch/Snacks, Chicken, Poultry, Cheese, Meat, Tex Mex, Southwestern U.S., Spicy, < 60 Mins, Oven, Beginner Cook, Easy, Inexpensive
Miss Annie | May 13th, 2002

Baked Salmon
Very Low Carbs
★★★★★ (440)
Make and share this Baked Salmon recipe from Food.com.
High Protein, High In..., < 60 Mins, Oven
ngibsonn | May 13th, 2002

Chicken, Shrimp and Andouille Jambalaya
One Dish Meal
★★★★★ (210)
A Cajun classic, meal in one pot! Have the hot sauce ready to splash on. Serve with some crusty french bread, a nice crisp salad, and some of your favorite wine. :)
Chicken Thigh & Leg, Chicken, Pork, Poultry, Rice, Meat, Cajun, Creole, Potluck, Spicy, Oven, < 4 Hours
PalatablePastime | May 13th, 2002

Cheat n' Eat Vietnamese Chicken Soup
Lunch/Snacks
★★★★★ (157)
Sometimes you just don't have all day to make Vietnamese soup! You can have this made and in your hands in 15 minutes!
Chicken Breast, Chicken, Poultry, Meat, Vietnamese, Asian, Very Low Carbs, Low Cholesterol, Spicy, < 15 Mins, Beginner Cook, Stove Top, Easy, Inexpensive
PalatablePastime | May 13th, 2002

Halloween Party Treat (Candy Corn and Peanut Mix)
Lunch/Snacks
★★★★★ (107)
Make and share this Halloween Party Treat (Candy Corn and Peanut Mix) recipe from Food.com.
Fruit, Nuts, Kid Friendly, Free Of..., Halloween, < 15 Mins, No Cook, Easy
Kim D. | May 16th, 2002

Belgian Waffles
Breakfast
★★★★★ (96)
Make and share this Belgian Waffles recipe from Food.com.
Fruit, Healthy, Kid Friendly, < 30 Mins, Stove Top, Easy
Charlotte J | May 15th, 2002

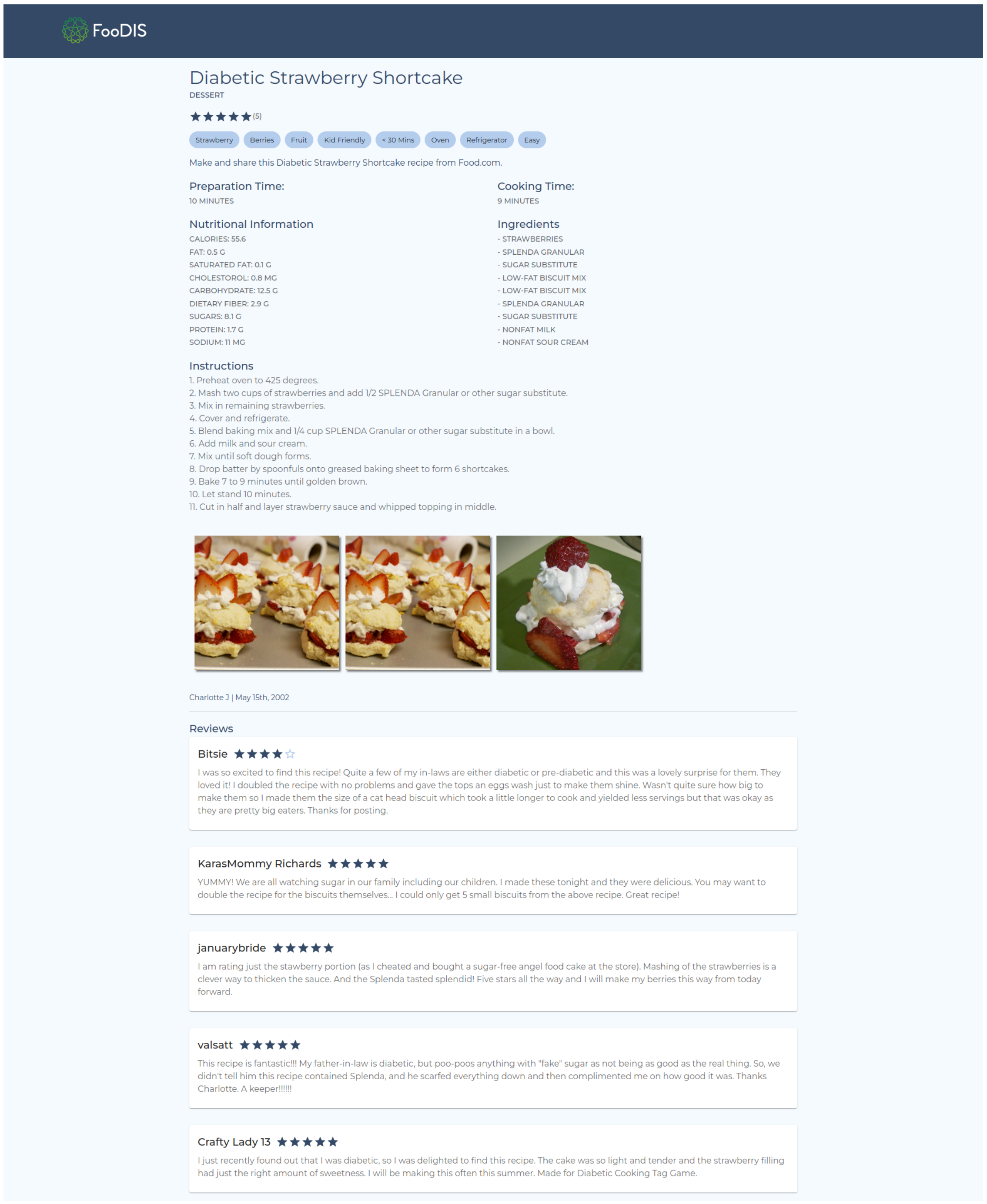
Crunchy Tuna Noodle Casserole
One Dish Meal
★★★★★ (92)
This is my sis-in-law's Linda's recipe. It's quick, easy, and real good. This recipe is easily made lowfat by using tuna packed in water, light cream cheese and Campbell's lowfat soup. My familie's preferences/changes on this recipe are listed after the baking time on the instructions, as I didn't want to change my sis-in-laws orginal recipe.
Tuna, Kid Friendly, < 60 Mins, Oven, Easy, Inexpensive
Teresa M | May 15th, 2002

Pork Chops Ole
Pork
★★★★★ (82)
I make this for my family on a regular basis. It has a wonderful flavor and the pork chops are always tender! I found the recipe in a Taste of Home Magazine .
Meat, High Protein, High In..., Weeknight, < 4 Hours
Kim D. | May 13th, 2002

Korean Hamburgers
Lunch/Snacks
★★★★★ (75)
Yes, it is one TABLESPOON of minced garlic! These are absolutely wonderful. Serve on a bun with teriyaki sauce..... ------------- If freezing in advance, you might want to lessen the garlic just a little as the flavor intensifies. Unless you're like me and LOVE the taste!
Meat, Korean, Asian, Lactose Free, High Protein, Free Of..., High In..., Summer, < 30 Mins, Easy, Inexpensive
wildheart | May 15th, 2002

Baked Macaroni and Cheese with Stewed Tomatoes
One Dish Meal
★★★★★ (61)
Boy, does this bring back memories! When I was in high school, they served this every Friday in the cafeteria. I guess they had to make sure we had our vegetables :) It sounds gross, but I couldn't wait until Fridays, the macaroni and tomatoes were so good together. To this day, almost 20 years later, whenever I have baked macaroni & cheese, I crave stewed tomatoes. With this recipe, I don't have to.
Cheese, Vegetable, Potluck, Weeknight, Oven, < 4 Hours, Easy, Inexpensive
Karen.. | May 13th, 2002

B RECIPE PAGE


FooDIS

Diabetic Strawberry Shortcake

DESSERT

★★★★★ (5)

Strawberry | Berries | Fruit | Kid Friendly | < 30 Mins | Oven | Refrigerator | Easy

Make and share this Diabetic Strawberry Shortcake recipe from Food.com.

Preparation Time:
10 MINUTES

Cooking Time:
9 MINUTES

Nutritional Information
CALORIES: 55.6
FAT: 0.5 G
SATURATED FAT: 0.1 G
CHOLESTOROL: 0.8 MG
CARBOHYDRATE: 12.5 G
DIETARY FIBER: 2.9 G
SUGARS: 8.1 G
PROTEIN: 1.7 G
SODIUM: 11 MG

Ingredients
- STRAWBERRIES
- SPLENDA GRANULAR
- SUGAR SUBSTITUTE
- LOW-FAT BISCUIT MIX
- LOW-FAT BISCUIT MIX
- SPLENDA GRANULAR
- SUGAR SUBSTITUTE
- NONFAT MILK
- NONFAT SOUR CREAM

Instructions
1. Preheat oven to 425 degrees.
2. Mash two cups of strawberries and add 1/2 SPLENDA Granular or other sugar substitute.
3. Mix in remaining strawberries.
4. Cover and refrigerate.
5. Blend baking mix and 1/4 cup SPLENDA Granular or other sugar substitute in a bowl.
6. Add milk and sour cream.
7. Mix until soft dough forms.
8. Drop batter by spoonfuls onto greased baking sheet to form 6 shortcakes.
9. Bake 7 to 9 minutes until golden brown.
10. Let stand 10 minutes.
11. Cut in half and layer strawberry sauce and whipped topping in middle.

Charlotte J | May 15th, 2002

Reviews

Bitsie ★★★★☆
I was so excited to find this recipe! Quite a few of my in-laws are either diabetic or pre-diabetic and this was a lovely surprise for them. They loved it! I doubled the recipe with no problems and gave the tops an eggs wash just to make them shine. Wasn't quite sure how big to make them so I made them the size of a cat head biscuit which took a little longer to cook and yielded less servings but that was okay as they are pretty big eaters. Thanks for posting.

KarasMommy Richards ★★★★★
YUMMY! We are all watching sugar in our family including our children. I made these tonight and they were delicious. You may want to double the recipe for the biscuits themselves... I could only get 5 small biscuits from the above recipe. Great recipe!

januarybride ★★★★★
I am rating just the stawberry portion (as I cheated and bought a sugar-free angel food cake at the store). Mashing of the strawberries is a clever way to thicken the sauce. And the Splenda tasted splendid! Five stars all the way and I will make my berries this way from today forward.

valsatt ★★★★★
This recipe is fantastic!!! My father-in-law is diabetic, but poo-poos anything with "fake" sugar as not being as good as the real thing. So, we didn't tell him this recipe contained Splenda, and he scarfed everything down and then complimented me on how good it was. Thanks Charlotte. A keeper!!!!!!

Crafty Lady 13 ★★★★★
I just recently found out that I was diabetic, so I was delighted to find this recipe. The cake was so light and tender and the strawberry filling had just the right amount of sweetness. I will be making this often this summer. Made for Diabetic Cooking Tag Game.