# Sous, an Ingredient Substitution and Task Scheduling Tool

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

**William Tonks**

**Spring, 2020.**

Nathan Brunelle, Department of Computer Science

David Evans, Department of Computer Science

# Sous, an Ingredient Substitution and Task Scheduling Tool

A Proposal for a new Technical Tool for Home Chefs

William Tonks
University of Virginia
wrt6af@virginia.edu

## ABSTRACT

Even after finding the right recipe, new home cooks face several major hurdles when it comes to undertaking more complicated or time-intensive meals. Having to make recipe alterations due to missing ingredients can be daunting, especially for chefs experimenting with new flavors or techniques. Secondly, time management within the kitchen is an issue on multiple fronts, whether in having to make rapid decisions about what to do next when preparing multiple dishes at once or having to structure a day around a longer task such as barbecuing. This project proposal details a new web tool to be built in Django called Sous which seeks to address these two issues by providing both automatic adjustment to recipes for missing ingredients with common substitutes and scaling ingredient quantities and by providing a task scheduler that structures a cook's time in the kitchen. The initial recipes listed on the site will be populated by scraped recipes off of the website Epicurious using a pre-built web scraping tool, and the initial substitutions listed will be pulled from an open-use spreadsheet of common substitutions put together by amateur enthusiasts on Allrecipes. This report introduces the current state of the art in cooking technology solutions, details how Sous would meet currently unmet user needs, system design for the tool, inherent challenges in implementation for the tool, design decisions made in developing a prototype implementation of the substitution and scaling service, and opportunities for future work.

## INTRODUCTION

When a home chef decides to start making a meal, they have a lot of factors they need to consider both before and while they are cooking. They have to first decide on a recipe that meets their dietary and family needs, determine if they have the ingredients necessary to make it, figure out what order to prepare different main and side dishes, and then get everything on the table at the appropriate temperature and at the right time for a meal. All of these factors combined can make the prospect of preparing dinner a daunting task even for a relatively seasoned home cook. Some cooks might try to solve this problem by planning out a week's meal in advance or making more frequent visits to the grocery store, but these solutions require additional time out of the user's day. Americans are eating more at home than ever according to a survey conducted by ReportLinker; this change is driven by the rising costs of dining out and a desire to eat healthy [5]. However, home cooks describe the cooking process as frequently too time intensive, with 77 percent of cooks wanting to spend less than an hour preparing a meal. Because of this desire to minimize time spent planning for meals and the hassle of finding recipes with ingredients on hand, cooks can find themselves reticent to explore and experiment with new recipes. Sous seeks to address these issues and make experimentation in the kitchen more accessible to less experienced cooks. The key issues Sous will address are providing support to cooks who need to alter parts of a recipe because of allergens or missing ingredients, provide easy recipe scaling to alter portion number, and aid in time management to help cooks maximize results while minimizing time spent in the kitchen.

## BACKGROUND AND RELATED WORK

### 1 Existing Commercial Solutions

Addressing the needs of home chefs through technology solutions is certainly not a new concept. There are already robust online forums and publications dedicated to providing access to hundreds of thousands of recipes such as Allrecipes and Epicurious. In conjunction with these sites are numerous blogs, Youtube channels, and standalone sites that provide cooking instructional content about techniques and cuisines. However, the majority of these sites are lacking in the support they provide to chefs in addressing ingredient substitution and time management. *Yummly,* an American mobile app and website, provides recipe recommendations based on either learned user personal preferences (such as allergies, diets, and experience level) or listed current pantry ingredients coupled with in-built food delivery service. Another similar tool is called *SuperCook*, which sorts its recipe search over a scraped recipe database by prioritizing recipes that utilize ingredients in the users pantry. Both of these tools allow for users to find recipes that utilize ingredients they already have, but neither enable users to adapt a desired recipe with available ingredients. One iPhone app, called *Substitutions,* provides users with a list of common substitutions; however, the list is not searchable and does not dynamically apply to recipes, which means users have to manually find ingredients. Solutions that exist for cooking scheduling problems are mostly just modified mobile timer apps that allow for users to set several timers at once, but none attempt to intelligently pipeline tasks to save time.

## 2 Techniques for Ingredient Substitution

Most commercial cooking apps perform very limited adjustment or analysis of recipes beyond keyword recognition for search because moving beyond this level requires either detailed annotations about a specific recipe or ingredient (which is typically infeasible considering these sites draw their information from user generated content) or requires advanced natural language processing techniques to produce this robust and structured data. While building out their recipe recommendation algorithm based on building relational networks of ingredients and substitutions, Teng et. al [7] found the problem of accurately identifying frequently used ingredients to be a challenging problem because of variation between ingredient lines and descriptions. They note that their approach of using regular expressions to remove weight quantifiers and filler text had limitations with highly specific ingredients, misspelled ingredients, or included Brand names (such as the difference between “Bush’s Baked Beans” and “baked beans”) [7]. Pesce [1] also originally attempted a regular expression based approach when attempting to identify recipes in news stories and quickly identified similar issues: either the written text needed to be stringently formatted to be parsable by string formatting or regular expressions or they needed to utilize natural language processing. Using Python’s NLTK and an entropy classifier for training a model, Pesce noted better success in identifying full text recipes but that moving beyond simple classification into interpretation required a very advanced understanding of the surrounding grammar for ingredients. For instance, “fry” could be either a direction or an ingredient dependent on the surrounding grammar.

Ingredient substitution is an archetypical problem for researchers attempting to further contextual language analysis, as ingredients have many uses and their properties within a particular recipe are context dependent. Substitution of one ingredient for another may be usually appropriate, but there may be recipes where that substitution is unacceptable. For instance, honey is commonly used as a substitute for white sugar, however, if a chef is told to cream together butter and sugar for a custard recipe, they will be sorely disappointed if they attempt using honey[3]. Identifying the most accurate substitution for a given recipe requires robust understanding of the properties of an ingredient and which of these properties are most important for a given recipe. Gaillard et. al [4] improved substitution accuracy by utilizing a technique called formal concept analysis where formal concepts are represented by pairs of *I* (a set of properties) and *E* (a set of objects that share those properties). These concepts can be structured into a *concept lattice*, which provides a hierarchy of properties and allows for identifying similar ingredients based on their use across recipes to develop explicit substitution rules. This strategy makes intuitive sense, for instance both lemon juice and vinegar are utilized as acidic and emulsifying components in recipes and structuring ingredients into a concept lattice would show the importance of these shared properties. Shirai et al. [3] criticize Gaillard’s approach as non-scalable as their concept lattice requires detailed notes about the relative properties about ingredients and recommends a Knowledge-Graph based approach with a substitutability heuristic (DIISH) driven by semantic information about frequency in similar recipes and co-occurrence of other ingredients across recipes containing target ingredients. This graph structured model outperformed explicit rule based substitution such as the TAABLE method used by Gaillard et. al [3] [4]. Another group that attempted to solve this substitution problem was Pan et. al which utilized Python’s word2vec, which is a Skip-gram model for NLP, in order to guess the surrounding context (recipe) for a central ingredient [2]. Their experiment demonstrated how easy it is for a processing algorithm to misidentify good ingredient substitutes if incorrect criteria is chosen as a heuristic. Their intuition was to assign substitutes based on ingredients appearing frequently together in recipes. However, this approach led to ingredients that are frequently used together (such as olive oil and garlic or carrots and squid) being considered as good substitutes, which is clearly incorrect in terms of their properties as ingredients.

### 3 Scaling Ingredient Quantities

Substitutions also have an issue in identifying the amount of a substituted ingredient to replace the original. While there are some substitutions with commonly understood ratios, there are issues wherever substitutes have different flavor and textural properties from the original ingredient. Gaillard et al. attempted to solve this problem by creating a series of heuristics for a recipe (such as weight, flavor balance, volume, alcohol content), calculating the values for these across the original recipe, and then attempting to conserve these values (calculated with Manhattan distance) in the new recipe by adjusting the quantity of the new ingredients [4]. They make special mention that if every ingredient variable is continuous, then this optimization problem poses an NP-Hard problem.

Even scaling non-substituted recipes to increase/decrease servings also poses some unique challenges. A naive approach to this problem simply scales all ingredients used in a recipe proportionally to the adjustment of servings; however, while this approach generally works, it has limitations. For instance, there is usually no need to scale proportionally the amount of fat used for searing if searing greater amounts of meat or there are many spices and aromatics that if proportionally increased will quickly overpower all other flavor components in a recipe.

## SYSTEM DESIGN

### 1 Sous Web Service

Sous is proposed as a web service rather than as a mobile application primarily to allow users without mobile devices to utilize its services; however, it would be built with both desktop and mobile flavors in mind. Sous would be built using the Django development framework for several reasons. Primarily, Django is Python-based and Python has several packages that could be incredibly useful to a more robust Sous service that utilized language processing for analyzing the context of ingredients, namely the Natural Language Toolkit [6]. Other benefits of utilizing Django are that its templates for forms and forums which would simplify the front-end development significantly, and Django's Object-Relational Management tool, which allows for interacting with SQL databases in a Pythonic way rather than having to specifically write SQL queries for manipulating information in the database [6]. Django also offers extensive security benefits, such as automatically disguising source code and having easy to deploy user authentication through third parties like Gmail.

The Sous service would include a central feed of recipes after a user searches for recipes including certain keywords, forms for submitting recommendations for recipe substitutions and assigning times to recipe steps, and forums for discussing recipe content.

### 2 Substitution and Recipe Scaler

*2.1 Data Collection and Preprocessing*

A dataset of recipes collected from Epicurious was found available for free public use and was adapted for a prototype implementation of Sous' substitution and recipe scaling service. The dataset contained over twenty five thousand non-duplicate user generated recipes formatted into a JSON file with associated ingredients, recipe steps, and title for each key. The list of unidirectional substitutions (87 in total) was collected from an article posted on Allrecipes.com, manually entered into a CSV file, and then read into the program. The information from each of these files was used to create python dictionaries of object hierarchies (recipe objects with ingredient child-objects and a substitution list for a given ingredient with component child-objects).

Parsing the recipe and substitution datasets posed a particular challenge, as the content was user generated and had inconsistent formatting and grammar across the board. For instance quantities could be listed with ranges of acceptable amounts ("4-6 tablespoons", be non-standard units (such as "3 lemons" or " 2 cubes"), and many other variations that had to be accounted for in preprocessing. Interpreting fractional amounts also took additional logic, but was simplified because all quantities had formatting similarities that were exploitable. Lines that contained multiple measurements were broken up into components when recognizable. Substitutions were considered to be unidirectional, meaning that even if an ingredient like applesauce was listed as a substitution for vegetable oil, vegetable oil would not be considered a substitute for applesauce.

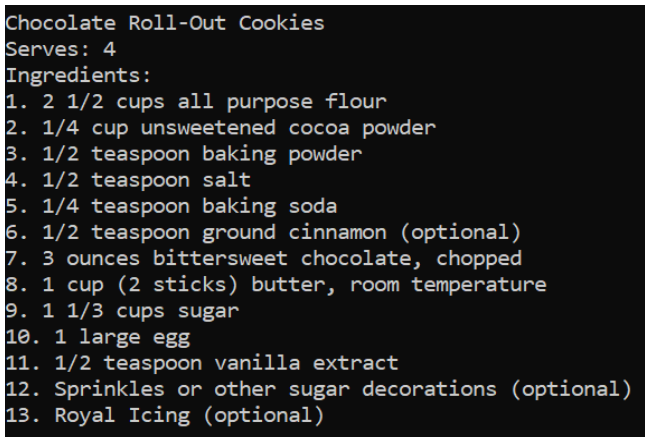
When storing quantities for both substitution ratios and ingredients in recipes, quantities would be converted from standard American volumes/masses to metric values for ease of calculation for substitutions and recipe scaling.

*2.2 Substitution and Scaling User Flow*

Users are able to select a recipe from the recipe database by entering its exact title. In an optimal Sous implementation, this would be replaced with a keyword search algorithm that prioritizes recipes based on similarity or filter criteria. After selecting a recipe, Sous presents the user with all listed ingredients from that recipe along with the number of servings. Due to limitations from the dataset found, the serving size was set to 4 for all entries to demonstrate the scaling aspects of the tool. Users are then able to select ingredients they do not have or indicate if they want to scale the overall servings produced. If an ingredient is not available, Sous searches its internal list of explicit substitution rules and determines if it has a substitution exactly matching or close to what the user entered. If Sous has matching substitutions, it lists them and allows the user to pick which substitution they would prefer. After selecting a substitution, the recipe is updated to reflect the ingredients and quantities. If the user indicates they would like to scale the recipe (which can be done before or after replacing recipe components with substitutes), Sous takes in their desired serving amounts and scales the entire recipe using the naive proportional approach described earlier. No dataset was found that described scaling rules based on ingredient properties, which left the naive approach.

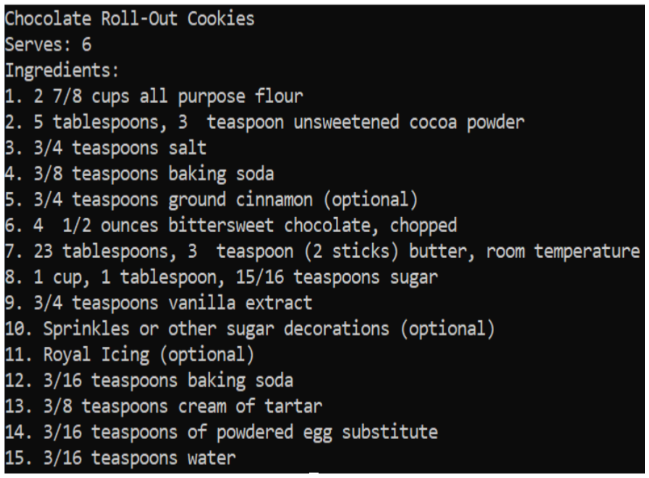
To better meet the needs of real life cooks, after quantities had been scaled or adjusted based on substitution ratios, they were formatted into American standard volumes/masses to be usable in the kitchen by the average home chef. Converting decimal metric measurements into fractional values posed an interesting challenge in implementation, as the prototype needed to deliver measurements that were actually usable in a kitchen setting. For instance, informing a cook that they need to utilize 2 33/ 71 tablespoons of butter in a kitchen would be fairly unhelpful and require the user to approximate this value. Instead, this prototype does this approximation for the user and delivers approximate fractions for new ingredient quantities.

The code and datasets used for this prototype are found at https://github.com/williamtonks/Sous . Below is a recipe before and after it had substitutions and scaling applied. There are some obvious errors in this output, such as how 23 tablespoons of butter is called two sticks; however, removing these inconsistencies would require extensive language processing capabilities.



```
Chocolate Roll-Out Cookies
Serves: 4
Ingredients:
1. 2 1/2 cups all purpose flour
2. 1/4 cup unsweetened cocoa powder
3. 1/2 teaspoon baking powder
4. 1/2 teaspoon salt
5. 1/4 teaspoon baking soda
6. 1/2 teaspoon ground cinnamon (optional)
7. 3 ounces bittersweet chocolate, chopped
8. 1 cup (2 sticks) butter, room temperature
9. 1 1/3 cups sugar
10. 1 large egg
11. 1/2 teaspoon vanilla extract
12. Sprinkles or other sugar decorations (optional)
13. Royal Icing (optional)
```

Figure 1: **Original recipe for Chocolate Roll-Out Cookies**



```
Chocolate Roll-Out Cookies
Serves: 6
Ingredients:
1. 2 7/8 cups all purpose flour
2. 5 tablespoons, 3  teaspoon unsweetened cocoa powder
3. 3/4 teaspoons salt
4. 3/8 teaspoons baking soda
5. 3/4 teaspoons ground cinnamon (optional)
6. 4  1/2 ounces bittersweet chocolate, chopped
7. 23 tablespoons, 3  teaspoon (2 sticks) butter, room temperature
8. 1 cup, 1 tablespoon, 15/16 teaspoons sugar
9. 3/4 teaspoons vanilla extract
10. Sprinkles or other sugar decorations (optional)
11. Royal Icing (optional)
12. 3/16 teaspoons baking soda
13. 3/8 teaspoons cream of tartar
14. 3/16 teaspoons of powdered egg substitute
15. 3/16 teaspoons water
```

Figure 2: **Recipe after replacing two ingredients and scaling to six servings**

## 3 Kitchen Task Scheduler

Sous's task scheduler would be intended to help cooks who are cooking multiple recipes at once, such as creating a main dish with several sides for a dinner or even more elaborate meals for festivities and events. At its core, this task scheduler can be viewed as a pipelining algorithm with a few adjustments to account and optimize for the realities of cooking. Each recipe can be viewed as a series of steps that need to be performed in a certain order within a recipe, but the cook can switch between tasks associated with different recipes. In addition, the scheduler could add in additional cooks and just treat them as additional threads working on the same problem. Sous' scheduler will take in all the steps associated with the recipes selected and then create an ordering for these steps to minimize time spent and improve results in the kitchen, primarily by pipelining steps that could be considered "hands off" with "hands on" tasks and maximizing efficiency with kitchen appliances like stand mixers, stove burners, and ovens.

Before discussing the data needs for this scheduler to work effectively, it is necessary to describe what "maximizing results" in the kitchen will be considered. The key factor for determining success will be that each recipe is finished within a certain margin of the "meal time", or the time when the last recipe is finished. The intuition behind this decision is that every dish has a certain range of times after it is cooked that it could be considered at its best, such as how pizza or roasted vegetables are at their best immediately after they come out of the oven. This is considered to be a range of times for two reasons. There are some dishes, such as bread, roasted meats, or a custard, that need to rest/chill for a certain time before being served. So, the data representation of each recipes' "life expectancies" will be as follows [*minimal time before serving, optimal time to serve, maximum time after completion to serve*]. Each of these will be a numerical value measured from the end of cooking for a certain recipe. The first and third variables represent the outer bounds for a particular dish's serving times, while the second provides the ideal serving time. When actually making decisions, the scheduler will consider the first and third as hard requirements, and then try to optimize completion times to get time to serve as close to the optimal time as possible by calculating absolute difference across all recipes being prepared. Below is the ideal data requirements for both recipes as a whole and individual steps in recipes to demonstrate how the task scheduler could effectively pipeline.

*3.1 Ideal Data for Recipes*

a. A list of steps for the recipe
b. Time to serve in the format of [*minimal time before serving, optimal time to serve, maximum time after completion*]
c. Flag - Reheatable? - some dishes can be reheated before being served. This flag will include an optional step that can be taken to reheat a dish if the scheduler has to place its completion earlier in the task order.
d. Serving Temperature: Indicates a cold or hot dish
e. Main: A boolean symbolizing a main dish versus a side dish. To be used for prioritization between dishes when there is no acceptable schedule to serve all dishes in their absolute ranges. Intuition is that cooks care more that their main dish is served at the appropriate temperature.
f. Interchangeable Step Ranges: Some dishes require steps to be performed in a very specific order, but others are recipes composed of different components (such as a meat with a sauce) and these components can be made in any order. If the scheduler has to account that a particular appliance is being used for another step, the scheduler could use this flexibility for reordering.

*3.2 Ideal Data for Steps*

a. User interpretable description of the step
b. Step Duration in minutes
c. Engagement Level - This is what allows for step pipelining. Some aspects of cooking require full attention such as chopping ingredients, while other steps are completely hands off such as baking or stewing. Steps will be categorized as either "Hands On" (not able to be pipelined), "Hands Off" (able to be pipelined with other steps), or "Interval" (can be pipelined but requires attention at a certain time increment, such as needing to stir a sauce or flip a burger while searing). Interval values will be associated with descriptive text and time intervals for completion.
d. Appliance(s) used - Another factor for the scheduler to consider is that steps cannot be pipelined if they require the same appliance, unless it is an oven. Users will be able to input their kitchen appliances (number of stove tops, oven, blender, etc.).
e. Oven Temperature - for steps involving an oven
f. Oven Splitting Flag. Oven splitting is a common cooking method for allowing multiple dishes to use the oven at once; however, certain dishes (such as souffles) are very temperamental and cannot be oven split. If two steps require an oven and can be oven split, the average temperature will be used and times proportionally adjusted for the steps in the task order, with additional steps added to the task order for checking these dish's progress.

*3.3 Scheduler Decision Structure*

Asymptotic run time for this scheduling algorithm is not a particularly important consideration for two reasons: most users will only be uploading at most 5 or 6 recipes into the scheduler and recipe steps have to typically be performed in a specific order (unless the recipe has interchangeable components like an entree and a separate sauce but this is generally limited as a concern). The scheduler can take advantage of both of these real world limitations to input size and just create every possible task order and see which one minimizes time in the kitchen while having each dish as close to their optimal time to serve as possible. While creating these task orders, the scheduler can take advantage of oven splitting, pipelining different series of tasks, and other real world optimizations. The specific weight given to both of these heuristics would be something the user could specify in their preferences for Sous.

The schedule would start by building out the task order from each recipe's last step, and slowly adding on steps until every step is in the task order. The scheduler would then determine which order performs best in terms of the heuristics and then display to the user. There is a potential algorithmic optimization for the scheduler by dynamically storing solutions to subproblems within the task order so that if other threads ever reach having the same tasks left to schedule, they could access the optimum ordering for these tasks rather than recalculating. Tasks would be given unique labels, and the scheduler would check to see if a remaining pool of tasks has already been solved. This does hit a problem with that some tasks need to be completed within a certain time period of the task preceding/after themselves, which would mean that subproblems wouldn't be considered shared. However, this can be solved by tasks that have to be completed within a certain time window of each other within a recipe being abstracted to being sub-tasks within a larger task that is considered by the scheduler.

## 4 Limitations

The primary limitation for both the substitution and scheduling services are that they require heavily formatted and robust data in order to function with any success. The substitution service requires ingredients to have relatively consistent formatting to accurately identify different ingredients and quantities, and the scaling service prototyped was only able to scale recipes proportionally as no rules list was found for how to best scale individual ingredients as this is likely different from recipe to recipe. The scheduling service requires an immense amount of metadata for individual recipe steps and the recipes themselves. This scheduler was not prototyped because no dataset could be found with recipes that have steps with associated time durations, which is the core requirement for the scheduler.

## CONCLUSIONS

The goal of this project was to demonstrate the need and feasibility of creating a tool for home chefs based on ingredient substitution and scaling. We demonstrated how existing cooking technology solutions fail to meet this need and described research attempts at solving the problem of ingredient substitution and recipe scaling. These studies demonstrate that language processing models for analyzing ingredient context have a ways to go before they would create a commercially viable service for substitution and scaling. A partial implementation for a recipe substitution and scaling service based on explicit rules and proportional scaling was successfully built, and the data requirements , decision logic, and limitations for a kitchen task scheduler were enumerated.

## FUTURE WORK

Sous's range of viable substitutions could be improved in two distinct ways. The first would be developing a language processing model similar to those discussed in the background section that is able to compare massive numbers of recipes, either analyzing context to develop an understanding of ingredient properties or to create trees of closest substitutes, and then generate unique ingredient substitution rules for each recipe based on this understanding. The second route would come from fully building out Sous' web presence and web forum to begin collecting user recommendations for substitute ingredients. As user suggestions are aggregated, more recipe/context specific substitution rules and restriction on rules could be developed.

The scheduling tool would be continuously improved as users submit information about how long steps in recipes take them to complete. One of the major limitations for prototyping in this paper was that no dataset could be found with recipes that have time allotments to different steps in the recipe. As users submit information about how long steps take them, Sous could aggregate these submissions and determine generalized rules for how long certain cooking tasks take and apply them to recipes without that information.

## ACKNOWLEDGMENTS

Thanks to Allrecipes for providing the common ingredient substitutions for the service prototype, found at https://www.allrecipes.com/article/common-ingredient-substitutions/. Thanks to Ryan Lee and his site Eight Portions for providing scraped recipes from Epicurious for the

recipe database for the substitution service, found at https://github.com/rtlee9/recipe-box .

## REFERENCES

[1] Pesce, Anthony, 2013. Natural Language Processing in the Kitchen. Los Angeles Times, Los Angeles, USA. http://datadesk.latimes.com/posts/2013/12/natural-language-processing-in-the-kitchen/

[2] Y. Pan, Q. Xu and Y. Li, 2020. Food Recipe Alternation and Generation with Natural Language Processing Techniques," 2020 IEEE 36th International Conference on Data Engineering Workshops (ICDEW), 2020, pp. 94-97, doi: 10.1109/ICDEW49219.2020.000-1.

[3] Shirai Sola S., Seneviratne Oshani, Gordon Minor E., Chen Ching-Hua, McGuinness Deborah L, 2021. Identifying Ingredient Substitutions Using a Knowledge Graph of Food. Frontiers in Artificial Intelligence, 3, 2021. doi: 10.3389/frai.2020.621766

[4] Emmanuelle Gaillard, Jean Lieber, Emmanuel Nauer. Improving Ingredient Substitution using Formal Concept Analysis and Adaptation of Ingredient Quantities with Mixed Linear Optimization, 2015. Computer Cooking Contest Workshop, Sep 2015, Frankfort, Germany. ffhal-01240383f.

[5] "Julia Child Would Be Thrilled: Most Americans Prefer to Cook at Home." ReportLinker, 2020. Retrieved from https://www.reportlinker.com/insight/americans-cooking-habits.html

[6] Vladimir Sidorenko, 2017. "The Advantages And Disadvantages of Using Django. " Datafloq

[7] Chun-Yuen Teng, Yu-Ru Lin, Lada A. Adamic, 2012. "Recipe Recommendation using Ingredient Networks". WebSci '12: Proceedings of the 4th Annual ACM Web Science Conference, June 2021, 298-307, doi: https://doi.org/10.1145/2380718.2380757