

# R.E.M.I. Recipe Exploration and Modification Intelligence

**By Cole Bennett &**

**Scott Lorance**

## Table of Contents

1. Introduction	1
2. Setup	1
3. Operation	2
4. System Description	
a. NLP Techniques Used	2
b. Preprocessing	2
c. Intent Classification	3
d. Term Identification	3
e. Recipe Exploration	3
f. Recipe Modification	4
g. GUI	4
5. Dialogue Structure	
a. Dialogue Tree	5
b. Dialogue Description	5
c. Sample Dialogue	6
6. Knowledge Base	
a. Creation	8
b. Format	9
c. Samples	10
7. User Model	10
8. Analysis and Conclusion	11

## Introduction

R.E.M.I. is a tool designed to aid new and experienced chefs with its knowledge of cooking, recipes and ingredients. It is able to help cooks better use the materials they have on hand to create dishes they have not tried, along with providing definitions and other useful information.

This report will show how R.E.M.I operates, how the system's knowledge base is structured, and how to set up R.E.M.I. for yourself.

## Setup

R.E.M.I. is a Python 3 program that requires a number of different libraries and resources to operate. These can all be installed from the command line by typing the command `Pip3 install <dependency name>`.

The dependencies include:

- glob
- nltk
- spaCy
- tkinter
- sklearn
- numpy
- pandas
- quantulum3
- (in command line) `python -m spacy download en_core_web_trf`

## Operation

To run the program, open the project directory and open the python program GuiRunner.py with the IDE of your choice, then run the code. After a few moments, a new window will pop up with a GUI Dialogue box. Send a greeting to begin communicating with R.E.M.I.

After that, you are now talking to R.E.M.I. There are 3 types of questions that R.E.M.I. can help you with. Remi can provide definitions for cooking terms, make substitutions for ingredients, or help you choose a recipe to cook. Just ask for one of these in natural language, and R.E.M.I. will assist you.

## System Description

### NLP Techniques Used

In the design phase of REMI, one of the key goals was to incorporate as many techniques taught in class as possible. We ended up utilizing the following NLP skills in the construction of R.E.M.I: web scraping, rules based parsing, simple machine learning techniques, part of speech tagging, dependency parsing, Wu Palmer similarity and WordNet querying.

Initially we intended to use Google Dialogue Flow, but decided against it as we believed we could implement the functionality on our own.

### Preprocessing

Before any work could begin on the chatbot itself, a knowledge base and labeled training data was required in order for the R.E.M.I. to learn the functionality it needed.

The knowledge base was generated by scraping through food blogs and parsing recipe information. Further information on this is available in the section about the knowledge base.

The other important dataset was training data to classify the intent of users when talking to R.E.M.I. to identify the function they intended to access.

This is in the form of the document intenttrain.csv, which contains sample queries and a letter label indicating the correct response. These were generated manually with the help of family members to make a dataset of over 200 example queries to train on.

These were all lowercase, stripped of punctuation then used to train the intent

classifier after being vectorized using a simple count vectorizer. Stopwords were left in as this produced better results.

## **Intent Classification**

R.E.M.I. uses two separate models to identify the intent of the user, Logistic Regression and a Naive Bayes. Both models are trained on the same dataset, however by utilizing them in combination we can leverage the benefits of both on certain input cases.

The chatbot is set up so that one model can override the other if that model is certain and the other model has low confidence in its prediction.

If both models are certain but with different results, or both models are uncertain there is a special case to prompt the user to use different terms.

The models are trained to predict three intents:

Define- the user wants to know what a given ingredient or cooking technique is

Explore- the user is requesting a recipe that uses a given ingredient or set of ingredients,

Modify- the user would like to know what they can replace a given ingredient or technique with.

Based on the intent identified the chatbot selects a next step.

## **Cooking Term Identification**

The identification of cooking related words is critical to all of REMI's functions, so reliably extracting them from the user's input as well as recipes is absolutely crucial.

Cooking term extraction is performed using spaCy's english language transformer model to tag for nouns and verbs, and then iterate through all synsets for each word, traversing up the hypernym hierarchy until it reaches a preset hypernym that is indicative of a given word being cooking or food related. If one of these hypernyms are found, the original word text and corresponding cooking synset are saved and returned.

## **Recipe Exploration**

This is REMI's core functionality- using what it knows about the current user, it will attempt to find the best fitting recipe based on the ingredients and cooking techniques provided.

Cooking terms are parsed and a pandas DataFrame is constructed with the columns corresponding to the parsed terms. The user's catalog of valid recipes (determined in the modification intent) is then iterated through searching for the relevant terms in the ingredients and techniques. After all recipes are examined, the degree of the

highest match is found and all recipes that match to that degree are returned to the user. If no matches are found, then the user is simply informed that none were found.

## Recipe Modification

Modification is the other critical aspect of REMI's capabilities, and the key area where user state information is recorded and updated. Once this intent is reached, if there are any methods parsed out in the user's input, the query is assumed to be of some form like the following: "what can I do instead of frying this fish?," so any ingredient substitutions are ignored in this case. To find potential substitutions, REMI iterates over the list of ingredients that it knows that the user has not already expressed dislike for (i.e. has not already been passed into the modification intent) and finds the one with the highest wu-palmer similarity. After all substitutions have been found, REMI updates the disliked and allowed ingredients, disliked and allowed methods, and catalog of recipes that are valid search results for the user. This lets REMI remember not to suggest things that the user has already said they do not want and tune recipe suggestions.

## GUI

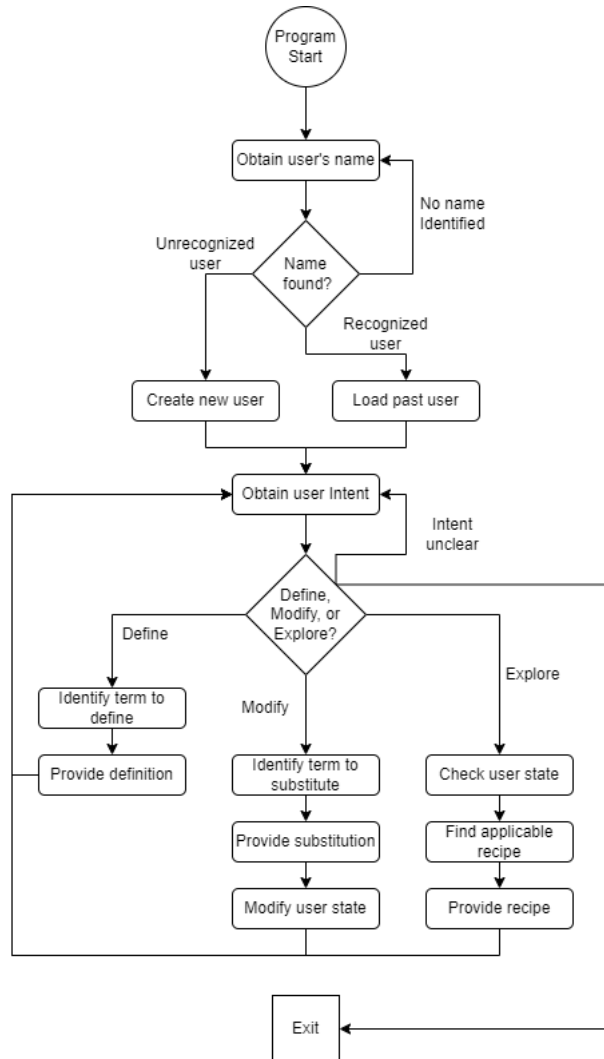
The GUI used for R.E.M.I. is an open source GUI made available under the MIT license by Patrick Loeber, available here:

<https://github.com/patrickloeber/python-fun/tree/master/chatbot-gui>

It uses the Python library tkinter to create the dialogue window, and calls the response function of REMI\_Main.py with the user input string to generate the output it displays back. Minor edits to the GUI itself were required to adapt it for use in this project, though a significant restructuring of the chatbot was required to make it fit the single function interface with the GUI.



## Dialogue Structure



Upon the initial execution of REMI, a user state must be determined before any dialogue functionality will work, so REMI looks for one of two things: if the user's response is one word, that word becomes the name for the user state, or if it is longer, then it searches for a proper noun and selects that as the user state. From there, it loads the previous list of user states, and selects the appropriate one if it is a returning user or

creates a new one if no match is found in the user list.

In REMI's main execution loop, user input is repeatedly taken in and classified as one of the three intent categories or as 'uncertain'. REMI uses two models to do this: a Naive Bayes model and a Logistic Regression Model. The logistic regression model tends to have high confidence in its predictions whereas the naive bayes model is more uncertain, so both of them are used to improve intent recognition performance.

### Definition Intent

If REMI thinks that the user is asking for a definition, it will parse the cooking terms from the user input. From there, it will iterate through the list of found definitions and append them to the output string.

### Exploration Intent

If REMI believes that the user wants to find a recipe, it will filter the current user recipe catalog based on the parsed cooking terms and return any that were found.

### Modification Intent

If REMI thinks the user is asking for suggestions, then it will parse out cooking terms, find the most similar according to

wu-palmer similarity (or lemma if no synset  
with wu-palmer score greater than 0.5 is  
found).

## Program Completion

Using the 'exit' keyword will indicate to  
REMI that the user is finished using it, and  
will trigger the saving of the current user  
state list to be read in upon the next starting  
of the program.

## Sample Dialogue

```
You: Start

REMI: Welcome to REMI_V1!
I'm REMI, the Recipe Exploration and Modification Intelligence!
To start, please tell me your name!

You: Hello REMI, it is very nice to meet you. My name is Scott

REMI: Nice to meet you Scott!
What can I help you with today?

You: I don't know what a coconut is, can you tell me what that means?

REMI: Sure thing, here's a definition for coconut
Coconut : the edible white meat of a coconut; often shredded for use in e.g. cakes and curries

You: Hey REMI, it's cole again

REMI: Welcome back Cole!
How can I help you today?

You: Can you find me a recipe that uses coconut and rice?

REMI: Here's what I was able to find for you:

Name: Chocolate Cereal Breakfast Bars
Cuisine:
Rating: 5.0
Rating Count: 1
Time: 0
prep Time: 0
cook Time: 0
Nutrition: {'calories': '264 kcal', 'carbohydrateContent': '38 g', 'fiberContent': '4 g', 'proteinContent': '5 g', 'saturatedFatContent': '3 g', 'sodiumContent': '247 mg', 'fatContent': '12 g', 'unsaturatedFatContent': '0 g'}
Serving Count: 8
Ingredients:
    rolled oats: 1.5 cups
    puffed brown rice cereal: 1 cup
    chopped raw almonds: 0.5 cup
    shredded coconut: 0.33 cup
    unsweetened cocoa powder: 3 tablespoons
```

kosher salt: 1 teaspoon  
 tiny pinch cayenne pepper: to taste  
 packed brown sugar: 0.33 cup  
 maple syrup: 0.5 cup  
 vegetable oil: 2 tablespoons  
 water: 1 tablespoon

#### How-to Steps:

Step 1: Preheat the oven to 350 degrees F (175 degrees C). Grease an 8x8 pan with butter and line with parchment paper.

Step 2: Combine rolled oats, puffed brown rice cereal, almonds, coconut, cocoa powder, salt, and cayenne in a bowl.

Step 3: Combine brown sugar, maple syrup, vegetable oil, and water in a small saucepan and bring to a simmer over medium-high heat. Stir, and let the mixture boil for 1 minute.

Step 4: Pour sugar mixture over the dry ingredients in the bowl and mix thoroughly. Transfer to the prepared pan, and smooth out the top, pressing lightly with a spatula.

Step 5: Bake in the preheated oven for 25 minutes. While still hot, use a spatula to score the pan into 8 portions. Let cool to room temperature before breaking into bars. You can also not score, and simply let cool, then cut with a knife.

REMI: Welcome to REMI\_V1!

I'm REMI, the Recipe Exploration and Modification Intelligence!

To start, please tell me your name!

You: cole

REMI: Welcome back Cole!

How can I help you today?

You: I am not a huge fan of cauliflower, what do you suggest I replace it with?

REMI: Sure thing, here's a definition for cauliflower

Cauliflower : a plant having a large edible head of crowded white flower buds

You: What could I substitute for cauliflower?

REMI: Cauliflower can be substituted with cabbage (Similarity: 90.9%)

REMI: Cauliflower can be substituted with cabbage (Similarity: 90.9%)

You: can you find me any recipes with cauliflower?

REMI: Sorry, I couldn't find any matches for you.

You: can you find me any that use cabbage?

REMI: I'm not quite sure what you meant, can you phrase that a little differently?

You: can you find me any recipes that use cabbage?

REMI: Here's what I was able to find for you:

Name: Apple Jack Pulled Pork Buns

Cuisine:

Rating: 5.0

Rating Count: 1

Time: 645

prep Time: 20

cook Time: 145

Nutrition: {'calories': '530 kcal', 'carbohydrateContent': '42 g', 'cholesterolContent': '89 mg', 'fiberContent': '3 g', 'proteinContent': '31 g', 'saturatedFatContent': '7 g', 'sodiumContent': '1172 mg', 'fatContent': '23 g', 'unsaturatedFatContent': '0 g'}

Serving Count: 8

## Knowledge Base

R.E.M.I.'s knowledge base consists of a dictionary of recipe objects, each containing nutritional information, processes and ingredients for cooking a delicious dish.

This section will detail how this knowledge base was assembled, how it is structured, and overview a few samples.

## Knowledge Base Creation

The recipe book that R.E.M.I. uses is first generated by the use of a web scraper. This web scraper is currently configured to begin scraping at [foodwishes.blogspot.com](http://foodwishes.blogspot.com).

Learning from issues encountered with the previous web crawler, this version is constrained to a much more specific domain- it only takes recipes found on the original site. Recipes on the original site are given in two possible formats: either a link to [allrecipes.com](http://allrecipes.com) containing fully written information (n=292), or a list of ingredients followed by a YouTube link (n=1,514). REMI's current functionality requires recipes to contain full information, so the knowledge base consists of the 292 recipes scraped from allrecipes.

Given allrecipes's consistent formatting of their websites, it proved more effective to operate on the data gathered directly from the urllib request rather than constructing a soup object using BeautifulSoup. Two unique and consistent strings were identified that specify the beginning and end of the actual recipe JSON data, so the data written out to the raw recipe file is simply the text between the two indices.

To avoid being banned from accessing the site, each time a new allrecipes link is scraped, the request is sent from a new Tor identity.

Once the webscraper completes, it generates a folder full of text files containing the recipes which then must be parsed. But all of these text files share a useful feature- they are structured as JSON files.

The parser makes a few string modifications to the imported file contents to patch up the JSON formatting that was damaged by the scraping process, before querying it for the various recipe components to add to a recipe object.

Not every recipe scraped contains all the attributes we look for, so every call against the JSON file is made to handle a key error



if it is not present.

These recipe objects are then added to a dictionary that is keyed by the recipe name.

This dictionary only has to be assembled once, and is then stored as a pickle for the R.E.M.I. to use later.

## Knowledge Base Format

The recipes are all stored as instances of the recipe class, which follow a format. Not every field is specified in every recipe.

Field Name	Data Type	Description
name	string	The name of the recipe
rating	double	The average rating of the meal
ratingCount	int	Number of ratings given
prepTime	int	The time in minutes to prepare the meal
cookTime	int	The time in minutes to cook the meal
totalTime	int	The total time in minutes to make the meal
nutrition	dict	A dictionary keyed by nutrient with quantity as the value
servingCount	int	The number of servings prepared
ingredients	dict	A dictionary keyed by the ingredient with the quantity as the value
howToSteps	list	A list of strings containing cooking steps
techniques	list	A list of strings listing necessary cooking skills

## Sample Entries

Here are a couple of randomly selected examples of recipes in the knowledgebase:

Name: Double Chicken Tea

Rating: 5.0

Rating Count: 2

Time: 555

prep Time: 45

cook Time: 510

Nutrition: {}

Serving Count: 10

Ingredients:

2 (4.5 pound) whole chicken  
3 large carrots, peeled and cut into 2-inch pieces  
3 ribs celery, cut into 2-inch pieces  
2 yellow onions, roughly chopped  
4 cloves garlic, unpeeled and halved  
2 small bay leaves  
10 whole black peppercorns  
7 quarts cold water, or as needed  
kosher salt to taste

How-to Steps:

Step 1: Slice breast meat off the chickens by cutting along either side of each breastbone, across the ribcage and down to the wing joints. Cut off the meat and save for another meal, reserving skin for chicken tea.

Step 2: Transfer whole chickens and skin from breasts into 1 extra large, or 2 standard (5 1/2- to 6-quart) pots. Divide carrots, celery, and onions between the pots, followed by garlic, bay leaves, and peppercorns. Fill each pot to the top with cold water.

Step 3: Place pots over medium-high heat and bring to a boil, watching carefully. Immediately reduce heat to low. Skim off any foam from the tops with a spoon. Adjust heat to maintain a very gentle simmer where only small bubbles poke up through the surface. Continue to let simmer for 8 to 12 hours.

Step 4: Use a slotted spoon to transfer the solids into a strainer set over a large bowl; discard chunks and pour strained broth back into a pot. Combine all the liquid into one pot. Strain broth again through a fine-mesh strainer into whichever pot was emptied.

Step 5: Bring to a boil over medium-high heat, skimming fat from the top; boil until reduced by half. Turn off heat and season with salt, tasting and adjusting as needed.

Step 6: Strain broth through a cheesecloth or nut milk bag into a container. Serve steaming hot in a pot as you would tea, or let cool and refrigerate until ready to serve.

Name: Deli-Style Roast Turkey for Sandwiches

Cuisine:

Rating: 5.0

Rating Count: 5

Time: 580

prep Time: 10

cook Time: 75

Nutrition: {'calories': '185 kcal', 'carbohydrateContent': '1 g', 'cholesterolContent': '102 mg', 'fiberContent': '1 g', 'proteinContent': '37 g', 'saturatedFatContent': '1 g', 'sodiumContent': '665 mg', 'fatContent': '3 g', 'unsaturatedFatContent': '0 g'}

Serving Count: 8

Ingredients:

1 (2 1/2 to 3 pound) skin-on, bone-in split turkey: breast  
1 tablespoon olive: oil  
2.5 teaspoons kosher: salt  
1 tablespoon freshly ground black: pepper  
1 teaspoon dried: rosemary  
0.5 teaspoon dried: oregano  
1 teaspoon smoked: paprika

How-to Steps:

Step 1: Preheat the oven to 450 degrees (230 degrees C).

Step 2: Pat turkey breast with a paper towel to remove any excess liquid and place into a baking dish just big enough to fit it. Use the tip of a sharp small knife to poke the skin of the turkey breast all over. Drizzle over olive oil and use tongs to rub; flip turkey breast over a few times, or until all surfaces are coated with the oil. You can also use a brush. Set aside for 15 to 20 minutes.

Step 3: While turkey rests, mix salt, pepper, thyme, rosemary, oregano, and paprika together in a small bowl until thoroughly combined.

Step 4: Use tongs to grab the turkey breast with one hand while using your other hand to sprinkle the spice rub evenly over the entire surface, including the sides. Finish with the skin side facing up.

Step 5: Add about 1/2 inch of water to the bottom of the baking dish.

Step 6: Place prepared turkey breast into the center of the oven. Immediately lower the heat to 300 degrees F (150 degrees C). Roast until a probe thermometer stuck into the center of the thickest part of the breast reads 150 degrees F (65 degrees C), about 1 hour and 15 minutes.

Step 7: Let cool before wrapping. Chill thoroughly in the refrigerator, about 8 hours, or overnight. Once meat is cold, slice thinly and serve.

The recipe class includes a toString function that returns it to an easy to read format.

For a complete list of recipes, the program RecipePrinter.py has been included in the project to view and print all the recipes at your leisure.

## User Model

The user model is centered around keeping track of user preferences, so it contains the following attributes: userName- the name provided by the user, userIngredients- the list of permissible ingredients to be checked against for substitution, dislikedIngredients- list of ingredients that the user has asked for substitutions for (used to update the permissible ingredients), similar lists for allowed/disliked cooking techniques, and thresholds for nutritional requirements, recipe ratings, and cooking time constraints, although these last three are not currently functional.

The model is updated upon completion of the modification intent, where the disliked methods and ingredients are updated and removed from the allowable ingredients and methods, and the recipe catalog is searched for the presence of any disliked terms in the recipe and removed if any are found.

## Analysis and Conclusion

R.E.M.I. Overall satisfies the expectations put out for it, but the model does have its advantages and drawbacks.

The primary advantage of R.E.M.I. is its ability to leverage a large amount of structured data for its knowledge base- the recipes provided are but a small sliver of the total that are available and already formatted on websites like allrecipes.com. Increasing the knowledge base is as simple as scraping more recipes, and all new knowledge can be easily integrated.

R.E.M.I. also demonstrates a clear grasp of NLP techniques, and does not rely on existing chatbot frameworks like dialogue flow or gpt for functionality. The dedicated intent categorization also is fairly effective at assigning user queries to a functionality of the chat bot.

The weakness of R.E.M.I. is that though it utilizes machine learning and neural networks for classifying user intent, the response generation is rules based.

Responses thus may seem a bit repetitive or robotic, the information contained may change but there are limited ways it can express mannerisms.

R.E.M.I. is not state of the art, however it serves as a good exhibition of natural language processing and a decent assistant with cooking related questions.