Recipe Substitution

Ezra Muskat, Tuvya Macklin, Solomon Firfer

1 - Motivation

Problem Statement

Food is an essential part of the human experience. While everyone has to eat to survive, food has taken a role in shaping communities, cultures, and households. Because of food's essentiality in our everyday lives, and the availability of a large variety of ingredients in today's markets, people have become obsessed with trying new recipes. While it's easy to find new recipes on the internet, the pool of recipes which a given person can eat can become a lot slimmer when factoring allergies and diets into the equation. The goal of this project is to create a tool which would help people who have dietary restrictions enjoy more recipes by converting recipes which don't adhere to their diets into recipes which do.

IO

To accomplish the aforementioned conversion, our model will take a recipe, defined as a list of ingredients as input, along with a dietary restriction. It will then find each ingredient that doesn't fit within the dietary restriction, search for the most similar ingredient that does, and replace the ingredient that doesn't fit with the one that does. Finally, the model will give the user a modified version of the original recipe which adheres to their diet, without any substantial changes.

Goals

The primary goal of this project is to create a solution to the problem stated above. We would like to create an application that converts recipes into similar recipes which adhere to user specified diets. Additionally, we would like to create a website that makes our model accessible to users who aren't tech-savvy or who don't know the API.

2 - Technical Challenges and Solutions

Data Collection

Classification Dataset

The first dataset we needed was a list of ingredients and whether they violated the dietary restrictions we worked with (vegan, vegetarian, dairy-free, and gluten-free). We struggled to find such a dataset online and eventually decided to create our own dataset by hand-labeling a set of ingredients. We initially started with the 500 most common ingredients and later added another 500 to enhance the training of the model using it.

There were two issues we faced when putting the dataset together. The first was that some ingredients have ambiguity when it comes to certain diets. For example, worcestershire sauce sometimes contains fish which would make it not vegetarian, and soy sauce usually contains gluten but sometimes is gluten-free. To resolve this ambiguity, we did research into what was more common and labeled the ingredients accordingly. The second related to the vegetarian label; there are many different types of vegetarians and we needed to decide on which we were labeling for. After doing some research into the various approaches, we settled on lacto-ovo vegetarians who eat milk and eggs but not fish since this seemed to be the standard and middle ground type of vegetarian that would be appropriate for the most number of people.

Recipe Dataset

Another dataset we used in our project was a recipe dataset[1]. This was a dataset, which we found online, was initially intended to be used training our distance model's embeddings. Those embeddings failed to prove useful and were scrapped, though we did parse the ingredients from the recipes in that dataset into the set of ingredients we used when creating the classification dataset.

Eval Framework Dataset

Evaluation proved to be a fairly tricky topic, as discussed in more depth below. We ultimately went with the RecipePairs dataset, taken from the SHARE paper, which takes the form of two datasets: the Recipe dataset, which contains several recipes with ingredients in a list format, and the Pairs dataset, which contains base-target pairs of recipe IDs referencing the Recipe dataset and restriction categories where the target recipe obeys them, but the base recipe does not.

Heuristic Model

The first model we developed to provide ingredient substitutes used simple heuristics to identify ingredients that violated the restrictions and made simple suggestions based on the violations. This was done by checking each ingredient for terms that indicate the type of food; for example, any ingredient with the word “beef” or “chicken” was flagged and replaced with “tofu”.

Research went into the general categories of food that violate each restriction, along with suitable replacements for each category. Then lists of terms which indicate that an ingredient belongs to one of the categories were written up. For example, one category for the dairy-free diet is any type of cheese. The list for this category includes terms like “cheese” (obviously), “mozzarella”, “cheddar”, and any other terms that have to do with cheese. The replacement for anything in this category is “vegan cheese”.

Distance Model

Filtering Model

The filtering model is a fundamental part of the distance model, whose purpose is to label ingredients as part of, or not part of a given diet. To accomplish this, the model embeds ingredients using a sentence similarity model, specifically sentence-transformers/all-MiniLM-L6-v2[2], and uses a (not pretrained) dense neural network to translate the embeddings into the correct labels. The filtering model went through a ton of testing iterations, to help fine-tune its prediction abilities. In addition to all of its necessary functionality, which includes its initialization, training, and prediction capabilities, it also has the ability to perform k-fold validation on a dataset. The chosen sentence similarity model was chosen because it is fast, lightweight, and reasonably accurate.

Some additional features which the filtering model has are the abilities to save and load decoder heads and to manually override its verdicts. The former enables the model to skip the training step when starting the application, while the latter enables support for a potential user feedback feature.

Similar Ingredients Model

The second major component of the distance model is the similar ingredients feature. We needed a way to find ingredients that could be alternatives to those that violated the diet. The approach we took to this is to use word embeddings which can find words similar to a given word.

Initially we attempted to train our own embeddings on only recipes using the recipe dataset discussed above. This would have had the benefit of being specialized for food and cooking since that would be the only thing it was trained on. However, it did not perform well and made suggestions like "mashed potatoes" when asked for foods similar to "milk". This is probably because of the small size of the dataset; our dataset had around 50k recipes and to train a word embedding model you generally need hundreds of thousands of examples.

After this, we decided to try pre-trained general purpose word embedding models. After testing out a few, we settled on the Google News 300 embeddings. This model has two distinct advantages: one, they provide meaningful and accurate food alternatives; two, they have good support for phrases with multiple words. This means that the Google News embeddings are able to handle multi-word ingredients out of the box. Many of the other models were not able to handle common ingredients like "brown sugar" because of this factor.

With the embeddings chosen, they needed to be prepared in two ways before they could be used for ingredient alternatives. The first was to prevent them from suggesting anything besides food. By default the embeddings will suggest any words that are similar to the ingredient, so you may get “cow” or “dairy” for the word “milk”. To filter out any non-food words, we created a list of valid ingredients from the recipes dataset and only allowed the model to make suggestions of words on the list. The second was to make sure the model could handle all multi-word ingredients. Despite its impressive support for this, it still could not handle every ingredient. When an ingredient causes this issue, we average the embeddings for each token and look for the ingredient with the embedding most similar to that.

We did try using BERT embeddings to get more context aware suggestions which would be very helpful for recipes where context is so significant, but we failed to get this working with reasonable performance. It was taking minutes or hours for a single suggestion.

Integration

The final step in creating the distance model was integrating the filtering and similar ingredients models together. The integrated model works in three steps. First, the filtering model is used to identify ingredients that violate the dietary guidelines. Second, those ingredients are passed to the similar ingredients model to find alternatives. Note that at this point the alternatives don’t necessarily fit the diet, all we have is a list of similar ingredients. Third, the filtering model is used again to pick out valid substitutions from the list of alternatives provided by the similar ingredients model. These substitutions replace the problematic ingredients in the original recipe and the new recipe is returned.

This process went very smoothly. The main issue we ran into was managing python imports and packages. This took time and research to figure out.

One major bug that came up once we did the integration was that the model could not handle the scenario when it couldn't provide any suggestions. To resolve this, we identify when this occurs and leave the problematic ingredient in the recipe with the words "no substitute found" appended to it.

Evaluation Framework

As mentioned earlier, evaluation was a fairly tricky topic to handle. Much of the literature on ingredient substitution models focus specifically on substitution ranking models - models where a single ingredient is input, and the model spits back out a series of ranked substitutions for that ingredient. As we took an entirely different approach to ingredient substitution, the approaches to evaluation taken therein were not particularly useful to us.

What ultimately helped us was the discovery of the SHARE paper, an ingredient substitution taking a similar approach to the task ingredient substitution, albeit with a different technical approach. In addition to borrowing their RecipePairs dataset, we also adopted their overall evaluation approach, of treating this as a multilabel classification problem where each ingredient in a recipe is a single label. We therefore went with classical classification metrics - precision, recall, and F-1. As the theoretical label space is enormous, the number of true negatives for any given generated recipe dwarfs all other factors and makes regular accuracy a useless metric; we therefore went with the Jaccard index metric, also known as the intersection over union, which is equivalent to accuracy with the true negative terms taken out.

Website & User Interaction

For enabling end-user use of our project, we created a website. This website uses Flask on the backend, React on the frontend, and is hosted on Render. We developed the backend and frontend early on, settling on a simple UI of checkboxes for restrictions and textboxes (later converted to a single, large textbox) for ingredients.

Deployment took a few steps. The biggest initial hurdle was that we had not taken into account deployment when developing the front and backends, as they had largely been tested on our local machines. As a result, there were a number of small issues surrounding integration that cropped up when attempting to deploy and use the website on the internet. After that, we had to alter how we handled persistence, as Render does not guarantee statefulness; we shifted our approach to have requests handled directly through POST responses. As a side bonus, this also ensured that the website would never display an old response.

3 - Tests and Benchmarks

In conducting our evaluations, we looked at how each model performed on the overall dataset, as well as its performance for each restriction. The rationale for which metrics were chosen is discussed earlier.

Heuristic Model

Overall

IoU: 0.28907005257612123

Precision: 0.43103496186105106
Recall: 0.4674269657137189
F1 Score: 0.44849393870943455

Vegan

IoU: 0.27257748656033776
Precision: 0.40832239753410443
Recall: 0.45052424225842974
F1 Score: 0.42838646674017494

Vegetarian

IoU: 0.2848157131785171
Precision: 0.43045862007255925
Recall: 0.45705122299507117
F1 Score: 0.4433565222733912

Dairy-Free

IoU: 0.30149885639203855
Precision: 0.44556235571633024
Recall: 0.48253060843016915
F1 Score: 0.4633102133148872

Distance Model

Overall

IoU: 0.2793791574279379
Precision: 0.4288017195056421
Recall: 0.4449814126394052
F1 Score: 0.43674176776429807

Vegan

IoU: 0.2512660693416439
Precision: 0.38738738738738737
Recall: 0.4169360051712993
F1 Score: 0.4016189290161893

Vegetarian

IoU: 0.2760768033212247
Precision: 0.44370308590492075
Recall: 0.4222222222222222
F1 Score: 0.4326962179747865

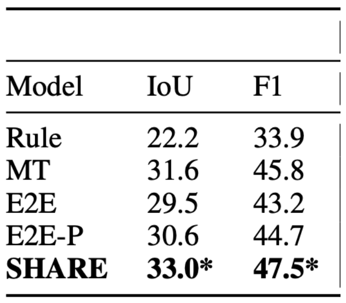
Dairy-Free

IoU: 0.29865030674846627
Precision: 0.44759102611254137
Recall: 0.47298872910998835
F1 Score: 0.4599395313681028

Comparisons

We can see from these results that the distance model did not live up to our expectations. While the heuristic model was intended to be a baseline for the distance model, it actually wound up outperforming the distance model. A number of possible reasons for this exist, such as issues with the particular embeddings we used, insufficient data and/or incongruencies between the datasets used for different parts of the distance model, and potentially even an inherent inability for the particular architecture of our distance model to perform this kind of ingredient substitution well.

Of note is that, while these results seem poor, they are not as bad as they seem at first glance. While we did not take the exact evaluation approach used by the authors of the SHARE paper, we used a similar enough one that it is worthwhile to compare the two (see the figure below. The results we achieved with our heuristic model are close enough to those achieved by their transformer model that, even applying a 5-point penalty to account for potential evaluation discrepancies, in-place ingredient substitution as a task may very well have a very low ceiling when it comes to what modern machine learning models are capable of.



Model	IoU	F1
Rule	22.2	33.9
MT	31.6	45.8
E2E	29.5	43.2
E2E-P	30.6	44.7
SHARE	**33.0***	**47.5***

4 - Conclusion

Summary

Our project tackled the challenge of enabling users with dietary restrictions to enjoy a broader range of recipes by automatically substituting incompatible ingredients. We explored both rule-based and embedding-based models, building a robust system that detects problematic ingredients, identifies suitable alternatives, and reconstructs recipes accordingly. Through careful data collection, custom model design, and evaluation using adapted classification metrics, we demonstrated the effectiveness of our approach. The final product—complete with a user-friendly website—empowers users to transform recipes to meet vegan, vegetarian, gluten-free, or dairy-free requirements with minimal manual effort.

Sources

1) Kaggle. “Recipe Ingredients Dataset.” *Kaggle*, 19 Jan. 2017, www.kaggle.com/datasets/kaggle/recipe-ingredients-dataset.
2) “Sentence-Transformers/All-Minilm-L6-V2 · Hugging Face.” *Sentence-Transformers/All-MiniLM-L6-v2 · Hugging Face*, huggingface.co/sentence-transformers/all-MiniLM-L6-v2. Accessed 11 May 2025.
3) Shuyang Li, et al. "SHARE: a System for Hierarchical Assistive Recipe Editing." *arXiv preprint*, 2022.