Ingredient Identification and Recommendation of Food Recipes

Dhivya Swaminathan, Shilpa Singh, Sumeet Mishra

December 19, 2019

Abstract

Major role of food in our everyday life has motivated many current day technologies revolving around it. With the prevalence of internet, whole world is connected, and different users of different countries are sharing millions of new recipes on the internet world-wide. So, as a result users are not aware of the all the recipes on the web. This has sparked a growing interest in developing food-related applications that help in harnessing this need. In such scenario, a recipe recommendation system which can predict the recipes by taking the list of ingredients as input will be very useful to them. Many AI based applications have cropped up that address these needs. In this project, we obtained a corpus of 469 recipes using a web crawler from Allrecipes.com and built an AI based NLP application with the main components of a Recipe Recommendation system which will take ingredients as input and suggest the recipes for these ingredients in two steps : Ingredient Identification from recipe text and Recipe Recommendation given a list of ingredients. The first step model has an Precision of 95%, Recall of 93% and an F1-Score of 94%, while the second step has a train accuracy of 92% and a test accuracy of 64% when trained with 155 recipe files.

1 Introduction

Our project has a long term and short term goal. Due to time constraints for this course, we intend to accomplish the short term goal for the course's project requirements.

1.1 Long term Goal

The long term goal of the project is to build and deploy a recipe recommendation system on any compatible speech based system. We intend to use the system's speech processing capabilities to parse user speech input of ingredients and quantities into text. This parsed text is our input. A

separate module is built using a recipe bank obtained by crawling recipe database. We train a Neural network to identify ingredients from the recipe text by feeding it recipe text as input and an ingredient annotated file as the label.

Post the recipe parsing for ingredients is done, we take the speech parsed user input of ingredients and quantities, cross reference our recipe database output from the neural network to rank top 10 best recipes for the ingredients provided by the user. In case there are no recipes that match the ingredients and quantities, we intend to provide the best match and also output what more is required to cook the recipe.

1.2 Short term goal

The short term goal is to achieve major parts of the final system we intend to build. The parts we intend to complete to achieve the short term goal are as follows:

1. Create a web crawler that scrapes recipe text from multiple websites according to a format as shown in Figure 1.
2. Use parsing techniques and stanford core-nlp libraries and tag the ingredients as part of train data labelling.
3. Built a neural network that identifies ingredients based on the above training data set.
4. Built a system that correctly predicts a set of recipe for given ingredients. In our case, this is again a neural network.

2 Previous work

The following is the compilation of the existing works in this area. Although the input varies in that it might be a picture of food items, speech instructions from user, or raw text, the core of the idea remains the same. The approaches taken to implement the idea is greatly varied.

2.1 Pic2Recipe

MIT's Computer Science and Artificial Intelligence Laboratory in collaboration with Qatar Computing Research Institute, built an artificial intelligence system **Pic2Recipe** to enable Recipe recommendation. The goal of this system is to predict the ingredients and recommend close match recipes from pictures clicked by the users.

The team scraped over 1 million recipes from websites like All recipes, food.com etc with a wide distribution of dishes and cuisines. A neural network model was then built that identified ingredients and recipes from the food images. the architecture of this system is shown in Figure 2. (4)

2.2 Deep-based Ingredient Recognition for Cooking Recipe Retrieval

As an addition to the existing recipe recommendation systems of identifying ingredients, this paper talks about going beyond that and also includes the nutrient information before suggesting recipes. Estimation of nutrition facts is helpful in many health relevant scenarios and pose to be a great value add to the existing systems. Chen et al experimented on a large Chinese food dataset of food images with complex appearances for this project. (5)

2.3 Images Recipes: Retrieval in the cooking context

This paper addresses the picture-recipe alignment problem and their approach is validated on the famous Recipe1M data bank. In this paper, Carvalho et al have discussed about building a deep neural model that is trained on recipe texts and images that rely on a multi-modal retrieval learning objective function as shown in the Figure 3. (6)

3 Methodology

As discussed in the project goals, this paper talks about a simple sequence architecture that would help accomplish the goal of recipe recommendation using deep neural networks. We obtain a corpus of 469 recipes from allrecipes.com using a web crawler implemented in python using a BeautifulSoup library, annotate ingredients in recipe text using Stanford Core-NLP library, Build a Neural network

model to identify ingredients in recipe text and finally build another model to recommend recipes given a list of ingredients. The architecture of the same is as outlined in Figure 4. The main steps to the architecture is as listed below:

1. **Data Collection**: We collected around 470 recipes from the Allrecipes website, its sub-categories and all its pages using a web-crawler using the BeautifulSoup package in python. From these above scraped recipes, a set of unique ingredients was obtained and a reference data was created for lookup to be used for the annotation task.

2. **Automated Annotation of Recipe Corpus**: All the recipe files were parsed using a script, divided into sentences using core-nlp and each token in the sentence annotated as "B-OTH" if it was an ingredient and "O" if it was not an ingredient.

3. **Neural Network for Ingredient Identification**: For ingredient identification in recipes, Bi-directional LSTM with ELMO Embedding was employed which gave good accuracy in predicting ingredients.

4. **Neural Network for Recipe Prediction**: For recipe recommendation, a text classification model was developed to predict recipes for a list of input ingredients.

3.1 Data Collection

We collected around 470 recipes from the website allrecipes.com and its sub categories from all its pages as listed below using a web-crawler in python. This is the maximum number which we could get from this recipe site.

1. https://www.allrecipes.com/

2. https://www.allrecipes.com/recipes/76/appetizers-and-snacks/

3. https://www.allrecipes.com/recipes/78/breakfast-and-brunch/

Allrecipes is a social networking platform in the food domain. The company founders are archaeology graduate students of University of Washington - Tim Hunt, Carl Lipo, Mark Madsen, and David Quinn. Members of allrecipes.com can post recipes that are later edited by the allrecipes.com staff. Members can also rate, review and upload pictures of the dish prepared using

the recipe. There are many functionalities available on allrecipes.com that include filtering recipes based on cooking style including grilling, baking, etc, filtering based on holidays specific dishes, filtering based on specific ingredients, etc.

The **BeautifulSoup** package from python was used to implement the crawler. The links for all the recipes listed in the above urls were first crawled and stored using the get_data_for_page() module as shown in Figure 5.

BeautifulSoup (3) is a python library that is used to scrape data from HTML and XML pages. It works with the parser and provides ways to search, iterating through and modify the parse tree.

This module first extracts all recipe blocks/cards from the article tag of class 'fixed-recipe-card'. Each **recipe card** is then perused to find the **link to the recipe page** from the 'href' value of the 'a' tag from the div tag of class 'fixed-recipe-card__info'. The **title** of the recipe was obtained from the h1 tag of class 'recipe-summary__h1', **author name** from span tag of class 'submitter__name', the list of **ingredients** from span tag of class 'recipe-ingred_txt added' and the **directions** from span tag of class 'recipe-directions__list–item'. These extracted items were encapsulated in a Recipe class with the definition as shown in Figure 1.

Post this, we also tried annotating (manually labelling) ingredients in recipe file using the WebAnno tool, but since annotation of a large number of files manually was extremely time consuming, we resorted to automate this process of annotation using the Stanford Core-NLP libraries. This helped us prepare training labels for labelling ingredients faster and in a more efficient and accurate manner.

3.2 Automated Annotation of Recipe Corpus

Ingredient Identification in a recipe corpus is analogous to a Named-Entity Recognition task,with the difference being that it is specific to a food corpus and there are no generic models available in core-nlp libraries which can annotate an ingredient.

Manual annotation of such a corpus which is specific to a domain like a corpus containing news articles,or a corpus having sports related information or a food corpus with recipes like in our

scenario is the most viable solution. We however automated the process of annotation using a script because there were around 400 recipe files and we needed to label all the text tokens which were ingredient as "B-OTH" and the ones which were not as "O" which would have been a time consuming task if done manually.

Also after annotating each token in the recipe text,we wanted to insert a start-of-sentence marker between tokens of distinct sentences to segregate and identify each sentence in the recipe which was important to prepare our training data in the right format as were using models which were to be trained with the context information. We followed certain steps to create the reference file containing the name of all ingredients and the annotation files which were as follows:

1. We wrote a script to scan the Ingredient section of all the recipe files and then removed the numbers and words like "cup","tablespoon","teaspoon" using regular expressions. The remaining words were mostly food items.

   Post this, a set operation was performed to get such unique list of food items. After that, we manually corrected the list to ensure that we have only parent level ingredients.For example, if two entries had cheddar cheese and feta cheese, we only kept cheese as our ingredient for identification.

   Same thing we did with food items like apple pie and apple sauce and retained only apple. This reduced our number of food ingredients and combinations to be matched which was a disadvantage,but since we did not have the time to manually annotate all the 470 files, we went with this approach to correctly identify only the parent ingredients.

   This was the first step to prepare a reference list of all the ingredients which we can use as a lookup in the script which annotates all the recipe files. There were around 200 unique ingredients in our reference list.

2. Each recipe file was then taken and using stanford-core nlp pipeline, the text was processed to create sentences. Also, all the punctuation and numbers were removed from these sentences.

They were then tokenized and a token "sentence start" was inserted before first token every sentence. This was to segregate the entire food corpus into distinct sentences which was an important requirement for our training data.

After that, every token in the file was matched against our reference list of ingredients prepared in the above step and the token was labelled as "B-OTH" if there was a match indicating that it was an ingredient and "O" if it there was no match. In short, we annotated every word of the corpus to identify if it was an ingredient or not. The labels "B-OTH" and "O" were selected to identify ingredient and non-ingredient to be consistent with CoNLL format of annotation.

3.3 Neural Network for Ingredient Identification

3.3.1 Machine Learning Approach

As discussed before, the problem which we were trying to solve here was analogous to Named Entity Recognition where we trying to train a model initially with the recipe text and its labels identified as entities so that when given a new data-set it could identify the entities present in it from the set of entities it has already seen before. There are broadly 2 main methods to approach this problem. First approach is to treat this problem as Multi-Class Classification where the named entities are our labels so that we can apply different classification algorithms. The challenge here is that identifying and labeling named entities require thorough understanding of context of the sentence and the sequence of the words in it. Another approach for this kind of problem is Conditional Random Field(CRF) model. It is a probabilistic graphical model that can be used to model sequential data such as labels of words in a sentence.The limitation with CRF model is that it is able to capture the features of the current and previous labels in a sequence but it cannot understand the context of the forward labels and there is extra feature engineering involved with training a CRF model which makes it less appealing to be used as an easy solution. So, we decide to take the approach of treating this problem as a Multi-Class text classification and proceed further.

3.3.2 Multi-Class Text Classification

Multi-Class Text Classification can be done in various ways. Here, we are following a Neural Network based approach because it allows us to consider the input text both as bag-of-words input or as a Sequence based input. The advantage of considering the input text as a Sequence is that we can use different Sequence based models like Convolutional, LSTM and Word Embedding to define our Entity Recognition network architecture. In our experiments, we tried various approaches like transforming input to countvectorizer form following a bag-of-word approach and adding dense layers to build a neural network which can classify each word as ingredient or non-ingredient. We also tried convolution based approach where we transformed the input to an ordered sequence and applied convolutional layers followed by a dense layer with sigmoid activation. Lastly, we tried a Bidirectional LSTM with Elmo Embeddings on a sequenced input which gave us the best results. We used accuracy, precision and recall as our performance evaluation metrics on these models.

3.3.3 Neural Network Architecture

The most important strategy in building a high-performing neural network architecture for Entity Recognition task is to make it understand the context of the sequence well. LSTMs by design are well suited for this task but in our case since our problem was Entity Recognition, we needed a Bidirectional LSTM because since it remembers the context of the input sentence and uses both past and future labels to make predictions. A bidirectional LSTM is a combination of two LSTMs, one runs forward from right-to-left and one runs backward from left-to-right. We have also used residual connections between the Stacked Bidirectional LSTM models to address the degradation problem of deep neural networks. We have used Elmo as Embedding to extract features from the input text which was fed in a fixed-length sequence format. Elmo has a great understanding of the language because it is trained on a massive data-set of about 1 Billion word. ELMo uses a pre-trained, multi-layer, bi-directional, LSTM-based language model to obtain an embedding representation for each word in the input sequence. These features extracted from Elmo are then given as input to another two stacked layer of Bidirectional LSTM with a residual connection in our architecture. The output of this is given to a dense layer wrapped with a time-distributed function so that we can get exactly one-to-one mapping between the input sequence and the output sequence.The most important advantage of using Elmo as an Embedding is that we do not have

to do feature engineering from the input text. The features are extracted using an unsupervised approach from the input text by Elmo in the form of a word embedding which are then fed to LSTM layers. We however train the model from this point giving it the embedding representation as for each word as the input and the label which it has the output. This is a semi-supervised approach to training a Neural Network model for a Named Entity Recognition task.

3.3.4 Pre-processing and preparation of Training data

The most important step in our execution was to prepare the training data in a way so that the model can understand the context as well the label of each token in that context. For this supervised part of the training, we needed to identify what could be the input X and Y and their corresponding dimensions. We tried few approaches here and compared their performance by observing model prediction on test data. For the first approach, we considered the entire recipe text in a single recipe file,removed the punctuation and tokenized them into tokens. Then each of these tokens were annotated as "O" for non ingredient and "B-OTH" for ingredient using the script we developed for annotation. The output of this script was a file with two columns, first column containing the token and the second column containing the tag "O" or "B-OTH'. We then prepared two parallel python lists from this file, the first list containing all the tokens and the second list containing the tags for these tokens in the same index position. Once we collected all such lists from each file in a collection, we scanned the length of each to find the maximum length of the input sequence. The maximum length was 877. Then we padded each input and output list to 900 to make them fixed length. This we fed as input X and output Y to the model for training in the first approach. In the second approach, we took each recipe text file and passed it through core nlp processing pipeline and extracted sentences out of it. Then we removed punctuation from each sentence and tokenized them. After that a start-of-sentence marker was inserted between each sentence and then everything was tagged as "O" or "B-OTH". This was written to a file as two columns again, first as token and second as label. This time when we prepared our input sequence, we just took each sentence separated by the start-of-sentence marker in a list and its labels as another list. We prepared a collection of such lists which contained all the logical sentences in the entire recipe corpus and the corresponding collection of labels. There were total 3936 sentences in the recipe corpus. Then we scanned the length of each list in this collection to find the longest sentence in the corpus. It was 50, so we padded all the input and output sequence till length of

52. This prepared a fixed length sequence input data for training and a 52 dimensional output vector Y as labels. We converted the labels to 0 and 1 depending whether it was "O" or "B-OTH". Since, the Elmo takes the input X in word form we did not convert it to a number mapped to a vocabulary. Our input X to the model was a 3936 sample of 52 dimensional X vector of words and 52 dimensional Y vector of 0 and 1. We used this to train our model and got fairly good results with validation accuracy as 98 percent. The predictions on the test data was pretty good with this approach.

3.3.5 Keras Implementation library

We have used keras functional API to build our models. The Keras Python library makes creating deep learning models fast and easy. The functional API in Keras is an alternate way of creating models that offers a lot more flexibility, including creating more complex models. Models are defined by creating instances of layers and connecting them directly to each other in pairs, then defining a model instance that specifies the layers to act as the input and output to the model.

3.3.6 Loss function and Optimizer

We have used sparse categorical cross-entropy as our loss function. This is because the problem at hand is treated as a multi-class classification problem and the output has a dimension of 52 out of which only few will be 1 for the ingredients and rest are 0. By specifying that the output is sparse, neural network will take care of any imbalance issue in the back propagation error. The Optimizer used is Adam Optimizer. Adam algorithm calculates an exponential weighted moving average of the gradient and then squares the calculated gradient. This algorithm has two decay parameters that control the decay rates of these calculated moving averages. The main advantage of this is that it avoids overshooting and fast change of gradient and adjust the rate accordingly as the global minimum gets closer.

3.4 Neural Network for Recipe identification

We have approached this as a Multi-Class Text classification problem. But before we look at the model architecture let's discuss about the data preparation which is a important part of the model.

3.4.1 Data Preparation

Firstly, we separated the ingredients from the annotated text with the help of user-defined function named 'ingredient_finder'. The function takes annotated text as input and separates 'B-OTH' tag words which are our ingredients from 'O' tag words. Then it stores all the words in an array and returns the array.

We then created a data frame using pandas library and stored recipe file name and recipe name as two columns of the data frame. Lastly, we ran our 'ingredient_finder' function to generate ingredients of each recipe stored in the data frame.

We have done this experiment in 3 phases. First by taking 32 recipes, second by taking 155 recipes and third by taking 501 recipes. Each time we have to make data frames and append data frames(for the second and third experiment) to make our code reusable. This also addresses the issue of scalability. If we get more data our model can handle those extra data without much changes in the code.

Once we got the data prepared we prepared train and test data out of the result data frame named 'food_df'. We took 20% data as our test data. We took all the data as train data as we used unique recipes in our experiment and our model needs to see all the unique data to generalize better.

We then created train tags and test tags as recipe names and train posts and test posts as ingredients from their respective portions of data. Then we train our tokenizer with train posts and converted x-train and x-test of text data to matrix. Similarly, we transformed y-train and y-test data with the help of label binarizer.

Label binarizer assigns a unique value to each label in a categorical feature. It converts multi-class labels also to binary labels easily with the transform method. At prediction time, it assigns the class for which the corresponding model gave the greatest confidence. Label binarizer makes this easy with the inverse transform method. Next, we defined a user defined function named plot history which captures the accuracy and loss of our model and plot them to visualize the trend.

3.4.2 Model Architecture

We built our model with 3 dense layers and 2 dropout layers. The dense layers also known as fully connected layers which connect every node to all of it's previous and next layer's nodes. We added dropouts to avoid overfitting the model to our train data. We used 3 activation layers 2 of which are 'relu' and the last one is 'softmax' to highlight the important features of the data to our model to learn. We have used 'categorical cross entropy' and 'adam' as loss function and optimiser for our model.

3.4.3 Model Prediction

We have defined another user-defined function to predict/suggest recipes. It take ingredients array as input to the function and then it converts it to matrix with the help of tokenizer. Then we convert it to numpy array and that array is passed to the model.predict function which is an inbuilt keras library function.

The function gave us back the probability of all the recipes and we have taken the maximum probability of recipes in otherwords the recipes which are more suitable from the given set of ingredients. Then we converted the probabilities to text values with the help of text label that we trained with encoder classes and it gave us recipe names that could be made from those ingredients.

4 Model Evaluation

4.1 Evaluation of Ingredients Identification model

We used an ELMo (Bi-LSTM) Neural network model for ingredients identification. We trained our model with 469 recipe text files crawled from allrecipes website. We used the following metrics to evaluate the performance of the ingredients identification model:

1. **Precision** : In general, precision is the fraction of retrieved documents that are relevant to the query in terms of information retrieval. Mathematically, It is the fraction of true positives over all the positively predicted values (true positives and false positives). (10)

$$\frac{TruePositives(TP)}{(TruePositives(TP) + FalsePositives(FP))}$$

2. **Recall** : In general, recall is the fraction of the relevant documents that are successfully retrieved in terms of information retrieval. Mathematically, it is the ratio of true positives over the sum of true positives and false negatives. (10)

$$\frac{TruePositives(TP)}{(TruePositives(TP) + FalseNegatives(FN))}$$

3. **F1-Score** : F1-Score, a measure of test accuracy is the weighted average of precision and recall. Mathematically, it is given by the below formula: (11)

$$\frac{2}{(Recall^{-1} + Precision^{-1})}$$

So, for the ELMo (Bi-LSTM) model built for ingredients identification, we obtained a precision of 95%, recall of 93% and an F1-Score of 94%. This can be considered a model with good performance. This is as shown in Figure 8.

4.2 Evaluation of Recipe Recommendation model

The Neural network model built for Recipe recommendation is a text classification model. Experimentation was carried out by training this model with varying number of training size. i.e., we trained it with 32, 155 and 501 recipe files. We used train and test accuracy as the metrics to evaluate the performance of this Recipe Recommendation model.

Accuracy : Accuracy is the fraction of predictions our model predicted correctly. Mathematically it can be defined as the number of correct predictions out of all the predictions made by the model. (13)

$$\frac{\text{Number of right predictions}}{\text{Total number of predictions}}$$

We obtained 87 %, 92 % and 89% as training accuracy for the training size of 32, 155 and 501 recipe files respectively and 42%, 64% and 49% as our test accuracy respectively. The results of this as shown in Figure 9. The low accuracy of the classification model is due to less training data. Although we got low accuracy for our test data we got very interesting recipe suggestions from our

classification model.

5 Results

This is a section about the different experimentation carried out for the two models and a discussion about the results illustrated using a few examples.

5.1 Results of Ingredient Identification model

Two sample results have been discussed here:

5.1.1 Sample - 1

INPUT : The input is a list of the following words as part of recipe text -

['serve','hot', 'with', 'tortillas', 'and', 'shredded', 'cheese', 'allowing', 'everyone', 'to', 'make', 'their', 'own', 'fajita', '.', 'PADword', 'PADword']

OUTPUT : The output is a list of classes the input words from recipe text were classified into

['O', 'O', 'O', 'B-OTH', 'O', 'O', 'B-OTH', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']

5.1.2 Sample - 2

INPUT : The input is a list of the following words as part of recipe text -

['in', 'a', 'large', 'bowl', 'cream', 'sugar', 'and', 'butter', 'or', 'margarine', '.', 'PADword', 'PADword', 'PADword', 'PADword', 'PADword', 'PADword', 'PADword']

OUTPUT : The output is a list of classes the input words from recipe text were classified into

['O', 'O', 'O', 'O', 'B-OTH','B-OTH', 'O', 'B-OTH', 'O', 'B-OTH', 'O', 'O', 'O', 'O', 'O']

5.2 Results of Recipe Recommendation model

First we wanted to check how is our model performing after trained with 32 recipe files. Then we increased our training data size to 155 and 501 respectively to compare the prediction results among all the 3 models.

5.2.1 Experiment-1

Refer figure-10 for Experiment-1.

1. Input Ingredients

We have given a set of ingredients to our model to find out perfect recipes for them. The model takes **['apples','bananas','eggs', 'tomato','lettuce', 'ham', 'creme', 'eggs', 'creamy', 'nicely']** ingredients array as input.

2. Predicted Recipes

Our **32** recipe files trained model recommends **['grilled onions','healthy asian apple soup', 'easy microwave tomato', 'lettuce soup', 'quick and dirty']** recipes that could be made from the ingredients.

Our **155** recipe files trained model recommends **['Awesome and Easy Creamy Corn Casserole', 'Apple Pie by Grandma Ople', 'lettuce soup', 'Janets Rich Banana Bread', 'Delicious Ham and Potato Soup', 'easy microwave tomato', 'quick and dirty']** recipes that could be made from the ingredients.

Our **501** recipe files trained model recommends **['lettuce soup', 'easy microwave tomato', 'Fruit Dip II', 'Banana Pancakes I', 'Baked Omelet Roll', 'Ham and Cheese Breakfast Quiche', 'healthy asian apple soup', 'Awesome and Easy Creamy Corn Casserole']** recipes that could be made from the ingredients.

3. Ground Truth

We can look at the following recipes and their ingredients we collected for training to get an idea of how precisely our model predicts the recipes.

grilled onions:['lime']

healthy asian apple soup:['apple', 'apples', 'pears', 'pearl', 'ham']

easy microwave tomato:['tomato']

lettuce soup:['lettuce', 'ham', 'creme', 'eggs', 'creamy', 'nicely']

quick and dirty:['coil']

Awesome and Easy Creamy Corn Casserole:['creamy', 'corn', 'butter', 'eggs', 'bread', 'creamed', 'cream']

Apple Pie by Grandma Ople:['apple', 'unsalted', 'butter', 'flour', 'water', 'sugar', 'apples',

'sliced', 'saucepan', 'boil', 'lattice']

Janet's Rich Banana Bread:['banana', 'bread', 'butter', 'sugar', 'eggs', 'vanilla', 'flour', 'soda', 'salt', 'cream', 'walnuts', 'bananas', 'sliced']

Delicious Ham and Potato Soup:['ham', 'potato', 'soup', 'diced', 'potatoes', 'celery', 'onion', 'water', 'chicken', 'salt', 'pepper', 'butter', 'flour', 'milk', 'boil', 'saucepan']

Fruit Dip II:['fruit', 'cream', 'cheese', 'creme']

Banana Pancakes I:['banana', 'pancakes', 'flour', 'sugar', 'powder', 'salt', 'egg', 'milk', 'vegetable', 'oil', 'bananas', 'oiled', 'pancake']

Baked Omelet Roll:['eggs', 'milk', 'flour', 'salt', 'pepper', 'cheese']

Ham and Cheese Breakfast Quiche:['ham', 'cheese', 'potatoes', 'butter', 'diced', 'eggs', 'cream']

5.2.2 Experiment-2

Refer figure-11 for Experiment-2.

1. Input Ingredients

We have given another set of ingredients to our model to find out perfect recipes for them. The model takes **['bean', 'chillies', 'choice', 'mint','tahiini', 'garbanzos', 'sesame']** ingredients array as input.

2. Predicted Recipes

Our **32** recipe files trained model recommends **['monastery style bean soup', 'hoomos']** recipes that could be made from the ingredients.

Our **155** recipe files trained model recommends **['monastery style bean soup','hoomos', 'Best Green Bean Casserole']** recipes that could be made from the ingredients.

Our **501** recipe files trained model recommends **['Extra Easy Hummus', 'monastery style bean soup', 'hoomos']** recipes that could be made from the ingredients.

3. Ground Truth

We can look at the following recipes and their ingredients we collected for training to get an idea of how precisely our model predicts the recipes for this experiment.

monastery style bean soup:['bean', 'chillies', 'choice', 'mint']

hoomos:['tahiini', 'garbanzos', 'sesame']

Best Green Bean Casserole:['bean', 'beans', 'cream', 'mushroom', 'soup', 'onions', 'cheese']
Extra Easy Hummus:['beans', 'garlic', 'cumin', 'salt', 'olive', 'oil', 'bean']

5.2.3 Inference from Experiments

We compared input ingredients with the ground truth ingredients of the predicted recipes and found out that in most cases the model gives good recommendations but it also gives bad recommendations in some cases. This is because low performance of the model which is due to less training data.
Also, since we are giving an array of ingredients as input the probability increases when 2 or more than 2 recipes matches. The model can't ignore this increased probability even if all the ingredients are not matching. Good news is it ignores the single matching ingredient recipes as it gets low probability.
It also depends on the input ingredients array. Sometimes the model gives bad output as it receives bad input.

6 Future Scope

1. The next step would be to collect more recipes from allrecipes.com and other websites as well and use them as part of the model training. This would help improve the model performance substantially.

2. The long-term goal of this project is to implement this onto a home-assistance device like Amazon Alexa or Google Home by using their speech processing module to take input from the user, parse the instructions to text and suggest recipes accordingly.

3. Technical enhancement as a next step include:

   (a) Taking into consideration quantities of ingredients available with the user and suggest recipes based on quantity as well.

   (b) If no recipe found with exact match of ingredients, build a module to suggest what more ingredients are required to accomplish the suggested recipe avoiding the allergic ingredients/recipes to the user if any.

7 Work Distribution

For the final report on the project, Dhivya Swaminathan wrote Abstract, Introduction (long term and short term goal), Previous Work, parts of Methodology (Introductory part and Data collection) (Pg 1-5) and Future Scope (Pg 17). Shilpa Singh wrote part of Methodology (Automated Annotation of Recipe Corpus and Neural Network for Ingredient Identification) (Pg 6-10). Sumeet Mishra wrote parts of Methodologies (Neural Network for Recipe Identification), Model Evaluation and Results (Pg 11-17).

For the project, Dhivya Swaminathan built the Recipe Crawler using BeautifulSoup in python and crawled all recipes to build the recipe corpus of 469 recipes from allrecipes website. Dhivya Swaminathan, Shilpa Singh and Sumeet Mishra worked towards annotating the Recipes as part of building the training set. Neural network building to find ingredients from recipe text, training and evaluation of the model was carried out by Shilpa Singh. Neural network building to recommend recipes given a list of ingredients, training and evaluation of the model, results and Comparison of models and results based on different number of recipe files taken as part of training set was carried out by Sumeet Mishra.

References

[1] *Large Scale Multi-label Text Classification with Semantic Word Vectors, Mark J.Berger, Stanford University, CA*

[2] "How AllRecipes.com Recipe Site". Retrieved 2011-10-01. *https://www.allrecipes.com/recipes/?prop24=PN_2.0.0$_T$N.Recipes*

[3] Beautiful Soup Documentation, Copyright 2004-2015, Leonard Richardson *https://www.crummy.com/software/BeautifulSoup/bs4/doc/*

[4] Artificial intelligence suggests recipes based on food photos, Adam Conner-Simons and Rachel Gordon, CSAIL July 20, 2017 *http://news.mit.edu/2017/artificial-intelligence-suggests-recipes-based-on-food-photos-0720*

[5] Deep-based Ingredient Recognition for Cooking Recipe Retrieval, Jingjing Chen

and Chong-wah Ngo, City University of Hong Kong, Hong Kong, China *https://dl.acm.org/citation.cfm?id=2964315*

[6] Images Recipes: Retrieval in the cooking context, Micael Carvalho, Remi Cadene, David Picard, Laure Soulier, Matthieu Cord, Sorbonne Universite – Paris, France , ETIS, UMR 8051 – Universite Paris Seine, Universit e Cergy-Pontoise, ENSEA, CNRS, May 2018 *https://arxiv.org/pdf/1805.00900.pdf*

[7] ELMo Deep contextualized word representations, Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer. NAACL 2018. *https://allennlp.org/elmo*

[8] Gentle Introduction to the Adam Optimization Algorithm for Deep Learning, Jason Brownlee on July 3, 2017, Deep Learning Performance *https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/*

[9] Keras: The Python Deep Learning library *https://keras.io/examples/imdb_bidirectional_lstm/*

[10] Wikipedia, open source edit *https://en.wikipedia.org/wiki/Precision_and_recall*

[11] Wikipedia, open source edit *https://en.wikipedia.org/wiki/F1_score*

[12] Google Machine Learning Crash course, Classification: Accuracy, content of this page is licensed under the Creative Commons Attribution 4.0 License, and code samples are licensed under the Apache 2.0 License. *https://developers.google.com/machine-learning/crash-course/classification/accuracy*

[13] Machine Learning Mastery: How to develop a word level neural language model in keras. *https://machinelearningmastery.com/how-to-develop-a-word-level-neural-language-model-in-keras/*

8 Appendix

```
<Title>
Recipe By : <author name>
Ingredients : <list of ingredients>
Directions : <steps to cook the recipe>
```

Figure 1: Recipe format

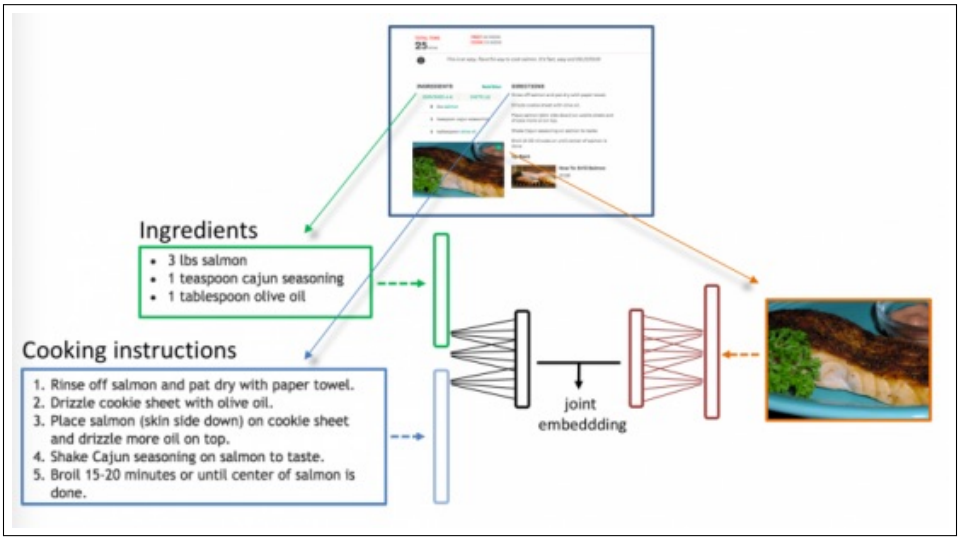

Figure 2: Pic2Recipe Architecture

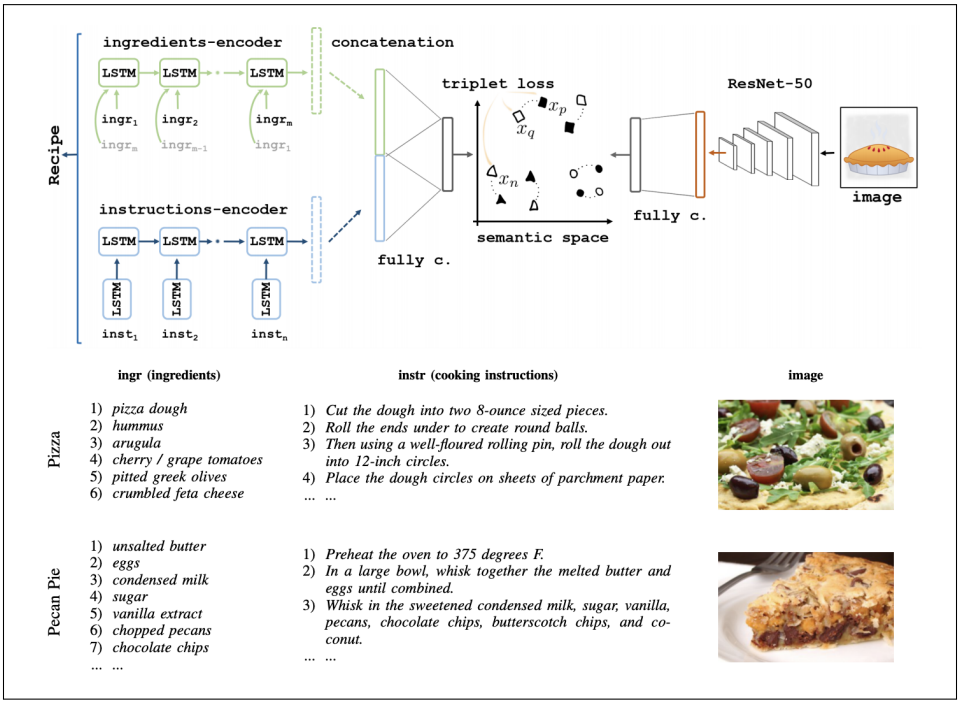

Figure 3: Multi-modal retrieval learning objective Architecture

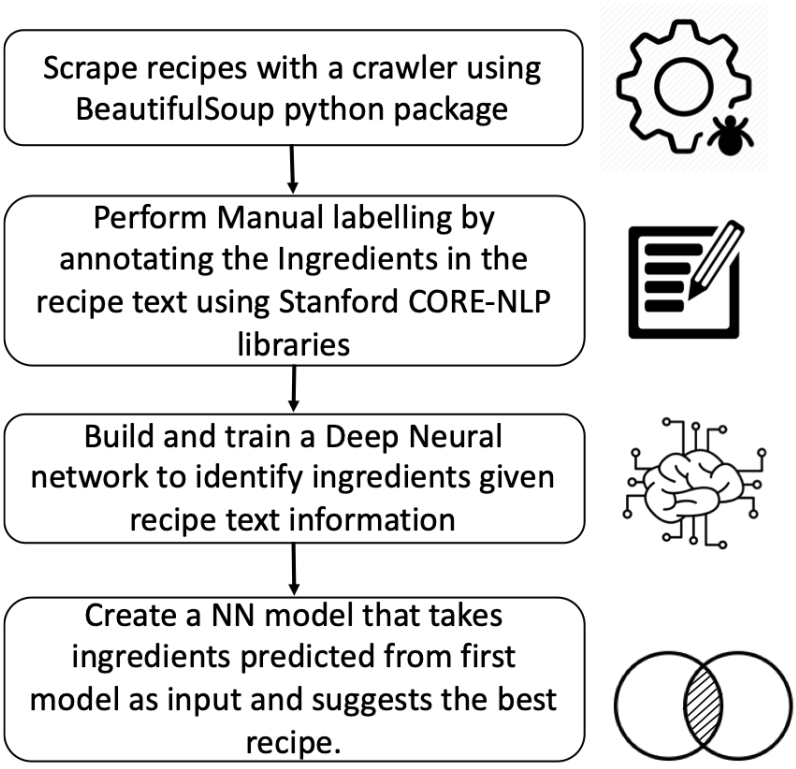

Figure 4: Recipe Recommendation Architecture

```
def get_data_for_page(page):
    response = requests.get(page ,headers={'User-Agent': 'Mozilla/5.0 '})
    soup = BeautifulSoup(response.content,"html.parser")
    contents = soup.findAll('article', {'class' : 'fixed-recipe-card'})
    recipes_links = []
    for content in contents:
        recipe_info = content.find('div', {'class' : 'fixed-recipe-card__info'})
        link = recipe_info.find('a', href = True)['href']
        recipes_links.append(link)
    return recipes_links
```

Figure 5: Recipe Link Extraction Code

```
class Recipe:
    def __init__(self, title, author, ingredients, directions):
        self.title = title
        self.author = author
        self.ingredients = ingredients
        self.directions = directions
        self.recipe_file = ""

    def consolidate_to_file(self):
        self.recipe_file += self.title + "\n"
        self.recipe_file += "Recipe by : " + self.author + "\n"
        self.recipe_file += "Ingredients : "
        for ingredient in self.ingredients:
            self.recipe_file += ingredient
        self.recipe_file += "\n"
        self.recipe_file += "Directions : " + self.directions

    def write_to_file(self):
        filename = recipe_path + self.title + ".txt"
        file = open(filename,"w")
        file.write(self.recipe_file)
```

Figure 6: Recipe Class

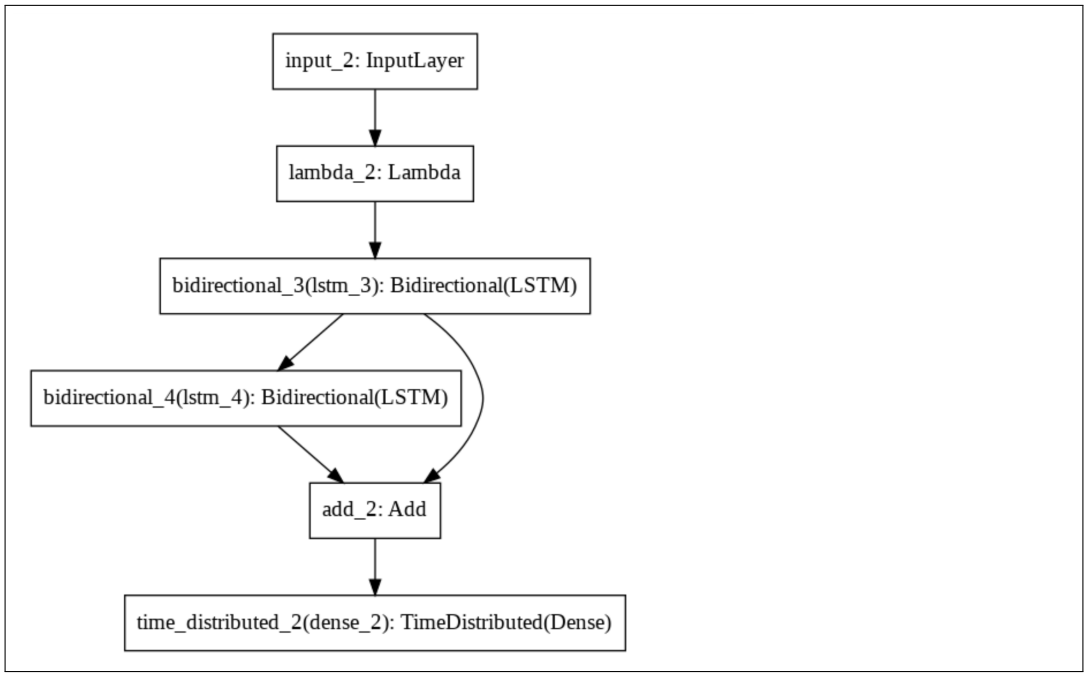

Figure 7: Model for Ingredients Identification

Neural Network Model	Work	Recipe files trained	Precision	Recall	F1-Score
ELMo (Bi-LSTM)	Predict ingredients from raw text	469	95%	93%	94%

Figure 8: Ingredients Identification Model Results

Neural Network Model	Work	Recipe files trained	Train Accuracy	Test Accuracy
Text Classification model	Predict Recipes from Ingredients	32	87%	42%
Text Classification model	Predict Recipes from Ingredients	155	92%	64%
Text Classification model	Predict Recipes from Ingredients	501	89%	49%

Figure 9: Recipe Recommendation Model Results

Input Ingredints-1	
['apples','bananas','eggs', 'tomato','lettuce', 'ham', 'creme', 'eggs', 'creamy', 'nicely']	
Recipe files trained	**Predicted Recipes from our model**
32	['grilled onions','healthy asian apple soup', 'easy microwave tomato', 'lettuce soup', 'quick and dirty']
155	['Awesome and Easy Creamy Corn Casserole', 'Apple Pie by Grandma Ople', 'lettuce soup', 'Janets Rich Banana Bread', 'Delicious Ham and Potato Soup', 'easy microwave tomato', 'quick and dirty']
501	['lettuce soup', 'easy microwave tomato', 'Fruit Dip II', 'Banana Pancakes I', 'Baked Omelet Roll', 'Ham and Cheese Breakfast Quiche', 'healthy asian apple soup', 'Awesome and Easy Creamy Corn Casserole']

Figure 10: Recipe Recommendation Results with Ingredients-1

Input ingredients-2	
['bean', 'chillies', 'choice', 'mint','tahiini', 'garbanzos', 'sesame']	
Recipe files trained	**Predicted Recipes from our model**
32	['monastery style bean soup', 'hoomos']
155	['monastery style bean soup','hoomos', 'Best Green Bean Casserole']
501	['Extra Easy Hummus', 'monastery style bean soup', 'hoomos']

Figure 11: Recipe Recommendation Results with Ingredients-2