Ingredient Identification in Food Recipes

Dhivya Swaminathan, Shilpa Singh, Sumeet Mishra

December 17, 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 are trying to build 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.

1 Introduction

Our project has a long term and short term goal and due to time constraints for this course, we intend to accomplish the short term goal for the course 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 the that 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 Existing Literature

The following is the compilation of the existing works in this arena. 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. (3)

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 experiment on a large Chinese food dataset of food images with complex appearances for this project. (4)

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. (5)

3 Architecture and Implementation

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. The architecture is as outlined in Figure 4. The main steps to the architecture is as listed below:

1. **Data Collection**: We collected around 435 recipes from the Allrecipes website, its sub-categories and all its pages using a web-crawler using the BeautifulSoup package in python.

2. **Reference data creation**: From these above said scraped recipes, a set of unique ingredients was obtained and a reference data was created for lookup to be used for the annotation task.

3. **Corpus Annotation**: 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.

4. **Pre-processing**: The training data was then prepared from these annotated files using pre-processing techniques.

5. **LSTM with Elmo Model**: For ingredient identification in recipes, models of LSTM and LSTM with Elmo Embeddings were employed that gave good accuracy.

6. **Recipe Prediction**: A text classification model was then developed to predict recipes for the input ingredients.

3.1 Data Collection

We collected around 435 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/

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 (2) 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 62.

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 435 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 Model Building for Ingredient Identification

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.

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. 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 degrada-

tion 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.

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.Then we scanned the length of each list in this collection to find the longest sentence in the corpus.It was 37 ,so we padded all the input and output sequence till length of 38. This prepared a fixed length sequence input data for training and a 38 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 38 dimensional word vector and output Y was a 38 dimensional 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.

3.4 Text Classification Model

1. It takes ingredients as input and recipe name as ground truth for training.
2. It learns to predict recipes from given set of ingredients.
3. It gives a 87% train accuracy and 42% test accuracy while trained with only 32 recipes.

4 Model Evaluation

Neural Network Model	Work	Recipe files trained	Train Accuracy	Test Accuracy
ELMo(Bi-LSTM)	Predict ingredients from raw text	32	99 %	99%
Text Classification	Predict recipes from ingredients	32	87 %	42%

The low accuracy of the classification model is due to less number of training data.We trained our model with only 32 recipe files and the performance will increase after training with all the 500 recipes we have collected from All recipe website.

5 Future Scope

1. The next step would be to train our model with all the recipes we scraped and annotated and observe model performance.
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 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.

References

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

[2] *https://www.crummy.com/software/BeautifulSoup/bs4/doc/*

[3] *http://news.mit.edu/2017/artificial-intelligence-suggests-recipes-based-on-food-photos-0720*

[4] 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*

[5] 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*

6 Appendix

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

Figure 1: Recipe format

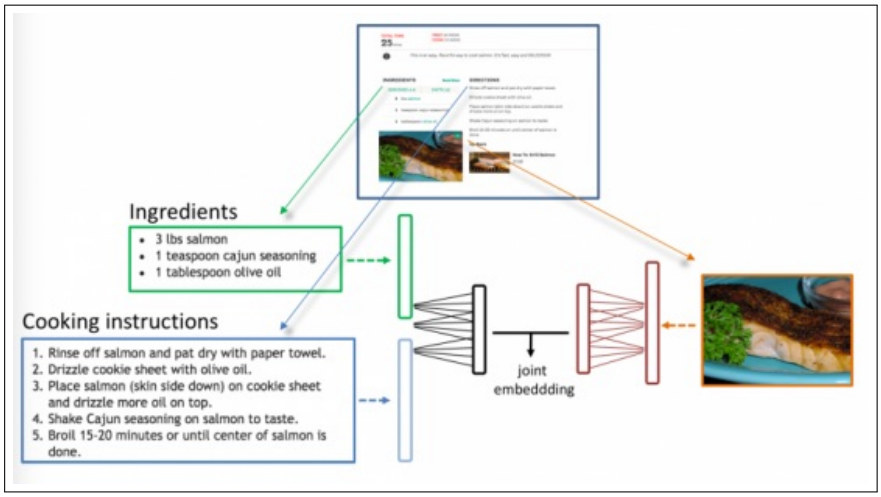

Figure 2: Pic2Recipe Architecture

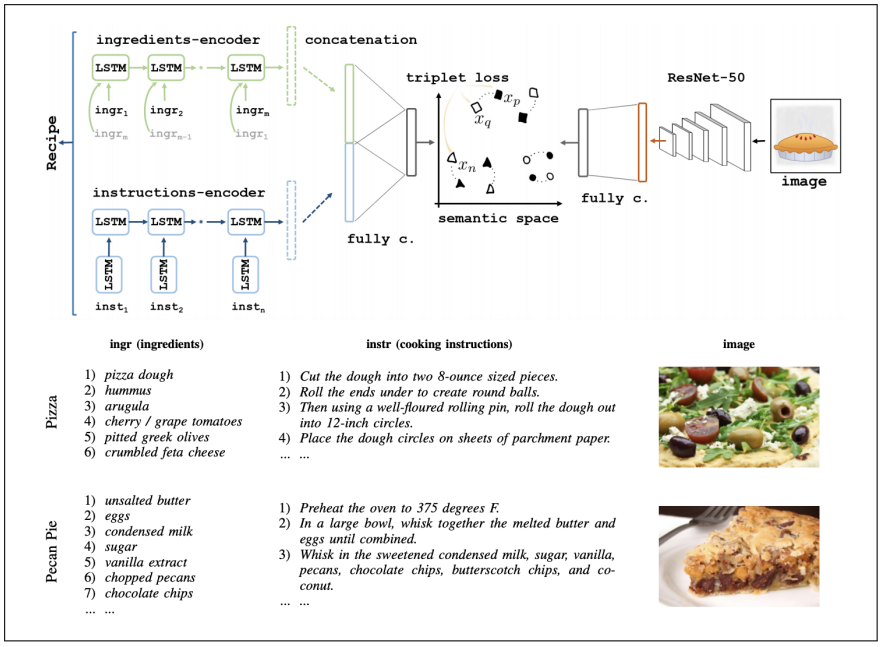

Figure 3: Multi-modal retrieval learning objective Architecture

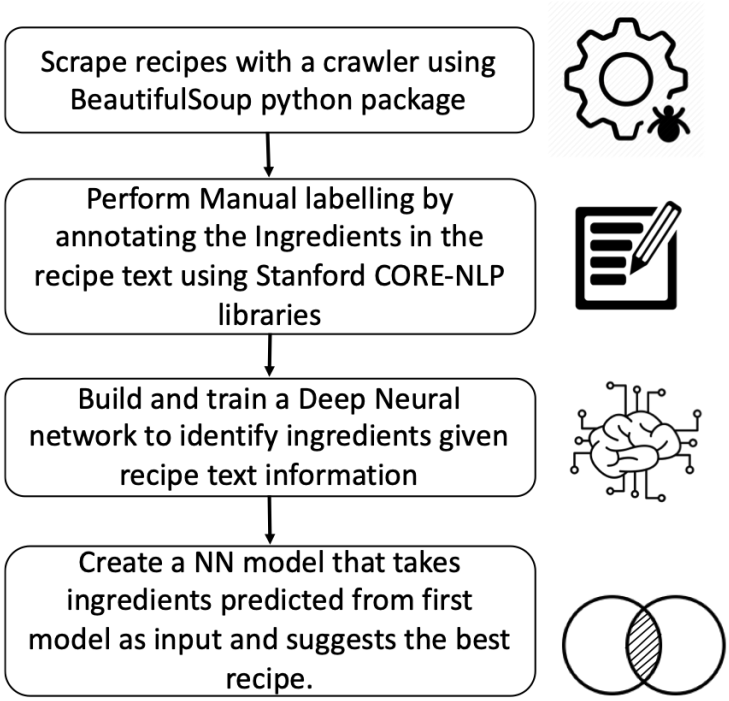

Figure 4: Recipe Recommendation Architecture

```
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