Cooking Recipe and Food Image Retrieval

Borna Ghotbi
UBC CS Department
bghotbi@cs.ubc.ca

Farnoosh Javadi
UBC CS department
fjavadi@cs.ubc.ca

Mona Fadaviardakani
UBC CS department
mfadavi@cs.ubc.ca

Abstract

In our project, we introduce a model that aims to learn embeddings of cooking recipes and food images that yields notable results on image-recipe captioning task. Food is one of the most fundamental issus for humans, since it can affect every other aspects of our lives, like health, sleeping habits, feelings and culture. Therefore, we think automization of recipe-image retrieval can be a beneficial task that makes cooking food easier. We propose a complex combined neural network that learns embeddings for ingredients, cooking instructions and food images through different joint models. This domain is fairly new and we hope these embeddings provide a basis for further works on not only recipe-image retrieval task but also for other domains like computer vision.

1 Introduction

Nowadays, there are tons of different recipes and food images on the internet that we see everyday and may ask ourselves how I can cook this food without even knowing the name of that. Or otherwise, asking how this food is going to look like if I follow its recipe and cook it. It is obviously interesting and advantageous to have a system that automatizes answering these kind of questions. Beyond its application solely, this system can be applied to the food images shared on social medias to get an insight into the relation between food's preparation and public's health (Garimella et al., 2016) or cultural heritage (Mejova et al., 2016). This task, because of its complexity , needs a rich huge dataset as well as a complex model to achieve good results. There are some medium-scaled datasets like Food-101 (Bossard et al., 2014), DeepFood(Liu et al., 2016) and Im2Calories(Meyers et al., 2015) that can be used for this task but they seem limited either in generality or size. Hence, in our project we use Recipe1M dataset (Marin et al., 2018a) because of its large size and generality (containing both cooking instructions and ingredients list). In this work, we present an image-recipe retrieval task that one of is utilities is to solve the practical problem of creation a dish that is seen but not described. For this purpose, we have trained a multi-modal neural model which jointly learns embeddings for food images and recipes. The performance of our project is evaluated against baseline, and also by ourselves to assess and interpret results better.
In the following sections, at first we elaborate Recipe1M dataset's details that we used, then explain our proposed model. In the following we talk about the experiments and results of our project.

2 Related Works

Herranz et al. (L. Herranz and Jiang, 2018), proposed an extended multimodal framework that not only consider the image and recipes but also nutritional information, geolocation and time, restaurant menus and food styles.

Min et al. (W. Min and Mei, 2017) present a multi-attribute theme modeling (MATM) approach is also considering different food properties such as the type of ingredient, flavour and the style of the food. A multimodal embedding is learned which is using a joint space between the different food attributes and the corresponding

Data Section	#Recipes	#Images
Train	720,639	619,508
Validation	155,036	133,860
Test	154,045	134,338
Total	1,029,720	887,706

Table 1: Number of samples in different data sections

food image to learn the embedding.

Chen et al. (J.-j. Chen and Chua, 2017) focuses on the appearance of food and the factors that affects on them to address the recipe2image task. For example, they try to consider such attributes like the manner of cutting and manner of cooking ingredients in forming the foods appearance. By using a food image, they attempt to predict ingredient, cutting and cooking attributes, and use these predictions to help retrieve the correct corresponding recipe. In our model, we attempt to retrieve the recipe directly, without first predicting attributes like ingredients, cutting and cooking attributes, separately and only return the food images.

Engilberge et al. (M. Engilberge and Cord, 2018) focus on image captioning.They use neural networks to create embeddings for each caption, and retrieve the embedding which is most closely matched the embedding of the original image.The difference of their work in comparison with us is using captions of images instead of recipe as their text data. Therefore, we have to handle the problem of both ingredients and instrcuctions, while they are more limited on their data.

3 Dataset

Recipe1M dataset includes over 1 million recipes and 800k images, which were collected by crawling more than two dozen popular cooking websites. The dataset already is free of non-ASCII characters and excessive whitespaces. Each recipe contains some basic featuers such as id, title, category and link, but three main components including cooking instructions, ingredients list, and images. Dataset is splitted into three sections: train, validation, and test by 70, 15, and 15 percentages. Table1 shows number of samples in each section.

- Cooking instructions: Each recipe contains at most 20 lines of instructions, and 10 lines in average . For example "Mix eggs and suger together" is an instruction line. Since we feed each instruction line, which is a meaningful sentence that its words' order matters, into an auto-encoder and LSTM, we don't do stopwords elimination or further preprocessing tasks such as normalization and stemming.

- Ingredients: There are 16k unique ingredients in this dataset, and each recipe includes nine ingredients in average. Each ingredient item is a sentence like "two cups of milk". As a preprocessing task we get rid of everything except from the actual name of the ingredient, that is "milk" in mentioned example. For representing a single ingredient name we use pretrained word2vec (Mikolov et al.) model, which is trained over Google News dataset (GoogleCode, 2013) and outputs a 300 dimension vector per word.

- Images: Images are provided as RGB in JPEG format. About half of recipes have linked images, which we just work on them.

4 Model

In this section we introduce our joint embedding learner model. This model mainly consists of three components: Instructions, ingredients, and images embedding generators. Each model component is responsible to extract an embedding from the given inputs. We learn a common embedding space utilizing this model as sketched in figure 1. We decided to use a uni-directional LSTM model for the instructions inputs, a graph convolutional Network (GCN)(Thomas N Kipf, 2017) for the ingredients, and a Resnet model for the images. The following sections will focus on each of these sub-models.

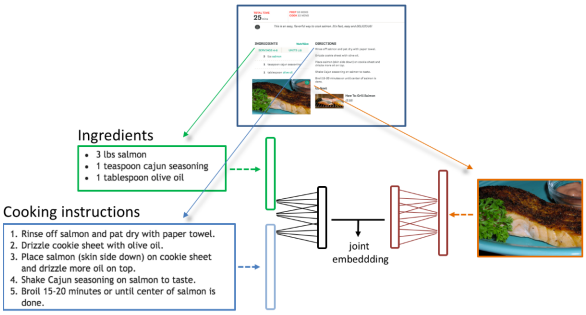

Figure 1: Model Overview

4.1 Graph Convolutional Network

We decided to use a Graph convolutional network (GCN) (Thomas N Kipf, 2017) for the ingredient embedding generation. The intuition behind this is our ingredient set properties. Each recipe has a set of ingredients which the ordering is not important. Therefore, it doesnt make sense to use sequential models like RNNs. Instead, GCNs can learn the relational properties between each two ingredients. In order to do so, we decided to use a fully connected GCN.

GCNs can be designed with various architectures, but they mostly follow a common pattern for their design. We can refer to these models as GCNs. The main objective for these graphs is to learn a function of features on the input. These graphs represented as g = (v,e) consist of two main components: 1) A feature representative for each node i. We summarize this feature in a NxD matrix H. 2) An adjacency matrix representing the relation between each of the nodes, showing the structure of the graph. This matrix is of size NxN. The graph generates a node-level output. In case we need a representation of the whole graph, a pooling operation can model our output. The model can consist of several layers. Each of these layers will represent a time-stamp in graph dataflow. Each layer can be represented as the following non-linear function:

$$H^{(l+1)} = f(H^l, A) \tag{1}$$

Here H(0) is the input and H(L) is the output Z. L represents number of layers. In our work we propose using GCNs instead of a bidirectional LSTM (reference). One of the important decisions to make is the choice of node initialization. After a preprocessing step on the raw ingredients, we feed each ingredient to a Word2Vec model and extract a feature representation. We used these W2V vectors to initialize our nodes and generate the graph.

Now we take a further step into the information propagation process. The graph follows a neighborhood aggregation strategy where we update the values of our W matrix on each iteration. These k iterations will help the model to learn the structural information of the nodes. In a more general way, the l-th layer of a GCN is (Xu, 2018):

$$a_v^l = AGGREGATE^l(h_u^{l-1} : u \in N) \tag{2}$$

$$h_v^l = COMBINE^l(h_v^{l-1}, a_v^l) \tag{3}$$

where N is a set of nodes adjacent to node v. In our model AGGREGATE has been formulated as:

$$a_v^l = MEAN(RELU(W.h_u^{l-1}), \forall u \in N(v)) \tag{4}$$

where W is our learnable matrix. The COMBINE step The COMBINE step is a concatenation followed by a linear mapping

4.2 Uni-directional LSTM model

Each recipe has the list of cooking instructions. We need to extract vectors from the instructions in a way that instruction sentences that share semantic and syntactic properties are mapped into similar vector representation (I. Sutskever and Le, 2014). LSTM is a good choice for extracting instruction's representation because the importance of the order in the instructions. However,since the instructions are quite lengthy (average 208 words), It is not proper to feed them through single LSTM model. Instead, we use a two stage LSTM model. As the first step, the auto-encoder is used to extract the representation of each single instruction. For this matter, we encode every instruction line of a recipe and use that encoding as context when decoding the next and previous instructions. For this purpose we add start and end to the instructions of each recipe and store the final output of a decoder as the skip-instruction vector as it is shown in figure 3. Each skip-instruction vector has a length of 1024 features. At the next stage, We feed all the skip-instruction vectors of each recipe to the one layer LSTM model with the same hidden-size and 20 LSTM units (since each recipe has at most 20 instructions) to encode the sequence of the sequence of instructions. The output of the LSTM model is the representation of all instructions and use as the instruction embedding in our joint embedding learner model.

4.3 Image Model

We adopt the Resnet50 model (He et al., 2016) for extracting our image features. Resnet model which stands for deep residual network suggests special architecture for nueral network layers that enables of training deeper networks. In these kind of networks each layer instead of just being connected to its post and previous layers are connected with some upper and lower layers as well. This architecture enables the whole network to learn deeper (Szegedy et al., 2017). The deep residual networks have been proved that has a

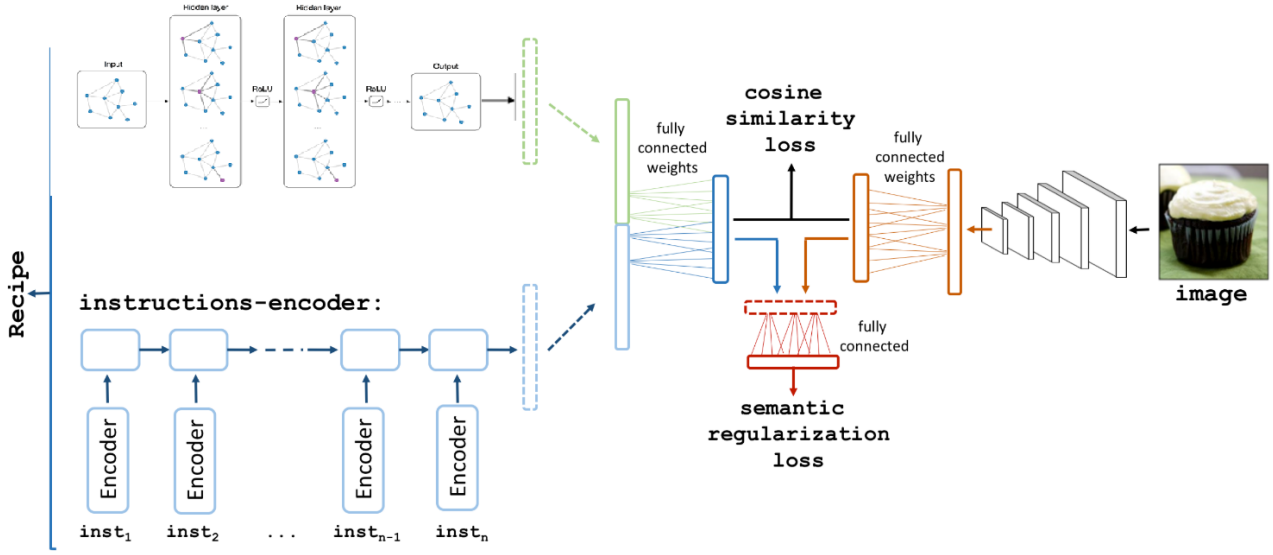

Figure 2: Joint Neural Embedding Model

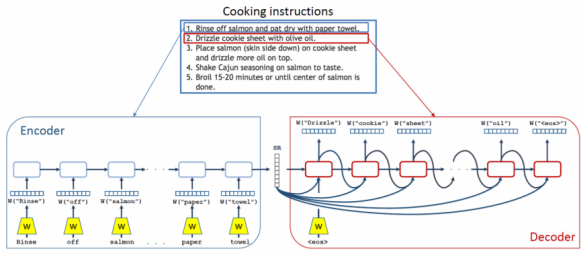

Figure 3: Sequence 2 Sequence Model

great success on variety of benchmarks especially on image processing tasks. We also tested VGG-16 (Ren et al., 2015) pretraiend model for extracting images features which yields not better results. That is why we decided to use resnet model as an image feature extractor for our project. We used Resnet50 pretrained model in pytorch by removing the last softmax layer and adding a linear and tanh layers which project 2048 features of Resnet to our desired of 1024 features. The output of the resnet model is the vector with length of 1024 features which is used in our joint embedding learner model.

4.4 Joint Embedding Learner Model

Our goal in this model is to learn transformation in a way that makes embedding for a given "recipe-image" pair. As described in previous subsections, we extract embeddings for ingredients by using GCN, embeddings for instructions by using two stages of auto encoder and LSTM, and embeddings of images by using Resnet. In order to build recipe representations, first we concatenate both embeddings of ingredients and instructions then apply a fully connected layer to set the size of the output to 1024, then project it into our joint embedding space. This 1024 dimension vector is considered as an embedding of a recipe. Image representations are also projected to our new space,after passing respectively through the resnet , linear and tanh layers . Our goal is to define The loss function which is getting maximized for matching pairs of images and recipes, and minimized for unmatched pairs. In order to address that, we generate the training data in a way to force 80 percent of image-recipe pairs are mismatched and only 20 percent of them are matched. We use target variable as our matching indicator which is +1 for matched pairs and is -1 for unmatched ones. Our loss is cosine similarity between a recipe embedding, and its corresponding images embedding. During the training we try to learn the weights and biases associated with both recipe embeddings and image embeddings . A schema of our model is shown in figure 2.

Formally, assume that we are given a set of the recipe-image pairs,(R_k, v_k) in which R_k is the k^{th} recipe and v_k is its associated image vector. Let $R_k = (s_k \oplus g_k)$, where s_k vector is the respesentor of all the ingredients in a recipe , g_k vector is the respesentor of all the cooking instructions in a recipe , and $\oplus$ corresponds to concatenation

operand. Then we map the recipe and image representations into the joint embedding space by applying linear functions as: $\phi^R = W^R R_k + b^R$ and $\phi^v = W^v v_k + b^v$. W^R and W^v are embedding matrices which are learned during training. Finally, the joint model is trained with matched and mismatched recipe-image pairs(ϕ^R, ϕ^v), using cosine similarity loss as follows:

$$\text{Loss}((\phi^R, \phi^v), y) = \begin{cases} 1 - cos(\phi^R, \phi^v) & if y = 1 \\ max(0, cos(\phi^R, \phi^v) - \alpha) & if y = -1 \end{cases} \quad (5)$$

where cos(.) is the normalized cosine similarity and α is the margin.

5 Experiments

5.1 Implementation details

All the neural networks models are implemented by using PyTorch framework. The learning rate which is used is 0.1 and we use the batch size of 6. In order to optimize the cosine loss, we pick the positive image-recipe pairs with 20 percent probability and randomly select negative image pairs with 80 percent probability. We store different parts of data including image skip-instruction vectors, ingredients, image ids, recipe-ids in LMDB (Recipe1M, 2017) format. Using LMDB file enables us to handle the memory issue with the huge dataset.

5.2 Evaluation

We evaluate the result of our work in the rec2img task. By given a test food recipe, the rec2img task aims to use cosine similarity in the common space for ranking the relevant images and return the most similar ones. We report median rank (MedR) and recall rate at top K (R@K) for all the retrievals. In this task R@K indicates the percentage of all image queries where the corresponding recipe is retrieved in the top k. We took (Marin et al., 2018b) as our baseline and compared the results of our project with it, which are reported in Table 2. The baseline uses Recipe1m dataset for learning recipe-image embeddings as same as our work that makes the comparisons more reliable and fair. As we can see in Table 2 we underperformed the baseline that we think it is because of the intrinsic complexity of task itself and also our limitations in time and hardware. Running the whole project for 10 epochs over the whole train set have taken about 4 days on our systems that didn't let us to do hyperparemeter tuning in a period of time that we had. So for training our model we use the default settings in pytorch. The regularization hyperparameter is set as $\lambda = 0.02$. We believe that hyperparameter tuning by using validation data will improve the results remarkably. In addition, the size of images and test dataset were too large to be possible for us to test our model multiple times and report the average value of each evaluation metrics. All the reported results are for one run of the project using the whole train and test data.

In figure 4 we have shown retrieved images for 3 test recipe samples comparing to their original images. We think that accuracy and precision metrics which are important features in all retrieval tasks can't fairly evaluate our task. Bacuase there are 800k images on the dataset which some of them are really similar and indicates one food, however there is just one true image in the test data. For popular recipes like "Chocolate cake" there are images more than average which retrieving each one of them is a success for the system, however there is just one true label for "chocolate cake" in the test data. Similarly, it is clear in figure 4 that the system did a good job on identifying "Mango Lassi" and "Pound cake", however none of the retrieved images are not as same as the original image. This is another important reason that R@k metric values seem low.

6 Conclusion

In this paper, we benefit form the large databases of images and recipes in order to address the recipe2image task. Our method uses three different neural network models to extract the embedding for the 3 different components and project them in the joint space. More generally, our method can be extended by considering other food features like nutritional information,food categories(e.g. appetizer, dessert, main dish), food style, type of ingredients, cooking manner and take use of them in the model for learning embeddings, in order to get more accurate result. The learned recipe embeddings can be a basis for so many other tasks such as flavour analysis, image2recipe retrieval , identifying style of the recipe, food classification and many other food-recipe-image related tasks. This task is fairly new and we hope defined embeddings doors of food and recipe understanding.

tabularllll heightModel
medR
R@1
R@5
R@10

Baseline
5.2
0.24
0.51
0.65
Our model$_{(usingResnet-50)}$
17.1
0.09
0.27
0.35
Our model$_{(usingVGG-16)}$
23.7
0.05
0.18
0.29

Table 2: Results of our method for rec2img task in comparison with baselines

References

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. 2014. Food-101–mining discriminative components with random forests. In *European Conference on Computer Vision*. Springer, pages 446–461.

Venkata Rama Kiran Garimella, Abdulrahman Alfayad, and Ingmar Weber. 2016. Social media image analysis for public health. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, pages 5543–5547.

GoogleCode. 2013. Googlenews-vectors-negative300.bin.gz. https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit. Accessed: 2018-09-16.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 770–778.

O. Vinyals I. Sutskever and Q. V. Le. 2014. Sequence to sequence learning with neural networks. *NIPs* .

C.-W. Ngo J.-j. Chen and T.-S. Chua. 2017. Cross-modal recipe retrieval with rich food attributes,. *in Proceedings of the 2017 ACM on Multimedia Conference, ser. MM 17. New York, NY, USA: ACM, 2017* .

W. Min L. Herranz and S. Jiang. 2018. Food recognition and recipe analysis: integrating visual content, context and external knowledge. *L. Herranz, W. Min, and S. Jiang* .

Chang Liu, Yu Cao, Yan Luo, Guanling Chen, Vinod Vokkarane, and Yunsheng Ma. 2016. Deepfood: Deep learning-based food image recognition for computer-aided dietary assessment. In *International Conference on Smart Homes and Health Telematics*. Springer, pages 37–48.

L. Chevallier P. Perez M. Engilberge and M. Cord. 2018. finding beans in burgers: Deep semantic-visual embedding with localization. *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognitio* .

Javier Marin, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. 2018a. Recipe1m: A dataset for learning cross-modal embeddings for cooking recipes and food images. *arXiv preprint arXiv:1810.06553* .

Javier Marin, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. 2018b. Recipe1m: A dataset for learning cross-modal embeddings for cooking recipes and food images. *arXiv preprint arXiv:1810.06553* .

Yelena Mejova, Sofiane Abbar, and Hamed Haddadi. 2016. Fetishizing food in digital age:# foodporn around the world. In *ICWSM*. pages 250–258.

Figure 4: Retrieved images for the query ingredients and instructions, in comparison with the original images

Austin Meyers, Nick Johnston, Vivek Rathod, Anoop Korattikara, Alex Gorban, Nathan Silberman, Sergio Guadarrama, George Papandreou, Jonathan Huang, and Kevin P Murphy. 2015. Im2calories: towards an automated mobile vision food diary. In *Proceedings of the IEEE International Conference on Computer Vision*. pages 1233–1241.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. ???? In *NIPS*.

Recipe1M. 2017. Recipe1m dataset in lmdb format:. http://im2recipe.csail.mit.edu/dataset/download/. Accessed: 2018-11-10.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. pages 91–99.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*. volume 4, page 12.

Max Welling Thomas N Kipf. 2017. Semi-supervised classification with graph convolutional networks .

S. Jiang S. Wang J. Sang W. Min and S. Mei. 2017. A delecious recipe: analysis framework for exploring multi-modal recipes with various attributes. *n Proceedings of the 2017 ACM on Multimedia Conference, ser. MM 17. New York, NY, USA: ACM, 2017* .

Keyulu Xu. 2018. How powerful are graph neural networks? .

A Division of Labour

The workload was divided as in shown in Table**??** among team members in a good balance:

- **Borna:** Data loading , GCN model , Documentation
- **Farnoosh:** LSTM model , Resnet model , Documentation
- **Mona:** Data loading , LSTM model , Documentation