PROJECT REPORT

Inverse Cooking:
Recipe Generation from Food Images

BY

SNOOP&DOGG

GITANK SAGAR- 11940430

MADHUR BHATTAD- 11940660

Abstract

We introduce an inverse cooking system that recreates cooking recipes of given food images. The system is trained on the large-scale Recipe1M dataset and is able to produce more compelling recipes than retrieval-based approaches according to human judgment. The goal of the model was to outperform the existing retrieval task. Where the image was retrieved from a fixed data set on the basis of similarity score. Now we are looking to upgrade our model for predicting more than one recipe from the ingredients extracted from the given image.

Previous Work

Our main focus was **conditional text generation** based on

1. image to generate ingredients
2. Image and ingredients both to generate recipes.

All the preprocessing steps were taken care to train the best model and the minimal dataset for the best result.

For our purpose transformers are chosen instead of RNN/LSTM because processing has to be performed sequentially in case of RNN and LSTM, where transformers can do processing in parallel. Thus training can be done fast by using transformers.

Introduction

The result of our model out performed the image-to-recipe as a retrieval task, where a recipe is retrieved from a fixed dataset based on the image similarity score. Our system solves the problem by generating a cooking recipe containing a title, ingredients and cooking instructions directly from the image.

We are looking for a new challenge which is generating new recipes from the ingredients present in the recipe of the given image. As well as we are taking care which ingredients can be added also which ingredients can be removed from the recipe.

Problem Definition

Automated recognition of a photographed cooking dish, with its recipe as output. This takes an image as input and outputs the title, ingredients and instructions to make the dish.

From those ingredients generating new recipes. Also adding more ingredients as well as removing some ingredients.

Objective

This problem is focused on changing the output to make a new outcome as a new recipe. For this we take the pretrained model of ResNet-18 for converting image to embeddings

and a transformer with self attention for ingredient prediction followed by instructions/recipe generation from both image and ingredients. We are then applying greedy search and beam search algorithms for generation of new recipes.

Technology Used

We are using the previous pre-trained model as there was no need to retrain 120 gb of data because we are able to get our desired output by using greedy search and beam search in the code output section.

Problems Faced

We have used the pretrained model. So no such big problem was faced. We had to just run through the code a couple of times. It took some time, that's all the problem we faced.

Datasets Used

Dataset used is Recipe1M composed of 1,029,720 recipes. The dataset contains 7,20,639 training, 1,55,036 validation and 1,54,045 test recipes, containing a title, a list of ingredients, a list of cooking instructions and optionally an image.

Link to dataset: http://im2recipe.csail.mit.edu/dataset/login/

Models Used

Model used for converting images to embedding is a pretrained CNN model ResNet-50.

For obtaining the ingredient embedding, an ingredient decoder which is a transformer is used to decode ingredients from the image embeddings followed by mapping each ingredient to a fixed size vector.

The transformer consists of 4 blocks with 2 multi-head attention. So the dimensionality of embeddings is 512.

For generating recipes, an instruction decoder which is again a transformer, is used,

with 16 blocks and 8 multi-head attention.

The transformer first takes the input text and makes a positional encoding, the reason for this will be cleared at the end of this section.

Each block in a transformer consists of 2 attention layers. First one self-attention layer applies self attention on previously generated output, followed by an add and normalization layer, followed by an attention layer which applies model conditioning to output of self-attention layer.

After all the blocks there is a linear layer to convert the output of the last block into a vector of dimension 512, followed by a softmax layer to introduce non linearity and calculate the probability of each word in vocabulary as the next word in the output.

Transformers are used because other methods of doing the same thing perform the training sequentially, i.e. one word at a time. Transformers can do better by treating every word independently and hence doing parallel execution. This is the reason to store the positional encoding of the input text, so that we know the relative position of 1 word w.r.t to others

Implementation

Following are the steps included in the implementation:

1) First step is to build ingredients and instruction vocabulary. This is created by considering each word in the input instructions and ingredients and then applying some preprocessing to clean this vocabulary. Steps to cleaning and reducing vocabulary size are:
    a) All recipes without images are removed resulting in 2,52,547 training, 54,255 validation and 54,506 test samples.
    b) All related ingredients (like many types of cheese) are merged to just form 1 ingredient (cheese). So the total number of unique ingredients was reduced to 1488. These vocabulary files are then stored, and used in further steps.
2) Using ResNet-50 image embeddings are obtained
3) Then the ingredient decoder is used to obtain the ingredients embeddings.
4) The instruction decoder then runs conditioned by both the image as well as ingredient embeddings.

A transformer can only condition on 1 factor, so different methods must be applied to condition on multiple modalities. Some of them are:

1. **Independent** : 1 attention layer attends image embeddings, and another layer attends ingredients embeddings. Both the outputs are then combined via summation.
2. **Sequential :** First layer attends 1 factor and its output is attended by the next layer. In this case there are 2 possibilities:
    a. Attending ingredients first followed by image
    b. Attending image first followed by ingredients
3. **Concatenated :** Concatenating both the factors and then 1 attention layer attending the concatenated condition.

    Concatenated attention gives the best possible result followed by sequential (image first), sequential (ingredients first) and at last independent.

    Model is optimized to reduce negative-log likelihood and trained using adam optimizer.

For the multiple recipes from the same ingredients we are using Greedy search. It is a fairly obvious way to simply take the word that has the highest probability at each position and predict that. It is quick to compute and easy to understand, and often does produce the correct result.

We are also using beam search.

Beam Search makes two improvements over Greedy Search.

With Greedy Search, we took just the single best word at each position. In contrast, Beam Search expands this and takes the best 'N' words.

With Greedy Search, we considered each position in isolation. Once we had identified the best word for that position, we did not examine what came before it (ie. in the previous position), or after it. In contrast, Beam Search picks the 'N' best sequences so far and considers the probabilities of the combination of all of the preceding words along with the word in the current position.

For removing the ingredients we have changed the length of ingredients predicted for every image, earlier it was 20, we changed it to 7. So that the ingredients with less probability can be discarded and sequence of recipes will be generated accordingly.

Adding ingredients was a little tough. After reviewing the code, we found out that the model was also giving the probabilities of each ingredient present in ingredients vocabulary as output. Previously, It would just sort out the ingredients according to their probability distribution and as soon as it encountered the end of ingredients token (<eoi>), it would discard the other ingredients present after it and pad the sequence with <pad> token. We used the other ingredients after <eoi> to recommend it to the user. Sometimes the output was reasonable and sometimes it didn't make sense. But it would give an intuition on what else can be added.

Result And Performance

The accuracy of the model is as it is as we have not retrained the model. But there are significant differences in our output as we are now having 4 recipes for single output instead of 1 recipe. Along with every recipe suggestions are given that which ingredients can be added and which can be removed.

We have added one of the results predicted by our model.

/content/modules/multihead_attention.py:128: UserWarning: masked_fill_ received a mask with
  float('-inf'),
RECIPE 1

Title: Garlic shrimp scampi

Ingredients:
shrimp, pepper, butter, clove, oil, salt, pasta, parsley

Instructions:
-Heat olive oil in a large skillet over medium heat.
-Cook and stir garlic in hot oil until fragrant, about 1 minute.
-Stir shrimp into garlic; cook and stir until shrimp are pink and opaque, about 3 minutes.
-Season with salt and pepper.
-Stir butter into shrimp mixture until melted and sauce is heated through, about 2 minutes.
-Stir parsley into shrimp mixture; cook and stir until heated through, about 1 minute.

Following ingredients can be added:
cheese
wine
==========

RECIPE 2

Title: Pasta ai carciofi

Ingredients:
shrimp, pepper, butter, clove, oil, salt, pasta, parsley

Instructions:
-Saute garlic in oil, until softened but not browned.
-Add shrimp, saute for 2 min.
-Add pasta sauce, and parsley.
-Toss to coat the pasta.
-Season with salt, pepper, and red pepper flakes.
-Serve warm.

Following ingredients can be added:
cheese
wine
==========
RECIPE 3

Title: Easy shrimp scampi

Ingredients:
shrimp, pepper, butter, clove, oil, salt, pasta, parsley

Instructions:
-Heat oil in heavy large skillet over medium-high heat.
-Add garlic, pepper, and salt; stir-fry 30 seconds.
-Add shrimp.
-Stir-fry 5 minutes or until shrimp turn pink.
-Add pasta, parsley, butter, and pepper.
-Toss to coat.

Following ingredients can be added:
cheese
wine

RECIPE 3

Title: Easy shrimp scampi

Ingredients:
shrimp, pepper, butter, clove, oil, salt, pasta, parsley

Instructions:
-Heat oil in heavy large skillet over medium-high heat.
-Add garlic, pepper, and salt; stir-fry 30 seconds.
-Add shrimp.
-Stir-fry 5 minutes or until shrimp turn pink.
-Add pasta, parsley, butter, and pepper.
-Toss to coat.

Following ingredients can be added:
cheese
wine

RECIPE 4

Title: Garlic shrimp pasta with fresh garlic

Ingredients:
shrimp, pepper, butter, clove, oil, salt, pasta, parsley

Instructions:
-Heat a skillet over high heat.
-Add oil and shrimp; sprinkle with salt and pepper.
-Stir-fry for 2-3 minutes or until shrimp turn pink.
-Remove shrimp from pan.
-Cook linguine pasta according to package directions, omitting salt and fat.
-Drain pasta; return to pan.
-Add butter and garlic; toss.
-Stir in parsley; toss gently to coat.
-Serve immediately.

Following ingredients can be added:
cheese
wine

After removing some of the ingredients:

Title: Garlic shrimp pasta

Ingredients:
shrimp, pepper, butter, clove, oil, salt, pasta

Instructions:
-Cook pasta according to package directions.
-Drain and set aside.
-In a large skillet, heat oil over medium heat.
-Add garlic and cook for 1 minute.
-Add shrimp and cook for 2 minutes.
-Add butter and cook for 2 minutes.
-Add salt and pepper.
-Toss in pasta and serve.

Conclusion

We have successfully done our upgradation task, our model is giving 4 recipes as output along with a list of ingredients which can be added and can be removed.

References

https://towardsdatascience.com/foundations-of-nlp-explained-visually-beam-search-how-it-works-1586b9849a24

https://ai.facebook.com/blog/inverse-cooking/

http://pic2recipe.csail.mit.edu/

http://im2recipe.csail.mit.edu/dataset/login/