Flavor similarity with Deep Learning

gautam@ischool.berkeley.edu

Abstract

In this paper, we explore different ways to detect similarities (or differences) between two or more food dishes considering their flavor profiles. By conducting experiments using NLP and deep learning techniques on food recipe data, it appears that breaking food instructions down to their flavor compounds and learning non-linear features on recipe data gives a higher degree of accuracy in identifying the right combination of flavors for a food recipe.

1 Introduction

Detecting similarity in flavor in food items or recipes is a complex task. While this task is easily achieved by humans, it is difficult to simulate the experience of "tasting" food in a machine. However, while difficult, this can be a very important feature for food based websites and apps which like to offer similar tasting foods based on user's flavor profiles.
A typical representation of a food item can be in terms of the picture of the dish or a recipe with ingredients and cooking instructions. While ingredient similarity between two dishes does offer some indication of similarity, a minor difference in a cooking instruction can drastically alter the flavor.
The goal of this system is therefore to provide an indicative measure of how similar or dissimilar two dishes are in terms of their taste by merely relying on cooking instructions. We look at various ways of comparing these cooking instructions and suggest the best possible way to do so.

2 Datasets

2.1 Cooking Instructions

We downloaded recipes from allrecipes (https://www.allrecipes.com), epicurious (https://www.epicurious.com/), bbc.co.uk (https://www.bbc.com/food) and cookstr (https://www.cookstr.com/recipes).
Each set of recipes was downloaded as a json file. At the least, each json file had information about the ingredients and the cooking instructions along with a title of the food dish.
In all, a total of 400,000 (approx.) recipes were downloaded.

2.2 Flavor network

We downloaded an ingredient-compound map from an article on nature.com. (https://www.nature.com/articles/srep00196).
The download consisted of the following files:
ingr_info.tsv (id, ingredient-name, type) – 1529 ingredients,
comp_info.tsv (id, compound-name, CAS) – 1106 flavour compounds and
ingr_comp.tsv (ingredient_id, compound_id).

2.3 Pre-trained Embeddings

We tried using bert (https://github.com/google-research/bert) but we had to abort due to limited compute resources. We then used GLoVe embeddings from a pre-trained dataset at http://nlp.stanford.edu/projects/glove.
We also used gensim's word2vec algorithm for generating embeddings on our recipe dataset.

3 Background

The main inspiration was drawn from a paper titled “Flavor network and the principles of food pairing”.
(https://www.nature.com/articles/srep00196).
This paper introduces a complementary dataset (mentioned in section 2) which attempts to break food ingredient into flavor compounds. Each ingredient can made up of more than one flavor compound and therefore it is possible for multiple ingredients to share a flavor compound.
We also conducted an exhaustive research on various ways to create vector representations of sentences and took some inspiration from https://github.com/epfml/sent2vec/blob/master/paper-sent2vec.pdf.
Finally, coming from the basic idea that since each recipe is a paragraph of sentences and words, we attempted to use Google’s Universal Sentence Encoder as a baseline for our experiments.
(https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/46808.pdf)

4 Evaluation

We took the help of some domain experts to come out with 5 recipes in their original and modified forms. The modified forms were altered forms of recipes where

a. a minor/major change in an ingredient or preparation method would significantly alter the flavor of the food dish.
b. a minor/major change in an ingredient or preparation method would not alter the flavor of the food dish

The goal was for the model to score the difference in such a way that as the score got closer to 1, the flavor difference was less significant.
We don’t have a benchmark to test the models on other than compare the scores with that of experts. We also compared the approaches with each other to detect which was the best for this type of data. Also, since not every ingredient in a recipe was available within the flavor network data, we chose only those recipes where each ingredient was a part of the flavor network for our evaluation process.

5 Method

5.1 Data preparation and cleanup

Since the format of each json file was different, we had to write custom parsers for each json type. The parser created a tuple
(title, ingredients, instructions)
for each recipe from all downloaded data. Data was then sanitized using the **nltk** package by removing all punctuation marks and lemmatizing the result so that plurals were converted to singulars. We did not remove quantities and measures because we expected them to play a role in the flavor.
The food flavor network data was also cleaned using the techniques described above. It was then parsed and three data structures were built out of it.
Map<ingredient_id, ingredient_name>
Map<compound_id, compound_name>
Map<ingredient_id, compound_id>

5.2 Universal Sentence Encoder directly on recipe data

Our first experiment was directly using the universal encoder on recipe test data. The steps followed were:

- Set the module up using tensorflow hub
- Get fixed length embedding vector (length 512) for each sentence using https://tfhub.dev/google/universal-sentence-encoder-large/3
- Get the cosine similarity between original recipe and modified recipe for each example.

5.3 Universal Sentence Encoder on “compoundized” recipes

We then replace the original ingredient with their compoundized counterparts. We use data gathered by the flavor network so that a recipe which has the form:

"Take a bowl of ingredient-1. Add ingredient-2 to it. Boil for 10 mins. Serve hot."
is transposed into:
"Take a bowl of flavor-compond-1 flavor-compond-2 flavor-compound-3. Add flavor-compound-2 flavor-compound-5 flavor-compound-6 to it. Boil for 10 mins. Serve hot."
using the following mapping:

Ingredient	Constituent compound
Ingredient-1	Flavor Compound-1
	Flavor Compound-2
	Flavor Compound-3
Ingredient-2	Flavor Compound-2
	Flavor Compound-5
	Flavor Compound-6

Table 2 summarizes the results for each recipe.

5.4 Word2Vec on original recipe

We train word embeddings for the entire recipe corpus using gensim's word2vec algorithm. We created embeddings for multiple combinations of embedding dimension size, window size, minimum count and iterations.
We built a small "recipe2vec" algorithm which each element of the word vector over all the words in a recipe.
We then use a cosine similarity measure on our test set as above, to determine the similarity score between original and modified recipes.

5.5 Word2Vec on compoundized recipe

As a continuation to using Word2Vec, we next train word2vec on compoundized recipes. This allows us to create embeddings for each compound and use the same function we used in 5.4 against a compoundized recipe.

5.6 Training a neural network using data from flavor network

We formulate this as a multi-class classification problem and train the weights of a model with different types of networks.

The input (independent variables/features) is a word sequence representation of the IDs of each word in a recipe. An embedding matrix is then created for this sequence using Word2Vec trained on the entire corpus of 400K recipes. The embedding matrix is therefore a contextual representation of each word in the corpus.
The size of the input for each sequence is therefore (max-length * dim-size)
The maximum input length of the sequence is limited to 100 words and where recipes are < 100 words, the sequence is padded with 0s.
The output variable (dependent variable) is a vector of 1s and 0s indicating the presence or absence of that flavor compound in that recipe with size = number of distinct compounds in flavor network.
i.e. if the output vector is [1, 1, 1, 0, 0, 1], the flavor compounds with index 0, 1, 2 and 5 are present in the recipe.
The neural networks therefore have a general structure shown in Figure 1.

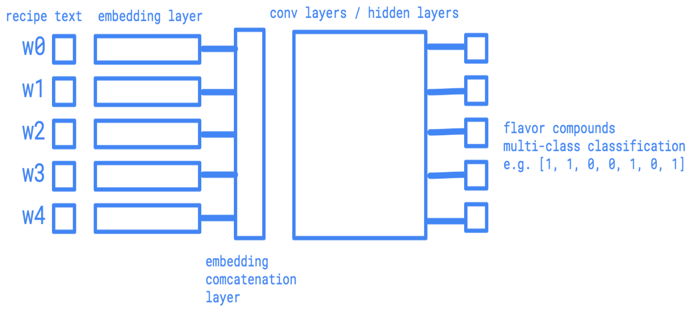


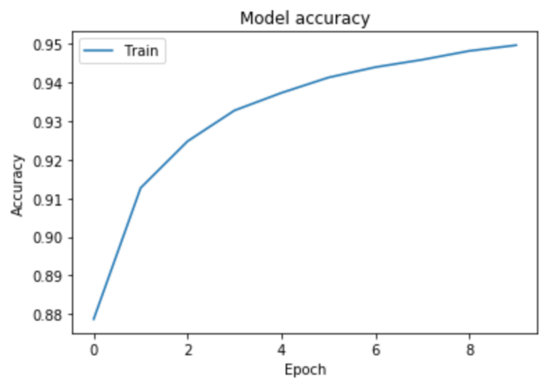
Figure 1: General structure of neural networks

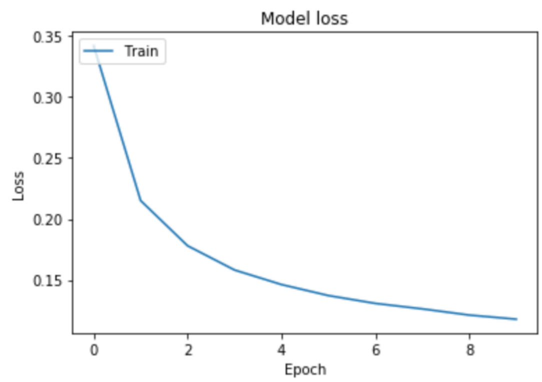
The final similarity score is arrived at by measuring the distance between the output vectors.
We try this approach on different network architectures as described below. This gives us an idea on how each of these architectures behave on the same dataset.
In each of the following experiments, we use a normalized Euclidean distance as a measure of how dissimilar two dishes are in their flavor. We also pick the top 5 most similar tasting foods and top 5 most dissimilar tasting food dishes.

To further ascertain the validity of our results, we test our programs on a 5 recipe dataset which has been modified by domain experts so that minor modifications alter the taste of the dishes.

5.7 Fully connected neural network

The basic network is a fully connected network with 4 hidden layers.
This network was trained on 3000 recipes with 10 epochs and an overall accuracy of 96% was achieved on a test set of 2000 recipes.
The model accuracy and loss progressions on 10 epochs is shown below:



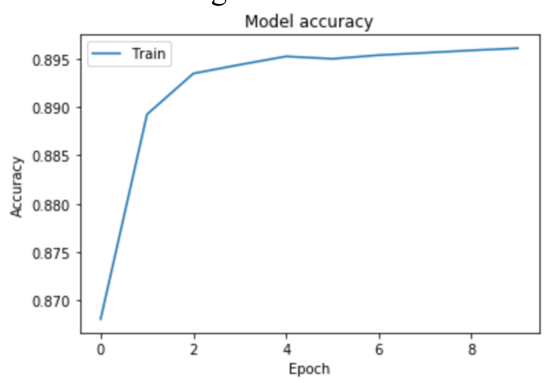


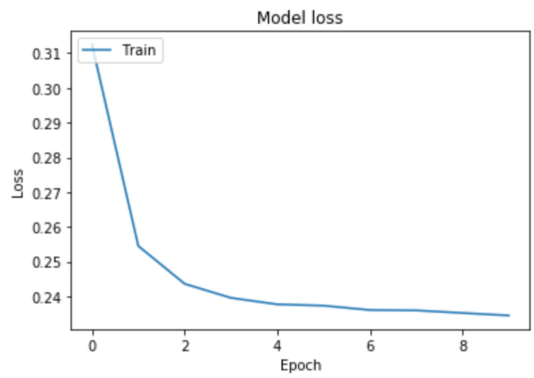
Similarly, we expanded the network to have more hidden layers but we did not get much performance improvement or accuracy numbers from this expansion.

5.8 Convolutional Neural Network

A slightly more involved neural network on which we trained and tested this data was a simple 1d convolutional network. The model accuracy was about 89.5% on 3000 recipes with 10 epochs.

We tried using both global average pooling and max pooling – though max pooling seemed more relevant to our case as we were trying to detect specific sequences in our recipes. We also used a dropout rate of 50% to prevent overfitting of our data.





The model plot is shown in Figure 2:

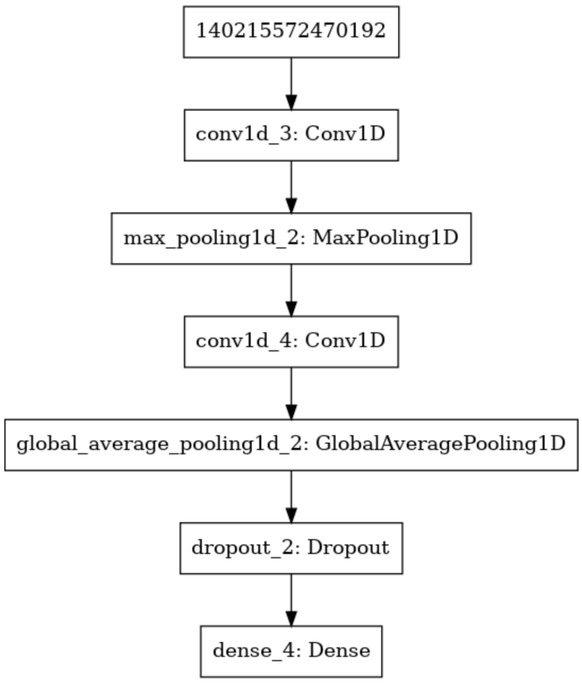
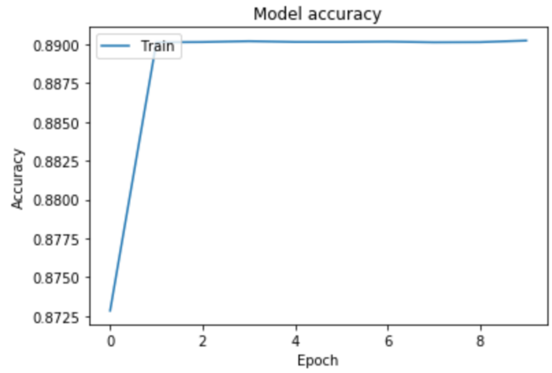
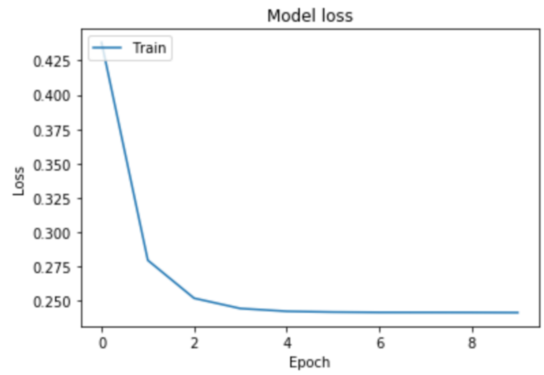


Figure 2: 1d CNN Model Plot

5.9 LSTM

In this model, we stack 3 LSTM layers on top of each other, making the model capable of learning higher-level temporal representations. The first two LSTMs return their full output sequences, but the last one only returns the last step in its output sequence, thus dropping the temporal dimension (i.e. converting the input sequence into a single vector).
Similar to the above runs, we train this model with 10 epochs and get a steady accuracy of about 90%.





The LSTM model plot is shown in figure 3.

6 Results & Discussion

The results of the top 5 most similar tasting pairs of dishes and the top 5 least similar tasting pairs of dishes as predicted by each of the above models is as shown in table 1.
In addition, we also test our models on our gold set of 5 recipes and compare the scores with top/bottom 5 obtained above.

Fully Connected Network

Top 5 Dissimilar pairs from bbc.co.uk recipes

1	Nonfat Sour Cream Cake Bread	Oatmeal Chocolate Chip Muffins
2	Yummy Lemon Bread	Nonfat Sour Cream Cake Bread
3	Cardamom Bread	Oatmeal Chocolate Chip Muffins
4	Candied Fruit Bread	Nonfat Sour Cream Cake Bread
5	Candied Fruit Bread	Cardamom Bread

Top 5 Similar pairs from bbc.co.uk recipes

1	Sun Dried Tomato and Asiago Cheese Bread	Pumpernickel Bread
2	Pumpernickel Bread	Cottage Dill Bread
3	Sun Dried Tomato and Asiago Cheese Bread	Cottage Dill Bread
4	Mom's Hazelnut Special	Cottage Dill Bread
5	Honey White Bread	Cottage Dill Bread

We got similar results for bbc.co.uk recipes from our convolutional network as well as our LSTM with some minor differences.
We then ran predictions on allrecipes.com recipes using the Convolutional Neural Network and got the following results

Convolutional Neural Network

If we combine allrecipes.com data with bbc.co.uk during prediction, the following are the top 5 dissimilar pairs:

1	Sunday Dinner Rolls	Poppy Seed Bread
2	Sourdough Bread	Darbey Bread
3	Cinnamon Cranraisin Bread	Challah
4	Pumpkin Bread	Challah
5	Yummy Lemon Bread	Chocolate bread

The following are the results on the 5-recipe gold set using a LSTM. We expect differences

in tastes of these recipes (a score of 1.0 indicates most similar) even though the number of ingredients changed are not too many.
Each recipe has been paired with a modified version of itself where only a few flavor influencing ingredients have been changed.

Recipe pair	Similarity score (1.0 = most similar)
Recipe 1	0.712
Recipe 2	0.534
Recipe 3	0.708
Recipe 4	0.845
Recipe 5	0.651

All the above results look very promising. It appears that data from the flavor network can be used in various ways to detect changes in flavor if some of the main flavor influencing ingredients of a recipe are altered.
As indicated above, these subtle changes in flavor are not caught by applying a similarity measure on the entire pair of recipes. We have seen that similarity scores are often very high even if some of the core ingredients of a recipe are changed.
Our approach of combining recipe text with data from flavor network using Deep Networks like CNN and LSTM have produced better results.
Note that our models were trained on a 24 vCPU instance and so it would take a considerable amount of time training all 400,000 recipes. We therefore, narrowed our training set to only a few 1000 recipes. It is extremely possible that extending the training set to include all recipes will give much better results.
One might argue that this approach is mainly rule based but we need to consider the point that the model is learning weights by using embeddings trained using the entire recipe data.
A few possible improvements to our approach are:

1. Train word embeddings on bigrams and trigrams
2. Consider the importance of verbs (boil, bake, fry etc.) in the classification.
3. Improve flavor network data by adding flavors of all ingredients and combination of verb-ingredient pairs in the recipe.

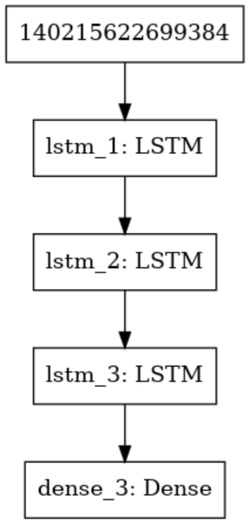


Figure 3: LSTM model plot

7 References

1. Yong-Yeol Ahn, Sebastian E. Ahnert, James P. Bagrow, Albert-László Barabási - *Flavor network and the principles of food pairing.*
2. Daniel Cer, Yinfei Yanga , Sheng-yi Konga et al. - *Universal Sentence Encoder*
3. Tomas Mikolov et al. - *Distributed Representations of Words and Phrases and their Compositionality*