Project Description:

This project tries to cover the knowledge of Python, spaCy library and text clustering. We want to do a flavor recommendation, so we need to extract the information we want from the data we have.

We have three files: srep00196-s2.csv, srep00196-s3.csv and yummly.json, srep00196-s2.csv, srep00196-s3.csv is to construct flavor network which from the paper: <http://www.nature.com/articles/srep00196>, we can use it to construct flavor network as our baseline.

The main steps we do will be config parsing, graph constructing, data cleaning, model training and flavor recommending.

Config parsing. We will parse the command arguments to get the data we are concerning. The data we concern is –-N and –-ingredient. We will use a function named parse\_config return a dict to store the parse result to use it later.

Graph constructing. We use srep00196-s2.csv and srep00196-s3.csv to build flavor network. For ingredients, we will use ingredients from srep00196-s3.csv, that is to say only ingredient in srep00196-s3.csv will be reserved and we use all\_ingredients to store it, the length of all\_ingredients is 381, then we use srep00196-s2.csv to construct flavor graph and the node is ingredient in all\_ingredients, to reproduce the article method, we also calculate prevalence for flavor network.

Data cleaning. In this step, we want to clean data from yummly.json, in this file we have dishes with different ingredients, we will clean the ingredients so that all ingredients are in all\_ingredients from previous step, and we will replace oil with olive\_oil in ingredients as oil is not in all\_ ingredients. We first do lemmatization for ingredient, like eggs will be replaced with egg to extract the main information of ingredients, and then we will ignore the character like ‘-’ in ingredients. After data clean, we will do a simple embedding for ingredients, the step is for all\_ingredients, we use True means ingredients for one dish contain such ingredient, and False if the ingredient in all\_ingredients is not in ingredients for dish. By the way, as the process is slow, so we only process part of the data, we use a constant number NUM to indicate how many rows we will process.

Model training. We use SVM as the model we do the train. We use the embedding we construct as the train data and the cusine data from yummly.json as the result value.

Flavor recommending. We first construct the embedding for input ingredients and use it as the input of the model we trained from previous step, then we print the result we predict.

How to run

pipenv run python project2.py --N 5 --ingredient paprika --ingredient banana --ingredient "rice krispies"

Data description

srep00196-s2.csv, srep00196-s3.csv and yummly.json, srep00196-s2.csv, srep00196-s3.csv is to construct flavor network which from the paper: <http://www.nature.com/articles/srep00196>. yummly.json is the dishes data.

Functions

project2.py \

parse\_config()- this function parse the command line to get config we need to deal with. It will return a config dict.

build\_graph()- this function is to build a flavor network from two data files: srep00196-s2.csv, srep00196-s3.csv. We use ingredients from srep00196-s3.csv to get all ingredients we concern and use srep00196-s2.csv to build flavor network, we use the method from the article: Flavor Network and the Principles of Food Pairing to build the graph. The return data is G for flavor graph built by networkx, graph\_dict for flavor graph represented by dict object, prevalence for flavor graph and all\_ingredients from srep00196-s3.csv.

replace\_flavour\_ingredient()-this function will replace oil to olive\_oil and return the result. The input is a string, if oil is in input it will return olive\_oil else return the input.

clean\_ingredients()- this function is to do lemmatization for ingredients. Input is ingredients to deal with, nlp from spacy and all\_ingredients for the ingredients we are concerned on. we do the lemmatization for ingredients and reserve the ingredient we are concerned on, then we return the result we process.

get\_pd\_ingredients()- this function is to do embedding for ingredients. Input is ingredients to deal with and all\_ingredients for the ingredients we are concerned on. We use True means ingredients contain such ingredient, and False if the ingredient in all\_ingredients is not in ingredients then we return the embedding result in DataFrame format.

train()- this function is to do model training. Input is train data and the model we trained.

get\_yummly\_data()- this function is to do the data cleaning and model training we describe above. The input is all\_ingredients for the ingredients we are concerned on. We do the data cleaning for yummly.json and return the result as yummly\_data in DataFrame format, and train the a SVM model for the yummly data return as model, and we will return a list format for all\_ingredients(all\_ingredients is in set format), and return the train\_data for model training.

calc\_cosine\_similarity()-this function is to calculate the cosine similarity for dish ingredients and ingredients for user input. The input is dish id, dishes and the ingredients for user input, and a list for (similarity, id) for dishes will return.

Bugs and Assumptions

Bugs:

We can’t process so many data in yummly.json as it will take long time.

Assumptions:

As for model training, we can use different model like random forest or k-means to train. For data cleaning, the method we use is simple, we can use a more effective to do such process.

Output format

The result is a dict format for the dishes we recommend. Cuisine is the country from the recommendation, id is the dish id and score is the match score for cuisine, closest is the top N dishes we recommend.