CGAS_Assignment1_MT20075

September 10, 2021

1 Answer to Question 1

1.1 (a)

1.2 ### RECIPE-1

INGREDIENTS SECTION
1 cup bajra (black millet) flour
1/2 cup grated carrot
1/2 cup finely chopped spinach (palak)
1 tsp green chilli paste
1 tsp garlic (lehsun) paste
salt to taste
bajra (black millet) flour for rolling
oil for cooking

**COOKING INSTRUCTIONS
1. Combine all the ingredients in a deep bowl and knead into semi-soft diugh using warm water.
2. Divide the dough into 8 equal portions and roll out ech portion into a circle of 100 mm (4") diameter, using a little bajra flour for rolling.
3. Heat a non-stick tava (griddle) and cook each paratha using a little oil, till it turns golden brown in colour from both sides.
4. Serve immediately.
**

**

--- ### RECIPE-2
--- INGREDIENTS SECTION**
3/4 cup urad (whole black lentil)
1 tbsp chana dal (split bengal gram)
1 tbsp rajma (kidney beans)
1 onion
25 mm (1") piece of ginger (adrak)
4 green chillies
1 tomato
1/3 cup fresh cream
1/3 cup curds (dahi)

1/2 tsp chilli powder
1/2 tsp turmeric powder (haldi)
1 tsp cumin seeds (jeera)
2 tbsp ghee
salt to taste

**COOKING INSTRUCTIONS
1. Soak the gram dal and rajma for 5 hours.
2. Chop the green chillies, ginger, onion and tomato.
3. Put the urad dal, gram dal and rajma in a vessel, add 2 teacups of water and cook in a pressure cooker.
4. Mash very well with a big spoon and boil again for 20 minutes.
5. Add the curds and ream to the cooked dal.
6. Heat the ghee in a vessel and fry the onion, chillies, ginger and cumin seeds for 2 minutes.
7. Add the tomato, turmeric powder and chilli powder. Fry again for 2 minutes.
8. Add the dal and salt and cook for a few minutes.
9. Serve hot with parathas.
**

**

--- ### RECIPE-3
--- INGREDIENTS SECTION**
1 1/2 cups paneer (cottage cheese) cubes
2 tbsp besan (bengal gram flour)
1/4 cup thick curds (dahi)
1/2 tsp chaat masala
1/2 tsp garam masala
1/4 tsp ginger (adrak) paste
1/2 tsp garlic (lehsun) paste
1 tsp chilli powder
1 tbsp oil
salt to taste
1/2 cup capsicum cubes
1/4 cup onion cubes
oil for greasing and cooking
1 tbsp cornflour
3 tbsp oil
1 small stick cinnamon (dalchini)
2 cloves (laung / lavang)
1 tsp garlic (lehsun) paste
1/2 tsp ginger (adrak) paste
1/2 cup onion paste
salt to taste
1 cup fresh tomato pulp
1 1/2 tsp chilli powder
1/4 tsp turmeric powder (haldi)
1/2 tsp coriander-cumin seeds (dhania-jeera) powder
1 tbsp fresh cream
2 tbsp finely chopped coriander (dhania)

**COOKING INSTRUCTIONS
1. Combine the besan, curds, chaat masala, garam masala, ginger paste, garlic paste, chilli powder, oil and salt in a deep bowl and mix well using a whisk.
2. Add the paneer, capsicum and onions and toss gently till they are evenly coated from all the sides.
3. Thread a piece of paneer cube followed by 2 capsicum and 2 onion cubes alternatively. Again thread another piece of paneer cube followed by 2 capsicum and 2 onion cubes alternatively and finally thread a paneer cube on a skewer or a long tooth pick.
4. Repeat with the remaining ingredients to make 3 more tikkas.
5. Grease a non-stick tava (griddle) with oil, place the prepared tikkas over it and cook them on a medium flame, using oil till they turn golden brown in colour from all the sides. Keep aside.
6. Separately, combine the cornflour and 1 cup of water in a deep bowl, mix well and keep aside.
7. Heat the oil in a deep non-stick kadhai, add the cinnamon, cloves, garlic paste and ginger paste and sauté on a medium flame for a few seconds.
8. Add the onion paste and salt and sauté on a medium flame for 2 to 3 minutes.
9. Add the fresh tomato pulp, mix well and cook on a medium flame for 2 to 3 minutes, while stirring occasionally.
10. Add the chilli powder, turmeric powder, coriander-cumin seeds powder, fresh cream, coriander and cornflour-water mixture, mix well and cook on a medium flame for 2 to 3 minutes, while stirring occasionally.
11. Just before serving, re-heat the gravy and slide the paneer tikkas from the skewers into the gravy. Mix gently and cook on a medium flame for 2 minutes, while stirring occasionally.
12. Serve the paneer tikka masala immediately.
**

**

--- **###** RECIPE-4
--- INGREDIENTS SECTION**
3 cups cauliflower florets
1/2 cup plain flour (maida)
1/4 cup cornflour
1 1/2 tsp chilli powder
1/2 tsp ginger (adrak) paste
1/2 tsp garlic (lehsun) paste
1 tbsp finely chopped coriander (dhania)
1/2 tsp garam masala
1/4 tsp turmeric powder (haldi)
1/2 tsp coriander (dhania) powder
1 tsp lemon juice
salt to taste
oil for deep-frying
1 tbsp oil
1 tsp finely chopped garlic (lehsun)
1 tsp finely chopped ginger (adrak)
2 slit green chillies
6 curry leaves (kadi patta)

1/2 cup sliced onions

**COOKING INSTRUCTIONS
1. Combine the plain flour, cornflour, chilli powder, ginger paste, garlic paste, coriander, garam masala, turmeric powder, coriander powder, lemon juice and salt in a deep bowl and mix well.
2. Add approx. 3/4 cup of water and mix well.
3. Add the cauliflower florets and mix gently.
4. Heat the oil for deep-frying in a deep non-stick pan and deep-fry a few cauliflower florets at a time till they turn golden brown in colour from all the sides. Drain on an absorbent paper.
5. Heat the oil in a broad non-stick pan, add the garlic, ginger, green chillies and curry leaves and sauté on a medium flame for 1 minute.
6. Add the onions and little salt and sauté on a medium flame for 2 minutes.
7. Add the deep-fried cauliflower and sauté on a medium flame for 2 minutes.
8. Serve immediately.
**

**

--- **###** RECIPE-5
--- INGREDIENTS SECTION**
2 cups crumbled gulab jamun mava (hariyali khoya)
1/4 cup plain flour (maida)
3 tbsp milk powder
3 tbsp arrowroot (paniphal) flour
3 cups sugar
a few saffron (kesar) strands
ghee for deep-frying

COOKING INSTRUCTIONS
1. Combine the sugar and 1 ¡ cups of water in a deep pan and cook on a medium flame for 7 to 8 minutes, while stirring occasionally.
2. Simmer on a medium flame for 4 to 5 minutes or till the syrup is of one string consistency.
3. Remove any impurities which float on top of the syrup using a slotted spoon.
4. Add the saffron and keep the syrup warm.
5. Separately, combine all the ingredients in a deep bowl and knead very well into a very smooth dough.
6. Divide this mixture into 30 equal portions and roll each into round balls, there should have no cracks on the surface as otherwise the gulab jamuns will crack while deep-frying.
7. Heat the ghee in a non-stick kadhai and deep-fry a few jamuns on a slow flame till they turn golden brown in colour from all the sides.
8. Drain and immerse in the warm sugar syrup. Soak for 1 hour.
9. Serve immediately or serve warm.

1.3 (b)

Storing the recipes in the form of a (Recipe ID)—(Ingredient Name) form in an Excel (.xlsx) file format and loading the Excel Data to show the final output after storing as shown below:

[1]:
```
#Loading Google Drive to access the file
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

[2]:
```
import numpy as np
import pandas as pd
```

[3]:
```
file_id = '1YQgHm7Q7PTYhz2DAoD84wrzqj9DSln18'
downloaded = drive.CreateFile({'id': file_id})
downloaded.GetContentFile('Recipe_Ingredients.xlsx')
df = pd.read_excel('Recipe_Ingredients.xlsx')
df
```

[3]:

	Recipe ID	Ingredient ID
0	Recipe1	Bajra (black millet) flour
1	Recipe1	Carrot
2	Recipe1	Spinach (palak)
3	Recipe1	Green chilli paste
4	Recipe1	Garlic (lehsun) paste
..	...	...
60	Recipe5	Milk powder
61	Recipe5	Arrowroot (paniphal) flour
62	Recipe5	Sugar
63	Recipe5	Saffron (kesar) strands
64	Recipe5	Ghee

[65 rows x 2 columns]

1.4 (c)

1. The process of coarse-graining leads to loss in quantity or the amount of ingredients needed to add into the recipe to make the dish.

2. Furthermore, information regarding the method or procedure to make the dish is also missing which makes it difficult to understand the sequence in which the ingredients are supposed to be put into the recipe to make the dish.
   For instance, let us look at the recipe for "Gulab Jamun" (Recipe-5 above), if we were to ignore the amount of different ingredients needed and also the methodology (cooking instructions), we are just left with the ingredient names along with the state of those

ingredients like finely chopped, paste etc. In this recipe, there are a lot of ingredients like flour, ghee, sugar, saffron etc. Thus, if we do not know the sequence of the ingredients to be put in the recipe, we may end up with a completely different end product with a completely different taste, texture and look rather than Gulab Jamun.

3. Other information regarding the types of utensils needed (bowl, tray etc.) is also missing which is also essential to know the nature of the dish being cooked.

4. Information like the temperature in which certain ingredients are required while cooking the main dish is also missing. This greately changes the dish end-product taste and texture being prepared.

5. Time needed for preparing different sub-ingredients to be mixed to prepare the dish and the entire cooking time of the dish is missing as a result of course-graining the data.

1.5 (d)

This problem can be mitigated by: 1. Storing the recipes in nested data-structures such as list of dictionaries or dictionaries of dictionaries, etc.
Below is an example of this proposed idea:
Using both dictionary of dictionaries and dictionary of list wise values, let us see the example format as follows:
'recipie-01' : 'ingredient' : ['ingredient1','ingredient2',.....], 'method' : ['instruction1','instruction2',.....] ,
 'recipie-02' : 'ingredient' : ['ingredient1','ingredient2',.....], 'method' : ['instruction1','instruction2',.....] ,.....
Here, in the above example, the 1st dictionary contains keys as the "recipe-ID" and its value contains another dictionary containing keys as "ingredient" and "method" with their values as list of ingredients and list of cooking instructions stepwise respectively.
2. Another solution to mitigate the above problem is to use NLP based algorithms, specifically techniques like Name Entity Recognition (NER) and POS tagging which can help classify the different detectable elements in recipes such as ingredients like sugar represents Noun (NN) and words like caramelizing represents a verb (VBD) for the action to caramelize the sugar. This helps segregate the elements into the corresponding ingredients and methodology categories. Further these ingredients can be segregated into ingredient name, quantity, units, ingredient form and comments upon the ingredient status in similar form as shown in the above example.
3. Many factors detectable from the recipe data like quantity, quality, temperature, flavour of the ingredients should be taken into account while storing the data in the above format by extracting them using NLP.
4. Nutritional Information of the different ingredients used and their resultant dishes by understanding the category of dishes the recipe falls under (such as healthy recipes etc.)
5. Keeping track of the different versions of the same dish available with the help of uniquely generated IDs linked to the same dish ID would help one understand the different ways in which one dish can be prepared.

---

2 Answer to Question 2

For this question we prepare the functions to sequentially generate answers for each sub-question as follows:

[ ]:
```
import requests
import re
import json
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import math
import operator
from collections import Counter
```

[ ]:
```
#Loading json file in colab
def getData():
    with open ('/content/drive/MyDrive/CGAS/archive/train.json',␣
 ↪encoding="utf8") as f:
        recipedata = json.load(f)
        length = len(recipedata)
        print("Number of Recipes : ", length)
        data=[] #list for storing the file data
        for i in range(len(recipedata)):
          #this stores the recipeId, cuisine, ingredientList and recipeSize for␣
 ↪each recipe which will be all utilised later
          data.append({'id':recipedata[i]['id'], 'cuisine':
 ↪recipedata[i]['cuisine'], 'ingredients':
 ↪recipedata[i]['ingredients'],'recipe_size':
 ↪len(recipedata[i]['ingredients'])})
    return data
```

[ ]:
```
#For number unique cuisine from the all recipes in the loaded data
def get_unique_cuisine(data):
  uniqueCuisines = set()
  for j in range(len(data)):
    uniqueCuisines.add(data[j]["cuisine"])
  print("Number of Cuisines : ",len(uniqueCuisines))
  return uniqueCuisines
```

[ ]:
```
#For number of unique ingredients from the all recipes in data
def get_unique_ingredients(data):
  uniqueIngredients = set()
  for i in range(len(data)):
    for j in data[i]["ingredients"]:
      uniqueIngredients.add(j)
```

```
  print("Number of Unique Ingredients : ", len(uniqueIngredients))
  return uniqueIngredients
```

[ ]:
```
#For number of recipes in each cuisine
def get_numberOfRecipesCuisine(data):
  cuisine_recipeNum={} #dictionary storing cuisine and total number of recipes␣
 ↪of each cuisine
  for i in range(len(data)):
    if data[i]['cuisine'] not in cuisine_recipeNum:
      cuisine_recipeNum[data[i]['cuisine']]=1
    else:
      cuisine_recipeNum[data[i]['cuisine']]+=1
  print("Number of Recipes per Cuisine : ",cuisine_recipeNum)
  return cuisine_recipeNum
```

[ ]:
```
#For plotting the statistics (bar plot) of number of recipes for each cuisine.
def generate_Plot1(recipe_cuisine):
  fig = plt.figure(figsize=(18,10))
  ax = fig.add_axes([0,0,1,1])
  plt.title('Number of recipes for each cuisine')
  ax.set_ylabel('Recipes for each Cuisine')
  ax.set_xlabel('Cuisines')
 ␣
 ↪colors=['orange','cyan','lightcoral','yellow','maroon','blue','green','lavender','violet','
  cuisines=[]
  #using recipe_cuisine dictionary
  for recipe in recipe_cuisine.keys():
    cuisines.append(recipe)
  recipeNum = []
  for val in recipe_cuisine.values():
    recipeNum.append(val)
  ax.bar(cuisines,recipeNum,color=colors)
  plt.xticks(rotation=90)
  plt.tick_params(labelsize=12)
  plt.legend()
  plt.show()
```

[ ]:
```
#For total number of recipes with same sizes
def get_recipeDistCuisine(data):
  recipe_size = {} #storing the size of each recipe and total number of recipes␣
 ↪of that same size
  for ing in data['recipe_size']:
    if ing not in recipe_size:
      recipe_size[ing]=1
    else:
      recipe_size[ing]+=1
  print("\nRecipe Size Distribution : \n", recipe_size)
  for i in recipe_size:
```

```
    #normalizing recipe sizes with total number of recipes and computing␣
 ↪percentage
    recipe_size[i] = (recipe_size[i]/data.shape[0])*100
  print("\nRecipe Size Distribution in Percentage (%) : \n", recipe_size)
  return recipe_size
```

[ ]:
```
def generate_Plot2(recipe_size):
  fig = plt.figure(figsize=(18,10))
  recipe_sort = dict(sorted(recipe_size.items(), key=lambda x: x[0])) #sorting␣
 ↪recipe size wise
  X=[] #storing recipe_sizes in sorted order
  Y=[] #number of recipes of that same size
  for i in recipe_sort:
    X.append(i)
    Y.append(recipe_sort[i])
  plt.scatter(X, Y,c='maroon')
  plt.plot(X,Y,c='maroon')
  plt.title('Recipe-Size Distribution')
  plt.ylabel('Percentage (%)')
  plt.xlabel('Recipe-size')
  plt.grid()
```

[ ]:
```
#For recipe size distribution per cuisine (for each cuisine having same size␣
 ↪instead of recipes)
def get_cuisineDist(recipe_cuisine,data):
  cuisine_dist={}
  for cuisine in recipe_cuisine:
    cuisine_size={}
    cuis = data['recipe_size'].loc[data['cuisine']==cuisine] #storing each␣
 ↪entry of cuisine with corresponding recipe-sizes

    for c in cuis:
      if c not in cuisine_size:
        cuisine_size[c]=1
      else:
        cuisine_size[c]+=1

    for cus in cuisine_size:
      #normalizing as in above function
      cuisine_size[cus] = (cuisine_size[cus]/recipe_cuisine[cuisine])*100
    sort_cuisine = dict(sorted(cuisine_size.items(), key= lambda x:x[0]))␣
 ↪#sorting as in above function for recipes
    cuisine_dist[cuisine] = sort_cuisine #dictionary of dictionary␣
 ↪data-structure for storing cuisine then size and recipes of that same size
  return sort_cuisine,cuisine_dist
```

[ ]:
```
def generate_Plot3(cuisine_dist):
  fig = plt.figure(figsize=(18,10))
```

```
  for cuisine in cuisine_dist:
    X=[]
    Y=[]
    for id in cuisine_dist[cuisine]:
      X.append(id)
      Y.append(cuisine_dist[cuisine][id])
    plt.scatter(X,Y)
    plt.plot(X,Y,label=cuisine)
  plt.title('Recipe Size Distribution for each cuisine')
  plt.ylabel('Percentages of Each Cuisine')
  plt.xlabel('Number of Recipies in Each Cuisine')
  plt.legend()
  plt.grid()
```

[ ]:
```
#For cumulative distribution of recipe size
def get_RecipeSizeDist_Plot(recipe_size_dist):
  fig = plt.figure(figsize=(18,10))
  #sorting recipes_size_distribution in descending order.
  recipe_sorted = dict(sorted(recipe_size_dist.items(), key=lambda x:␣
 ↪x[0],reverse = True))
  cumulative={}
  count = 0
  for i in recipe_sorted:
    count+=recipe_sorted[i] #computing cumulative value(N<=9; less than equal␣
 ↪to 9), that is 9+8+7... so on
    cumulative[i] = count
  print("Cumulative Scores of recipe-sizes: ",cumulative)
  plt.title('Cumulative-Distribution of recipe size')
  plt.xlabel('Rank(based on popularity)')
  plt.ylabel('Utilisation-frequency')
  plt.grid()
  plt.plot(list(cumulative.keys()),list(cumulative.values()),'-o')
```

2.1 (a)

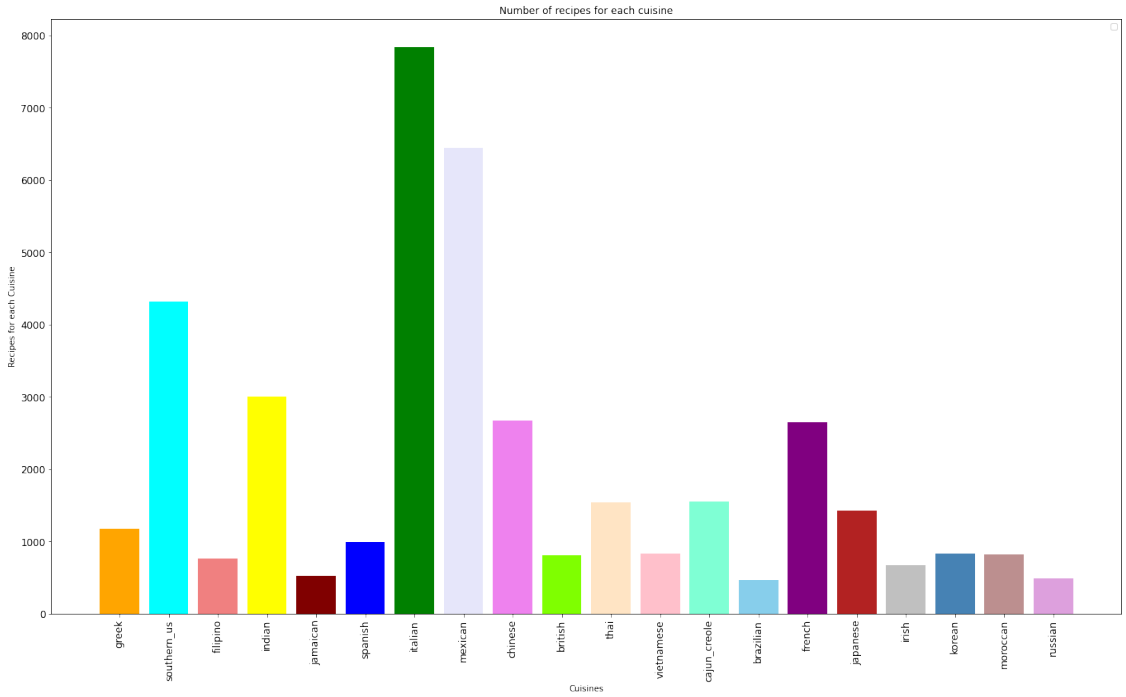
[ ]:
```
# Q2(a)
fin_data = getData()
df = pd.DataFrame(fin_data)
uniqueCuisine = get_unique_cuisine(fin_data)
uniqueIngredients = get_unique_ingredients(fin_data)
```

```
Number of Recipes :  39774
Number of Cuisines :  20
Number of Unique Ingredients :  6714
```

2.2 (b)

[ ]:
```
# Q2(b)
recipe_cuisine = get_numberOfRecipesCuisine(fin_data)
generate_Plot1(recipe_cuisine)
```

```
No handles with labels found to put in legend.

Number of Recipes per Cuisine :  {'greek': 1175, 'southern_us': 4320,
'filipino': 755, 'indian': 3003, 'jamaican': 526, 'spanish': 989, 'italian':
7838, 'mexican': 6438, 'chinese': 2673, 'british': 804, 'thai': 1539,
'vietnamese': 825, 'cajun_creole': 1546, 'brazilian': 467, 'french': 2646,
'japanese': 1423, 'irish': 667, 'korean': 830, 'moroccan': 821, 'russian': 489}
```



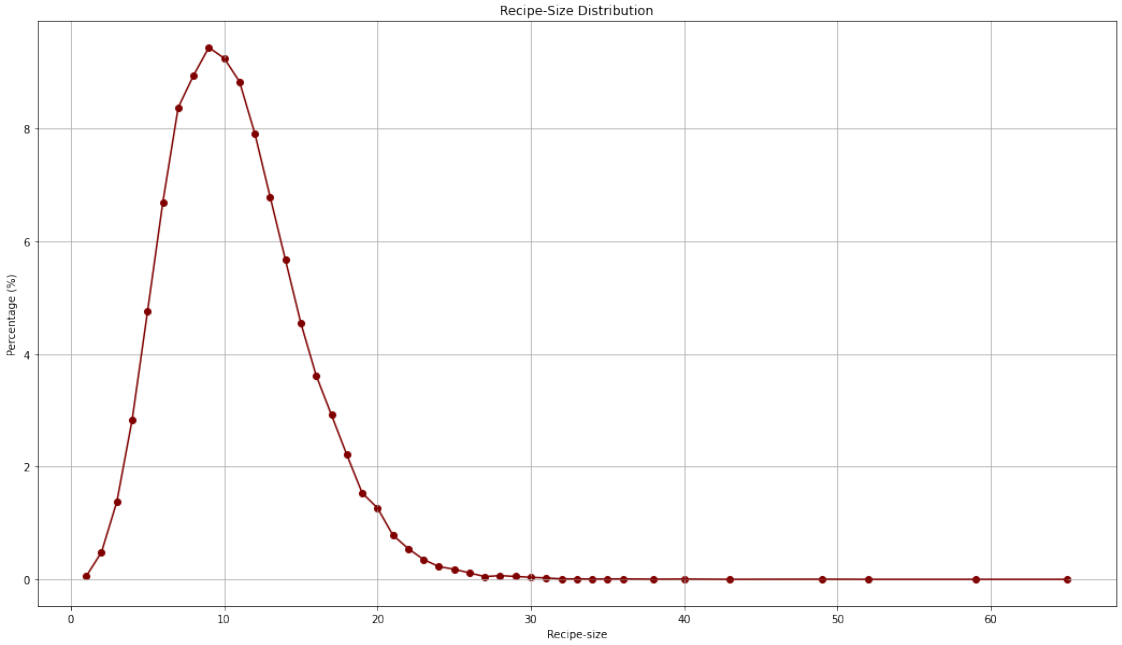
2.3 (c)

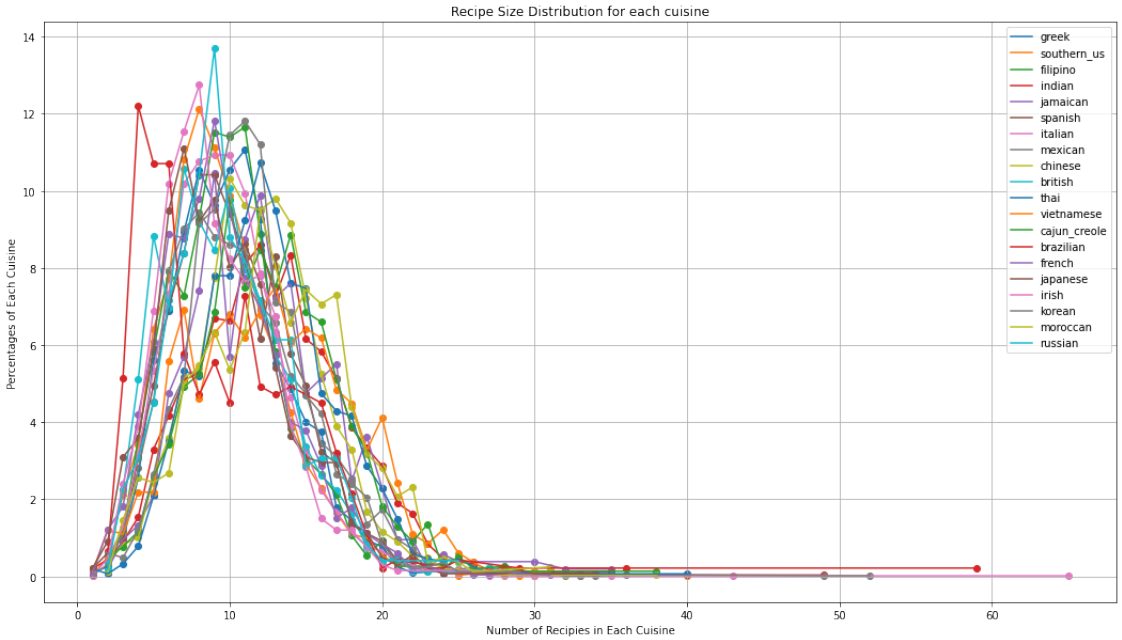
Recipe size distribution for each cuisine as well as for all the recipes:

[ ]:
```
recipe_size_dist = get_recipeDistCuisine(df)
generate_Plot2(recipe_size_dist)
sort_cuisine,cuisine_dist= get_cuisineDist(recipe_cuisine,df)
generate_Plot3(cuisine_dist)
```

Recipe Size Distribution :
{9: 3753, 11: 3512, 12: 3146, 4: 1128, 20: 504, 13: 2698, 10: 3677, 6: 2662, 15: 1809, 17: 1160, 14: 2253, 5: 1891, 16: 1439, 8: 3556, 7: 3329, 22: 218, 3: 549, 2: 193, 18: 879, 24: 91, 38: 2, 21: 313, 40: 3, 23: 141, 19: 610, 25: 72, 26: 46, 27: 20, 1: 22, 34: 3, 28: 27, 30: 15, 29: 21, 32: 4, 36: 4, 49: 2, 31: 11, 35: 3, 65: 1, 33: 4, 52: 1, 59: 1, 43: 1}

Recipe Size Distribution in Percentage (%) :
{9: 9.435812339719414, 11: 8.829888872127519, 12: 7.909689747070951, 4: 2.836023532961231, 20: 1.2671594508975714, 13: 6.783325790717553, 10: 9.244732739980892, 6: 6.692814401367728, 15: 4.548197314828783, 17: 2.916478101272188, 14: 5.664504450143309, 5: 4.75436214612561, 16: 3.6179413687333435, 8: 8.940513903555086, 7: 8.369789309599236, 22: 0.5480967466183939, 3: 1.3802986875848544, 2: 0.4852416151254589, 18: 2.2099864232915976, 24: 0.2287926786342837, 38: 0.0050284105194348064, 21: 0.7869462462915472, 40: 0.00754261577915221, 23: 0.3545029416201539, 19: 1.533665208427616, 25: 0.18102277869965305, 26: 0.11565344194700054, 27: 0.05028410519434806, 1: 0.05531251571378287, 34: 0.00754261577915221, 28: 0.06788354201236989, 30: 0.03771307889576105, 29: 0.05279831045406547, 32: 0.010056821038869613, 36: 0.010056821038869613, 49: 0.0050284105194348064, 31: 0.027656257856891436, 35: 0.00754261577915221, 65: 0.0025142052597174032, 33: 0.010056821038869613, 52: 0.0025142052597174032, 59: 0.0025142052597174032, 43: 0.0025142052597174032}





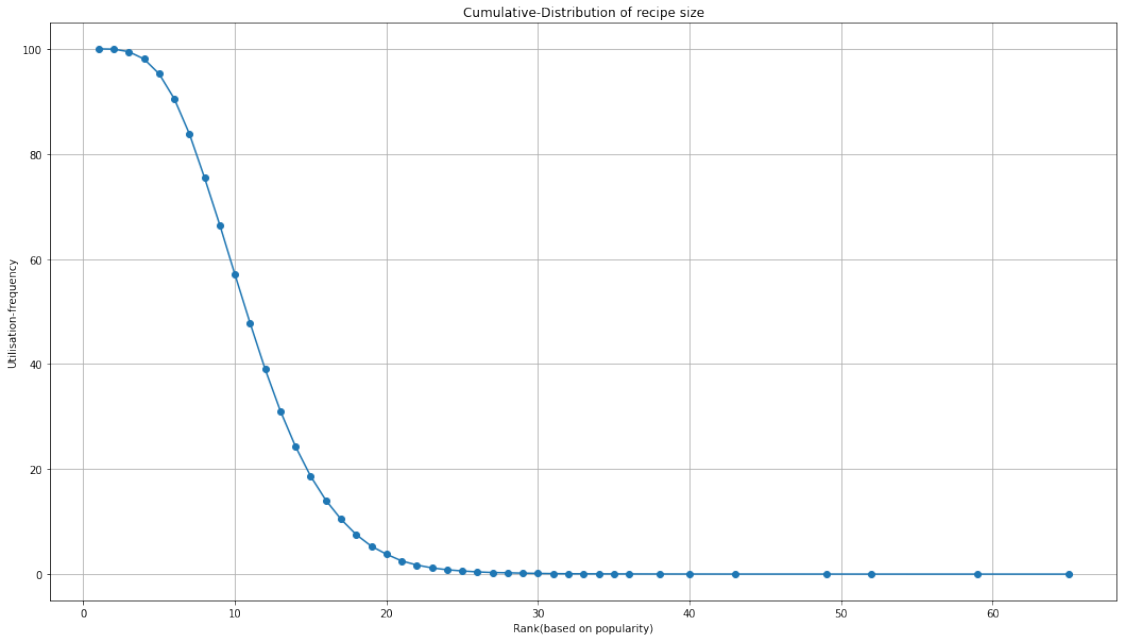
2.4 (d)

Plotting cumulative distribution of recipe size as given below:

[ ]:
```
recipe_size_dist = get_recipeDistCuisine(df)
get_RecipeSizeDist_Plot(recipe_size_dist)
```

```
Recipe Size Distribution :
 {9: 3753, 11: 3512, 12: 3146, 4: 1128, 20: 504, 13: 2698, 10: 3677, 6: 2662,
15: 1809, 17: 1160, 14: 2253, 5: 1891, 16: 1439, 8: 3556, 7: 3329, 22: 218, 3:
549, 2: 193, 18: 879, 24: 91, 38: 2, 21: 313, 40: 3, 23: 141, 19: 610, 25: 72,
26: 46, 27: 20, 1: 22, 34: 3, 28: 27, 30: 15, 29: 21, 32: 4, 36: 4, 49: 2, 31:
11, 35: 3, 65: 1, 33: 4, 52: 1, 59: 1, 43: 1}

Recipe Size Distribution in Percentage (%) :
 {9: 9.435812339719414, 11: 8.829888872127519, 12: 7.909689747070951, 4:
2.836023532961231, 20: 1.2671594508975714, 13: 6.783325790717553, 10:
9.244732739980892, 6: 6.692814401367728, 15: 4.548197314828783, 17:
2.916478101272188, 14: 5.664504450143309, 5: 4.75436214612561, 16:
3.6179413687333435, 8: 8.940513903555086, 7: 8.369789309599236, 22:
0.5480967466183939, 3: 1.3802986875848544, 2: 0.4852416151254589, 18:
2.2099864232915976, 24: 0.2287926786342837, 38: 0.0050284105194348064, 21:
0.7869462462915472, 40: 0.00754261577915221, 23: 0.3545029416201539, 19:
1.533665208427616, 25: 0.18102277869965305, 26: 0.11565344194700054, 27:
```

```
0.05028410519434806, 1: 0.05531251571378287, 34: 0.00754261577915221, 28:
0.06788354201236989, 30: 0.03771307889576105, 29: 0.05279831045406547, 32:
0.010056821038869613, 36: 0.010056821038869613, 49: 0.0050284105194348064, 31:
0.027656257856891436, 35: 0.00754261577915221, 65: 0.0025142052597174032, 33:
0.010056821038869613, 52: 0.0025142052597174032, 59: 0.0025142052597174032, 43:
0.0025142052597174032}
Cumulative Scores of recipe-sizes:  {65: 0.0025142052597174032, 59:
0.0050284105194348064, 52: 0.00754261577915221, 49: 0.012571026298587015, 43:
0.015085231558304418, 40: 0.022627847337456628, 38: 0.027656257856891436, 36:
0.037713078895761046, 35: 0.045255694674913256, 34: 0.05279831045406547, 33:
0.06285513149293508, 32: 0.07291195253180469, 31: 0.10056821038869612, 30:
0.13828128928445718, 29: 0.19107959973852265, 28: 0.2589631417508925, 27:
0.3092472469452406, 26: 0.42490068889224114, 25: 0.6059234675918942, 24:
0.8347161462261778, 23: 1.1892190878463318, 22: 1.7373158344647257, 21:
2.524262080756273, 20: 3.791421531653844, 19: 5.32508674008146, 18:
7.535073163373058, 17: 10.451551264645246, 16: 14.06949263337859, 15:
18.617689948207374, 14: 24.282194398350683, 13: 31.065520189068238, 12:
38.97520993613919, 11: 47.805098808266706, 10: 57.049831548247596, 9:
66.48564388796702, 8: 75.4261577915221, 7: 83.79594710112134, 6:
90.48876150248907, 5: 95.24312364861468, 4: 98.07914718157592, 3:
99.45944586916077, 2: 99.94468748428623, 1: 100.00000000000001}
```



From the above results we can interpret the following:

1. For smaller recipies sizes, the scattering is around maximum, therefore we can conclude that popularity of these recipie size is around maximum. Hence, they are easy to transmit due to their simplistic structure size.
2. From the cumulatives values we can conclude that there exists around 57% of recipes that

consist of size 10. Approximately 3% of recipes have size greater 20.
3. Slope from recipe-size 10 to 20 is very steep, thus after range, area under curve is less because there are only a few recipes that have size more than 20.
4. Observing the area under the curve, we can interpret that the area is more for recipe sizes less than 20 which means that most of the recipes consist of number of ingredients less than 20.

---

3 Answer to Question 3

For this question we prepare the functions to sequentially generate answers for each sub-question as follows:

[ ]:
```
#For frequency of each ingredient in case of Frequency-Rank Distribution
def get_ingredient_freq(df):
  ingf={} #to store ingredient-frequency in this dictionary
  count=0
  for recipe in df['ingredients']:
    for ingr in range(len(recipe)):
      if recipe[ingr] not in ingf:
        ingf[recipe[ingr]]=1
      else:
        ingf[recipe[ingr]]+=1
  #For computing most popular ingredient
  d = max(ingf.items(), key=operator.itemgetter(1))[1]
  for i in ingf:
    ingf[i] = ingf[i]/d #normalizing
  ingf_sort = sorted(ingf.items(),key=lambda x: x[1],reverse=True) #Sorting the␣
 ↪term frequency in descending order, So that most popular ingredient comes␣
 ↪first
  return ingf,ingf_sort
```

[ ]:
```
#For plotting Frequency-Rank distribution
def generate_Plot_FreqDist(ingf, ingf_sort):
  #For each ingredient out of 6715 ingredients
  fig = plt.figure(figsize=(18,10))
  X=np.arange(1,6715)
  Y=[]
  for i in range(len(ingf_sort)):
    Y.append(ingf_sort[i][1])
  plt.title("Frequency-Rank Distribution for all recipes")
  plt.xlabel("Rank")
  plt.ylabel("Frequency")
  plt.loglog(X,Y)
  plt.plot(X,Y,color='r')
```

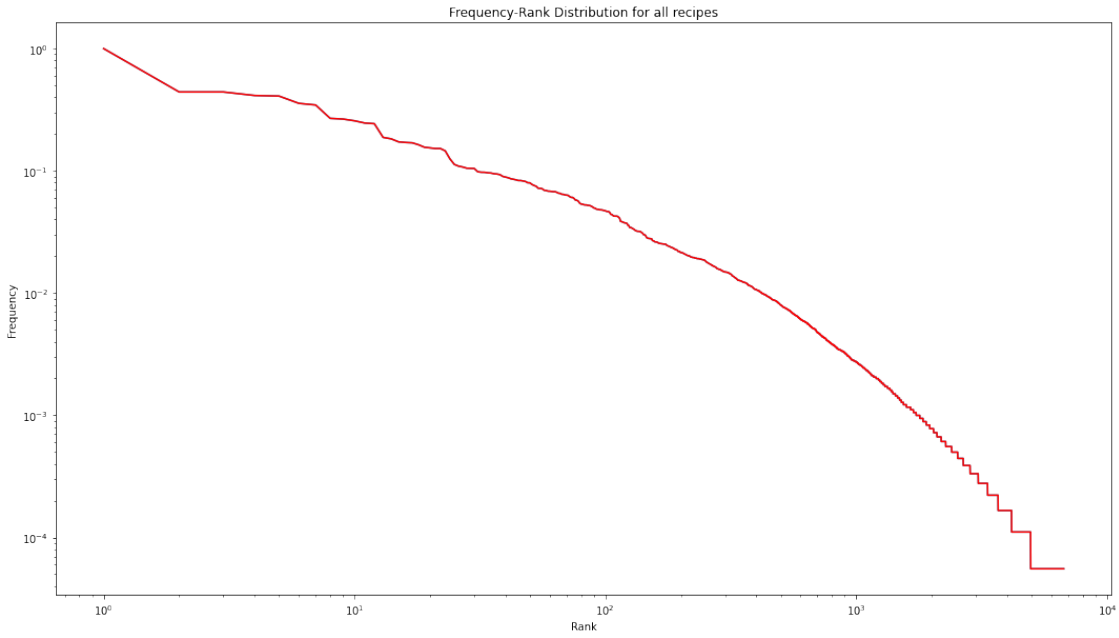
[ ]:
```
#For Top 10 most common/popular ingredients found in recipes
def get_top10_ingredients(ingf):
  top =  Counter(ingf)
  top_occur = top.most_common(10)
  print("Top 10 most-popular ingredients in recipe : \n",top_occur,"\n")
```

[ ]:
```
#For Ingredient Rank Distribution for each of the cuisines
def get_ingredientRank_distPlot(cuisine_recipeNum,df):
  ingr_rank={} #stores info of each cuisine having ingredient as key and␣
↪frequency as value in dictionary
  fig = plt.figure(figsize=(18,10))
  for cuisine in cuisine_recipeNum:
    cuis = df.loc[df['cuisine']==cuisine]
    li = cuis['ingredients'].tolist() #Converting into list format
    flat_list = [item for sublist in li for item in sublist] #flattening the␣
↪list
    #Computing ingredient count of each ingredient using Counter library
    ingr_count = dict(Counter(flat_list))
    denominator = max(ingr_count.items(), key=operator.itemgetter(1))[1] #Most␣
↪popular ingredient
    for i in ingr_count:
      ingr_count[i] = ingr_count[i]/denominator #normalizing the ingredient␣
↪count
    ingr_rank = dict(sorted(ingr_count.items(),key=lambda x:␣
↪x[1],reverse=True)) #Reverse Sorting most popular at first
    print("\nThe most popular ingredient for ",cuisine, "cuisine is : ",␣
↪max(ingr_rank.items(), key=operator.itemgetter(1)),", with Total Ingredients␣
↪: ", len(ingr_rank))
    #print("\n\n")
    X = np.arange(1,len(ingr_rank)+1)
    Y=list(ingr_rank.values())
    plt.loglog(X,Y)
    plt.plot(X,Y,label=cuisine)
  print("\n\n")
  plt.title("Ingredient-Rank Distribution for each cuisine")
  plt.xlabel("Ingredient-Rank")
  plt.ylabel("Ingredient-Frequency")
  plt.legend()
  plt.show()
```

3.1 (a)

Frequency-rank distribution plot for all the recipes:

[ ]:
```
ingf,ingf_sort = get_ingredient_freq(df)
generate_Plot_FreqDist(ingf,ingf_sort)
```



From this resultant plot we can conclude the following interpretation:
1. Salt is seen as the most common ingredient among almost all the recipes since salt is known to be the main ingredient to enhance aromas, attenuates or dampens the bitterness in dishes which helps reveal the vital taste of food.
2. The plot trend is seen to be exponentially decreasing but we also observe that the least value of the plot trend never reaches zero (0) level. Hence, from this plot-trend we can conclude that it follows "Power-Law Distribution".

3.2 (b)

The 10 most popular ingredients found in the recipes of this data are as listed below:

[ ]:
```
get_top10_ingredients(ingf)
```

```
Top 10 most-popular ingredients in recipe :
 [('salt', 1.0), ('onions', 0.4416865200288105), ('olive oil',
0.4416865200288105), ('water', 0.4131530832733115), ('garlic',
0.4088869189428777), ('sugar', 0.35647404288326223), ('garlic cloves',
0.34555931076513935), ('butter', 0.2686021386226384), ('ground black pepper',
0.26511164053410163), ('all-purpose flour', 0.25663471660479803)]
```

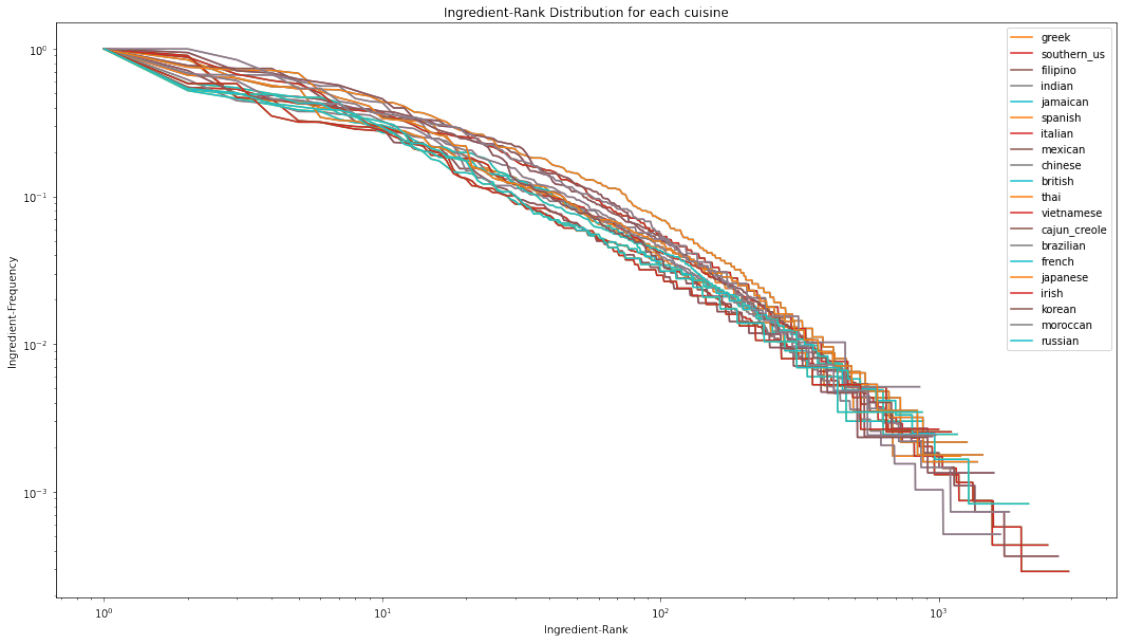
3.3 (c)

Listing the most popular ingredient for each cuisine and then plotting the ingredient-rank distribution per cuisine as follows:

[ ]:
```
get_ingredientRank_distPlot(recipe_cuisine,df)
```

```
The most popular ingredient for  greek cuisine is :  ('salt', 1.0) , with Total
Ingredients :  1198

The most popular ingredient for  southern_us cuisine is :  ('salt', 1.0) , with
Total Ingredients :  2462

The most popular ingredient for  filipino cuisine is :  ('salt', 1.0) , with
Total Ingredients :  947

The most popular ingredient for  indian cuisine is :  ('salt', 1.0) , with Total
Ingredients :  1664

The most popular ingredient for  jamaican cuisine is :  ('salt', 1.0) , with
Total Ingredients :  877

The most popular ingredient for  spanish cuisine is :  ('salt', 1.0) , with
Total Ingredients :  1263

The most popular ingredient for  italian cuisine is :  ('salt', 1.0) , with
Total Ingredients :  2929

The most popular ingredient for  mexican cuisine is :  ('salt', 1.0) , with
Total Ingredients :  2684

The most popular ingredient for  chinese cuisine is :  ('soy sauce', 1.0) , with
Total Ingredients :  1792

The most popular ingredient for  british cuisine is :  ('salt', 1.0) , with
Total Ingredients :  1166

The most popular ingredient for  thai cuisine is :  ('fish sauce', 1.0) , with
Total Ingredients :  1376

The most popular ingredient for  vietnamese cuisine is :  ('fish sauce', 1.0) ,
with Total Ingredients :  1108

The most popular ingredient for  cajun_creole cuisine is :  ('salt', 1.0) , with
Total Ingredients :  1576

The most popular ingredient for  brazilian cuisine is :  ('salt', 1.0) , with
Total Ingredients :  853
```

```
The most popular ingredient for  french cuisine is :  ('salt', 1.0) , with Total
Ingredients :  2102

The most popular ingredient for  japanese cuisine is :  ('soy sauce', 1.0) ,
with Total Ingredients :  1439

The most popular ingredient for  irish cuisine is :  ('salt', 1.0) , with Total
Ingredients :  999

The most popular ingredient for  korean cuisine is :  ('soy sauce', 1.0) , with
Total Ingredients :  898

The most popular ingredient for  moroccan cuisine is :  ('salt', 1.0) , with
Total Ingredients :  974

The most popular ingredient for  russian cuisine is :  ('salt', 1.0) , with
Total Ingredients :  872
```



3.4 (d)

From the above list of most popular ingredient in each cuisine and the plot as shown above, we can interpret that:
1. We can see that salt is the most populr ingredient that is present in almost 15 out of 20 recipes. Other popular ingredients include soy sauce and fish sauce, where soy sauce is the most frequenty used ingredient in Chinese, Japanese and Korean cuisines and fish sauce is used most frequently in Thai and Viatnamese cuisines.
2. The cuisine with the utilisation of most number of ingredients is the Italian Cuisine.
3. The plots show a monotonous decreasing trend, which indicates the fact that the recipes and cuisines follow a uniform use of each ingredient in its dishes.
4. No ingredient in the above plot has a zero frequency since the least frequent ingredient may atleast have been used once or a few number of times.
5. All 20 cuisines interestingly follow the same power-law distribution as also discussed in the previous plot of frequency-rank distribution. This hints towards the most frequent use of certain ingredients in each of the cuisines despite each cuisine having distinct blend of ingredients.

---

4 Downloading the notebook as PDF

[ ]:
```
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('CGAS_Assignment1_MT20075.ipynb')
```

```
--2021-09-10 15:33:04--  https://raw.githubusercontent.com/brpy/colab-
pdf/master/colab_pdf.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)...
185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1864 (1.8K) [text/plain]
Saving to: colab_pdf.py

colab_pdf.py        100%[===================>]   1.82K  --.-KB/s    in 0s

2021-09-10 15:33:05 (32.4 MB/s) - colab_pdf.py saved [1864/1864]


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.
```

```
Extracting templates from packages: 100%
```

[ ]: