P2R_T5

April 30, 2024

[1]:
```
from nlp.recipes_utils import *
import sys
import os
import ast
import random
import pandas as pd
import numpy as np
import pickle

from matplotlib import pyplot as plt
import tensorflow as tf

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics.pairwise import cosine_similarity

import torch

# !pip install bayesian-optimization
from bayes_opt import BayesianOptimization
```

[2]:
```
print("Python version: ", sys.version)
print(sys.version_info)
```

```
Python version:  3.8.19 (default, Mar 20 2024, 15:27:52)
[Clang 14.0.6 ]
sys.version_info(major=3, minor=8, micro=19, releaselevel='final', serial=0)
```

[ ]:
```
device = set_device()  # Set device to maximize performance
```

[ ]:
```
clear_memory(device)  # Clear unused memory
```

[2]:
```
# make sure ./data/recipes.pickle exists
!python -c 'from recipes_utils import *;␣
 ↪split_and_save_recipe_datasets(preprocess=True)'
```

```
Traceback (most recent call last):
  File "<string>", line 1, in <module>
  File "/Users/nc/Library/CloudStorage/OneDrive-
```

```
Personal/gt/DL/final_project_p2r/nlp/recipes_utils.py", line 156, in
split_and_save_recipe_datasets
    recipes = process_recipe_data(recipes, MAX_TEXT_LENGTH=500, filter=True)
  File "/Users/nc/Library/CloudStorage/OneDrive-
Personal/gt/DL/final_project_p2r/nlp/recipes_utils.py", line 138, in
process_recipe_data
    recipe = filter_by_text_length(recipe, 'combined', MAX_TEXT_LENGTH)
  File "/Users/nc/Library/CloudStorage/OneDrive-
Personal/gt/DL/final_project_p2r/nlp/recipes_utils.py", line 126, in
filter_by_text_length
    data_filtered = data[data[column].apply(len) <= MAX_TEXT_LENGTH]
  File "/Applications/anaconda3/envs/p2r/lib/python3.8/site-
packages/pandas/core/series.py", line 4630, in apply
    return SeriesApply(self, func, convert_dtype, args, kwargs).apply()
  File "/Applications/anaconda3/envs/p2r/lib/python3.8/site-
packages/pandas/core/apply.py", line 1025, in apply
    return self.apply_standard()
  File "/Applications/anaconda3/envs/p2r/lib/python3.8/site-
packages/pandas/core/apply.py", line 1076, in apply_standard
    mapped = lib.map_infer(
  File "pandas/_libs/lib.pyx", line 2834, in pandas._libs.lib.map_infer
TypeError: object of type 'float' has no len()
```

[111]:
```python
# RECIPE DATASET PROCESSING
DIR_DATA_PATH = '../data/recipes'
FULL_RECIPES_PATH = f'{DIR_DATA_PATH}/full_dataset.csv'
FULL_RECIPES_PICKLE_PATH = f'../data/recipes.pickle'
TRAIN_PICKLE_PATH = f'{DIR_DATA_PATH}/train.pickle'
VAL_PICKLE_PATH = f'{DIR_DATA_PATH}/val.pickle'
TEST_PICKLE_PATH = f'{DIR_DATA_PATH}/test.pickle'

KEEP_COLUMNS = ['title', 'ingredients', 'directions', 'NER']

# Special markers to separate the recipe components
MARKER_TITLE = '<t>'
MARKER_INGREDIENTS = '<i>'
MARKER_DIRECTIONS = '<d>'
MARKER_STOP = '_'
MAX_TEXT_LENGTH = 500
```

[5]:
```python
full_dataset = load_pickle(FULL_RECIPES_PICKLE_PATH, columns=COLUMNS)
```

[29]:
```python
# recipes = process_recipe_data(full_dataset)
full_dataset = full_dataset.dropna()
full_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2231141 entries, 0 to 2231141
```

```
Data columns (total 4 columns):
 #   Column       Dtype
---  ------       -----
 0   title        object
 1   ingredients  object
 2   directions   object
 3   NER          object
dtypes: object(4)
memory usage: 85.1+ MB
```

[114]:
```
def transform_list_to_string(list_str, as_list=True):
    # Parse the string as a list
    parsed_list = ast.literal_eval(list_str)
    # Format as a bulleted list or join with semicolons based on the flag
    if as_list:
        return "\n".join(f"- {item}" for item in parsed_list)
    else:
        return " ".join(parsed_list)


def recipe_to_str(data):
    # Apply the transformation to ingredients and directions with list␣
 ↪formatting
    data['ingredients_string'] = data['ingredients'].apply(lambda x:␣
 ↪transform_list_to_string(x, as_list=True))
    data['directions_string'] = data['directions'].apply(lambda x:␣
 ↪transform_list_to_string(x, as_list=False))

    # Create a new column by concatenating title, ingredients, and directions␣
 ↪with specific markers
    data['combined'] = (MARKER_TITLE + "Recipe: " + data['title'] + "\n\n" +
                        MARKER_INGREDIENTS + "Ingredients:\n" +␣
 ↪data['ingredients_string'] + "\n\n" +
                        MARKER_DIRECTIONS + "Directions: " +␣
 ↪data['directions_string'])

    # Drop intermediate columns used for processing
    data.drop(['ingredients_string', 'directions_string'], axis=1, inplace=True)

    return data
```

[115]:
```
recipes = full_dataset[KEEP_COLUMNS]
recipes = recipe_to_str(recipes)
recipes.head()
```

[115]:
```
                  title                                        ingredients  \
0    No-Bake Nut Cookies  ["1 c. firmly packed brown sugar", "1/2 c. eva…
```

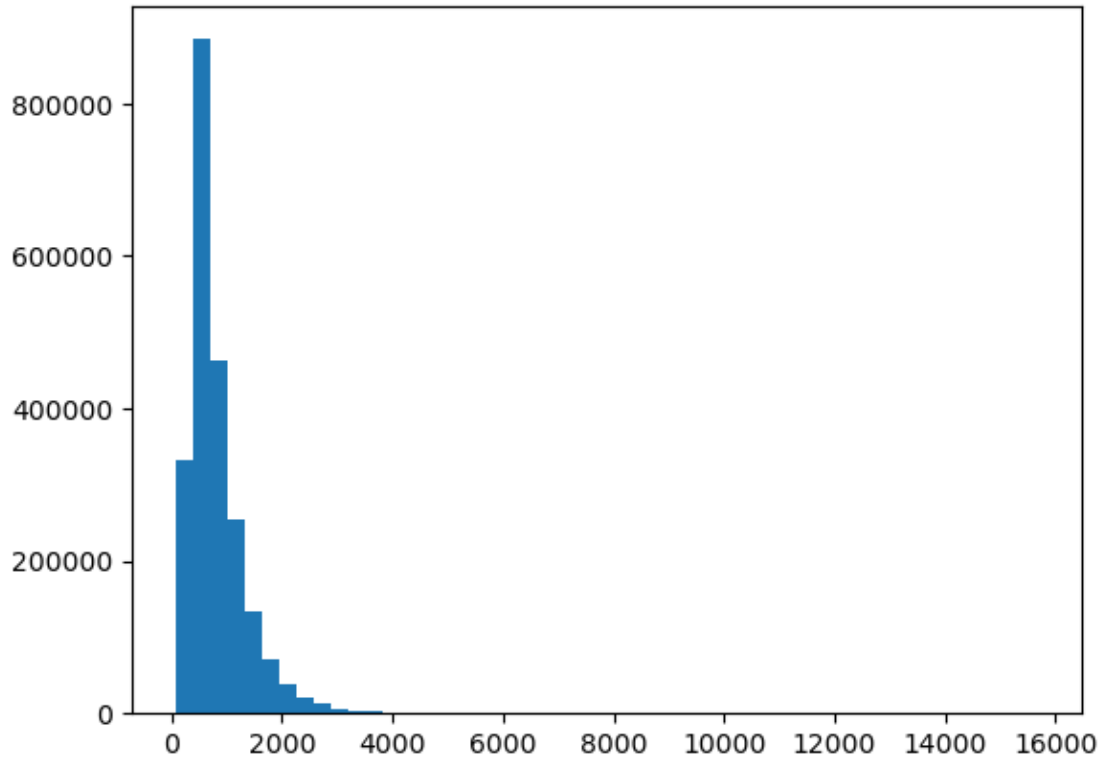
```
1  Jewell Ball'S Chicken  ["1 small jar chipped beef, cut up", "4 boned …
2            Creamy Corn  ["2 (16 oz.) pkg. frozen corn", "1 (8 oz.) pkg…
3          Chicken Funny  ["1 large whole chicken", "2 (10 1/2 oz.) cans…
4   Reeses Cups(Candy)    ["1 c. peanut butter", "3/4 c. graham cracker …

                                          directions  \
0  ["In a heavy 2-quart saucepan, mix brown sugar…
1  ["Place chipped beef on bottom of baking dish…
2  ["In a slow cooker, combine all ingredients. C…
3  ["Boil and debone chicken.", "Put bite size pi…
4  ["Combine first four ingredients and press in …

                                                NER  \
0  ["brown sugar", "milk", "vanilla", "nuts", "bu…
1  ["beef", "chicken breasts", "cream of mushroom…
2  ["frozen corn", "cream cheese", "butter", "gar…
3  ["chicken", "chicken gravy", "cream of mushroo…
4  ["peanut butter", "graham cracker crumbs", "bu…

                                           combined
0  <t>Recipe: No-Bake Nut Cookies\n\n<i>Ingredien…
1  <t>Recipe: Jewell Ball'S Chicken\n\n<i>Ingredi…
2  <t>Recipe: Creamy Corn\n\n<i>Ingredients:\n- 2…
3  <t>Recipe: Chicken Funny\n\n<i>Ingredients:\n-…
4  <t>Recipe: Reeses Cups(Candy)  \n\n<i>Ingredie…
```

[116]:
```
# Sample Recipe as string
print(recipes['combined'][0])
```

```
<t>Recipe: No-Bake Nut Cookies

<i>Ingredients:
- 1 c. firmly packed brown sugar
- 1/2 c. evaporated milk
- 1/2 tsp. vanilla
- 1/2 c. broken nuts (pecans)
- 2 Tbsp. butter or margarine
- 3 1/2 c. bite size shredded rice biscuits

<d>Directions: In a heavy 2-quart saucepan, mix brown sugar, nuts, evaporated
milk and butter or margarine. Stir over medium heat until mixture bubbles all
over top. Boil and stir 5 minutes more. Take off heat. Stir in vanilla and
cereal; mix well. Using 2 teaspoons, drop and shape into 30 clusters on wax
paper. Let stand until firm, about 30 minutes.
```

[117]:
```
# Check which recipe length is best for sequence length limit
recipe_lengths = [len(recipe_text) for recipe_text in recipes["combined"]]
plt.hist(recipe_lengths, bins=50)
```

```
plt.show()
```



[118]:
```
# Filter data based on the maximum text length for specified column. We keep␣
 ↪those with MAX_TEXT_LENGTH
recipes_filtered = recipes[recipes['combined'].apply(len) <= MAX_TEXT_LENGTH]
print("Before filtering: ", len(full_dataset))
print("After filtering: ", len(recipes_filtered))
```

```
Before filtering:  2231141
After filtering:  736028
```

[119]:
```
##########################################################################################
# RUN if ./data/recipes/ does not exist
##########################################################################################
# Split the dataset into train, val and split
# train_dataset, val_dataset, test_dataset = split_data(recipes, train_size=0.
 ↪7, val_size=0.15, test_size=0.15)
#
# # Save the datasets to files
# if not os.path.exists(DIR_DATA_PATH):
#     os.makedirs(DIR_DATA_PATH)
#
```

```
# save_pickle(train_dataset, TRAIN_PICKLE_PATH)
# save_pickle(val_dataset, VAL_PICKLE_PATH)
# save_pickle(test_dataset, TEST_PICKLE_PATH)
# save_pickle(recipes_filtered, DIR_DATA_PATH + '/recipes_processed.pickle')
#
##########################################################################################
```

[130]:
```
# Load only a % of entries
# Note: 0.001 approx. 1500+ entries; 0.0001% = 155/33/33; 0.00005% = 78/16/16

keep_pct = 0.0001
# train, val, test = load_splits(TRAIN_PICKLE_PATH, VAL_PICKLE_PATH,␣
 ↪TEST_PICKLE_PATH, keep_pct=pct)

train = load_pickle(TRAIN_PICKLE_PATH)
val = load_pickle(VAL_PICKLE_PATH)
test = load_pickle(TEST_PICKLE_PATH)

train = train.sample(frac=keep_pct, random_state=42)
val = val.sample(frac=keep_pct, random_state=42)
test = test.sample(frac=keep_pct, random_state=42)

print(f'MAX_RECIPE_LENGTH: {MAX_TEXT_LENGTH}')
print(f'TOTAL RECIPES: {len(train) + len(val) + len(test)}')
print(f"- Train: {len(train)} \n"
      f"- Val: {len(val)} \n"
      f"- Test: {len(test)}")
```

```
MAX_RECIPE_LENGTH: 500
TOTAL RECIPES: 222
- Train: 156
- Val: 33
- Test: 33
```

[131]:
```
# only keep column with recipe texts
train = train["combined"]
val = val['combined']
test = test['combined']
```

https://www.tensorflow.org/text/tutorials/text_generation

0.1 TOKEN

[132]:
```
tokenizer = tf.keras.preprocessing.text.Tokenizer(
    char_level=True,
    filters='',
    lower=False,
    split=''
```

```
)

tokenizer.fit_on_texts([MARKER_STOP])
tokenizer.fit_on_texts(train)
tokenizer.get_config()
```

[132]:

```
{'num_words': None,
 'filters': '',
 'lower': False,
 'split': '',
 'char_level': True,
 'oov_token': None,
 'document_count': 157,
 'word_counts': '{"_": 1, "<": 468, "t": 4970, ">": 468, "R": 239, "e": 8297,
"c": 3064, "i": 5153, "p": 2444, ":": 484, " ": 15378, "C": 309, "r": 4618, "a":
5122, "n": 5038, "b": 1221, "y": 554, "l": 3237, "s": 4235, "h": 2061, "\\n":
1827, "I": 241, "g": 1899, "d": 3039, "-": 1371, "1": 1116, "o": 5170, "2": 596,
"u": 2206, "/": 444, ".": 1652, "x": 247, "w": 790, "J": 17, "O": 27, "k": 860,
"(": 187, ")": 188, "D": 223, "W": 51, ",": 786, "P": 191, "G": 39, "m": 1547,
"A": 185, "f": 887, "S": 280, "4": 261, "3": 290, "v": 603, "T": 143, ";": 84,
"M": 137, "B": 177, "0": 164, "Y": 10, "\'": 25, "F": 88, "q": 69, "j": 125,
"6": 76, "z": 149, "H": 41, "N": 8, "5": 147, "\\u00b0": 50, "7": 18, "[": 4,
"]": 4, "K": 12, "L": 56, "&": 13, "U": 10, "8": 74, "9": 29, "Q": 5, "E": 26,
"V": 17, "!": 6, "Z": 3, "\\"": 10, "#": 3, "%": 7, "\\u2019": 1, "\\t": 6}',
 'word_docs': '{"_": 1, "i": 156, ")": 93, "t": 156, "y": 135, "e": 156, "C":
135, "g": 156, ".": 155, "k": 150, " ": 156, "P": 99, "W": 40, "d": 156, "s":
156, ",": 131, "<": 156, "b": 150, "O": 16, "A": 95, "n": 156, "w": 146, "h":
156, "\\n": 156, "-": 156, "2": 141, "1": 153, "x": 118, "f": 146, "r": 156,
"a": 156, "l": 155, "p": 156, "D": 156, "(": 92, "I": 156, "R": 156, "m": 156,
"G": 35, "o": 156, "/": 130, "u": 156, ":": 156, "J": 14, ">": 156, "c": 156,
"v": 136, "S": 117, ";": 49, "T": 81, "3": 114, "4": 115, "Y": 10, "0": 84, "B":
96, "M": 84, "q": 36, "\'": 21, "j": 62, "F": 59, "6": 58, "\\u00b0": 46, "5":
81, "N": 8, "7": 16, "z": 67, "H": 30, "]": 1, "[": 1, "L": 42, "K": 7, "&": 7,
"U": 9, "8": 53, "9": 25, "Q": 5, "E": 14, "V": 10, "!": 5, "Z": 3, "\\"": 6,
"#": 1, "%": 2, "\\u2019": 1, "\\t": 3}',
 'index_docs': '{"1": 156, "84": 1, "4": 156, "43": 93, "7": 156, "29": 135,
"2": 156, "34": 135, "16": 156, "18": 155, "24": 150, "42": 99, "59": 40, "12":
156, "9": 156, "26": 131, "31": 156, "21": 150, "64": 16, "45": 95, "6": 156,
"25": 146, "15": 156, "17": 156, "20": 156, "28": 141, "22": 153, "38": 118,
"23": 146, "8": 156, "5": 156, "10": 155, "13": 156, "41": 156, "44": 92, "39":
156, "40": 156, "19": 156, "62": 35, "3": 156, "33": 130, "14": 156, "30": 156,
"68": 14, "32": 156, "11": 156, "27": 136, "36": 117, "54": 49, "50": 81, "35":
114, "37": 115, "72": 10, "47": 84, "46": 96, "51": 84, "57": 36, "66": 21,
"52": 62, "53": 59, "55": 58, "60": 46, "49": 81, "75": 8, "67": 16, "48": 67,
"61": 30, "81": 1, "80": 1, "58": 42, "71": 7, "70": 7, "73": 9, "56": 53, "63":
25, "79": 5, "65": 14, "69": 10, "77": 5, "82": 3, "74": 6, "83": 1, "76": 2,
"85": 1, "78": 3}',
```

```
 'index_word': '{"1": " ", "2": "e", "3": "o", "4": "i", "5": "a", "6": "n",
"7": "t", "8": "r", "9": "s", "10": "l", "11": "c", "12": "d", "13": "p", "14":
"u", "15": "h", "16": "g", "17": "\\n", "18": ".", "19": "m", "20": "-", "21":
"b", "22": "1", "23": "f", "24": "k", "25": "w", "26": ",", "27": "v", "28":
"2", "29": "y", "30": ":", "31": "<", "32": ">", "33": "/", "34": "C", "35":
"3", "36": "S", "37": "4", "38": "x", "39": "I", "40": "R", "41": "D", "42":
"P", "43": ")", "44": "(", "45": "A", "46": "B", "47": "0", "48": "z", "49":
"5", "50": "T", "51": "M", "52": "j", "53": "F", "54": ";", "55": "6", "56":
"8", "57": "q", "58": "L", "59": "W", "60": "\\u00b0", "61": "H", "62": "G",
"63": "9", "64": "O", "65": "E", "66": "\'", "67": "7", "68": "J", "69": "V",
"70": "&", "71": "K", "72": "Y", "73": "U", "74": "\\"", "75": "N", "76": "%",
"77": "!", "78": "\\t", "79": "Q", "80": "[", "81": "]", "82": "Z", "83": "#",
"84": "_", "85": "\\u2019"}',
 'word_index': '{" ": 1, "e": 2, "o": 3, "i": 4, "a": 5, "n": 6, "t": 7, "r": 8,
"s": 9, "l": 10, "c": 11, "d": 12, "p": 13, "u": 14, "h": 15, "g": 16, "\\n":
17, ".": 18, "m": 19, "-": 20, "b": 21, "1": 22, "f": 23, "k": 24, "w": 25, ",":
26, "v": 27, "2": 28, "y": 29, ":": 30, "<": 31, ">": 32, "/": 33, "C": 34, "3":
35, "S": 36, "4": 37, "x": 38, "I": 39, "R": 40, "D": 41, "P": 42, ")": 43, "(":
44, "A": 45, "B": 46, "0": 47, "z": 48, "5": 49, "T": 50, "M": 51, "j": 52, "F":
53, ";": 54, "6": 55, "8": 56, "q": 57, "L": 58, "W": 59, "\\u00b0": 60, "H":
61, "G": 62, "9": 63, "O": 64, "E": 65, "\'": 66, "7": 67, "J": 68, "V": 69,
"&": 70, "K": 71, "Y": 72, "U": 73, "\\"": 74, "N": 75, "%": 76, "!": 77, "\\t":
78, "Q": 79, "[": 80, "]": 81, "Z": 82, "#": 83, "_": 84, "\\u2019": 85}'}
```

[133]:
```
# @see: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/
  ↪Tokenizer
VOCABULARY_SIZE = len(tokenizer.word_counts) + 1
print('VOCABULARY_SIZE: ', VOCABULARY_SIZE)
```

```
VOCABULARY_SIZE:  86
```

[134]:
```
print(tokenizer.index_word[5])
tokenizer.word_index['s']
```

```
a
```

[134]:
```
9
```

[135]:
```
js_vocabulary = tokenizer.sequences_to_texts([[word_index] for word_index in␣
  ↪range(VOCABULARY_SIZE)])
print([char for char in js_vocabulary])
```

```
['', ' ', 'e', 'o', 'i', 'a', 'n', 't', 'r', 's', 'l', 'c', 'd', 'p', 'u', 'h',
'g', '\n', '.', 'm', '-', 'b', '1', 'f', 'k', 'w', ',', 'v', '2', 'y', ':', '<',
'>', '/', 'C', '3', 'S', '4', 'x', 'I', 'R', 'D', 'P', ')', '(', 'A', 'B', '0',
'z', '5', 'T', 'M', 'j', 'F', ';', '6', '8', 'q', 'L', 'W', '°', 'H', 'G', '9',
'O', 'E', "'", '7', 'J', 'V', '&', 'K', 'Y', 'U', '"', 'N', '%', '!', '\t', 'Q',
'[', ']', 'Z', '#', '_', '’']
```

0.2 VECTORIZING FOR RNN

[136]:
```
def recipe_sequence_to_string(recipe_sequence):
    recipe_str = tokenizer.sequences_to_texts([recipe_sequence])[0]
    recipe_str = recipe_str.replace('   ', '_').replace(' ', '').replace('_', '␣
 ↪')
    print(recipe_str)
```

[137]:
```
train_vectorized = tokenizer.texts_to_sequences(train)
print('Vectorized train size', len(train_vectorized))
print(train_vectorized[0][:10], '...')
```

```
Vectorized train size 156
[31, 7, 32, 40, 2, 11, 4, 13, 2, 30] …
```

[138]:
```
recipe_sequence_to_string(train_vectorized[1])
```

```
<t>Recipe: Country Apple Slaw

<i>Ingredients:
- 2 Granny Smith apples, thinly sliced
- 4 cups shredded green cabbage (about 1/2 head)
- 4 cups shredded red cabbage (about 1/2 head)
- 1 cucumber, seeded and sliced
- 3/4 cup dried cherries
- 1/2 cup cider vinegar
- 1/3 cup honey
- 1/3 cup olive oil
- 2 teaspoons salt
- 1/2 teaspoon freshly ground black pepper
- 1/4 cup roasted, salted pepitos (shelled pumpkin seeds)

<d>Directions: Combine apples and next 4 ingredients in a large bowl. Whisk
together vinegar and next 4 ingredients in a medium bowl. Toss vinaigrette with
apple mixture; cover and chill up to 4 hours. Sprinkle with pepitos.
```

0.3 PADDING

[139]:
```
for i, recipe in enumerate(train_vectorized[:5]):
    print('Recipe #{} length: {}'.format(i + 1, len(recipe)))
```

```
Recipe #1 length: 429
Recipe #2 length: 644
Recipe #3 length: 386
Recipe #4 length: 292
Recipe #5 length: 999
```

[140]:
```

```

```
# Use -1 and +1 to make sure that all recipes will have at least 1 stop sign at␣
 ↪the end, since each sequence will be shifted and truncated afterwards (to␣
 ↪generate X and Y sequences).

trainVect_padded_no_stops = tf.keras.preprocessing.sequence.pad_sequences(
    train_vectorized,
    padding='post',
    truncating='post',
    maxlen=MAX_TEXT_LENGTH - 1,
    value=tokenizer.texts_to_sequences([MARKER_STOP])[0]
)

trainVect_padded = tf.keras.preprocessing.sequence.pad_sequences(
    trainVect_padded_no_stops,
    padding='post',
    truncating='post',
    maxlen=MAX_TEXT_LENGTH + 1,
    value=tokenizer.texts_to_sequences([MARKER_STOP])[0]
)
```

[141]:
```
for i, recipe in enumerate(trainVect_padded[:5]):
    print('Recipe #{} length: {}'.format(i, len(recipe)))
```

```
Recipe #0 length: 501
Recipe #1 length: 501
Recipe #2 length: 501
Recipe #3 length: 501
Recipe #4 length: 501
```

[142]:
```
recipe_sequence_to_string(trainVect_padded[1])
```

```
<t>Recipe: Country Apple Slaw

<i>Ingredients:
- 2 Granny Smith apples, thinly sliced
- 4 cups shredded green cabbage (about 1/2 head)
- 4 cups shredded red cabbage (about 1/2 head)
- 1 cucumber, seeded and sliced
- 3/4 cup dried cherries
- 1/2 cup cider vinegar
- 1/3 cup honey
- 1/3 cup olive oil
- 2 teaspoons salt
- 1/2 teaspoon freshly ground black pepper
- 1/4 cup roasted, salted pepitos (shelled pumpkin seeds)

<d>Directions: Combine apples and next 4 ingredients in a large bowl. Whisk tog
```

[143]:
```
dataset = tf.data.Dataset.from_tensor_slices(trainVect_padded)
print(dataset)
```

```
<_TensorSliceDataset element_spec=TensorSpec(shape=(501,), dtype=tf.int32,
name=None)>
```

[144]:
```
for recipe in dataset.take(1):
    print('Raw Recipe:\n', recipe.numpy(), '\n\n\n')
    print('Text Recipe:\n')
    recipe_sequence_to_string(recipe.numpy())
```

```
Raw Recipe:
 [31  7 32 40  2 11  4 13  2 30  1 34  8  5  6 21  2  8  8 29  1 40  2 10
  4  9 15 17 17 31  4 32 39  6 16  8  2 12  4  2  6  7  9 30 17 20  1 22
  1 13  2  2 10  2 12  1  3  8  5  6 16  2 17 20  1 28  1  5 13 13 10  2
  9 17 20  1 22  1 11  5  6  1 11  8 14  9 15  2 12  1 13  4  6  2  5 13
 13 10  2 17 20  1 28  1 22 33 28  1 11 18  1  9 14 16  5  8 17 20  1 22
  1 10  5  8 16  2  1 21  3 38  1  9  7  8  5 25 21  2  8  8 29  1 68  2
 10 10 20 64 17 20  1 22  1 10 21 18  1 13 24 16 18  1 11  8  5  6 21  2
  8  8  4  2  9  1 44  8  5 25 43 17 20  1 28  1 11 18  1 21  3  4 10  4
  6 16  1 25  5  7  2  8 17 17 31 12 32 41  4  8  2 11  7  4  3  6  9 30
  1 59  5  9 15  1  5  6 12  1 16  8  4  6 12  1 11  8  5  6 21  2  8  8
  4  2  9 18  1 34  3  8  2  1  5 13 13 10  2  9 26  1 21 14  7  1 12  3
  1  6  3  7  1 13  2  2 10 18  1 42  2  2 10  1  3  8  5  6 16  2 18  1
 62  8  4  6 12  1  7 15  2 19  1 25  4  7 15  1 11  8  5  6 21  2  8  8
  4  2  9 18  1 41  8  5  4  6  1 13  4  6  2  5 13 13 10  2 18  1 45 12
 12  1  7  3  1  8  2  9  7 18  1 45 12 12  1  9 14 16  5  8 18  1 45 12
 12  1 28  1 11 14 13  9  1 21  3  4 10  4  6 16  1 25  5  7  2  8  1  7
  3  1 68  2 10 10 20 64 26  1  7 15  2  6  1  5 12 12  1  5 10 10  1  7
  3 16  2  7 15  2  8 18  1 40  2 23  8  4 16  2  8  5  7  2 18 84 84 84
 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84
 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84
 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84 84]



Text Recipe:

<t>Recipe: Cranberry Relish

<i>Ingredients:
- 1 peeled orange
- 2 apples
- 1 can crushed pineapple
- 2 1/2 c. sugar
- 1 large box strawberry Jell-O
- 1 lb. pkg. cranberries (raw)
- 2 c. boiling water
```

```
<d>Directions: Wash and grind cranberries. Core apples, but do not peel. Peel
orange. Grind them with cranberries. Drain pineapple. Add to rest. Add sugar.
Add 2 cups boiling water to Jell-O, then add all together. Refrigerate.
```

0.4 Split INPUT and TARGET texts

Each sequence is duplicated and shifted

[145]:
```
def split_input_target(recipe):
    input_text = recipe[:-1]
    target_text = recipe[1:]

    return input_text, target_text
```

[161]:
```
dataset_target = dataset.map(split_input_target)
print(dataset_target)

split_input_target(list("Recipe"))
```

```
<_MapDataset element_spec=(TensorSpec(shape=(500,), dtype=tf.int32, name=None),
TensorSpec(shape=(500,), dtype=tf.int32, name=None))>
```

[161]:
```
(['R', 'e', 'c', 'i', 'p'], ['e', 'c', 'i', 'p', 'e'])
```

[162]:
```
for input_example, target_example in dataset_target.take(1):
    print('Input sequence size:', repr(len(input_example.numpy())))
    print('Target sequence size:', repr(len(target_example.numpy())))
    print()

    input_string = tokenizer.sequences_to_texts([input_example.numpy()[:50]])[0]
    target_string = tokenizer.sequences_to_texts([target_example.numpy()[:
 ↪50]])[0]

    print('Input:  ', repr(''.join(input_string)))
    print('Target: ', repr(''.join(target_string)))
```

```
Input sequence size: 500
Target sequence size: 500

Input:   '< t > R e c i p e :   C r a n b e r r y   R e l i s h \n \n < i > I n
g r e d i e n t s : \n -   1   p'
Target:  't > R e c i p e :   C r a n b e r r y   R e l i s h \n \n < i > I n g
r e d i e n t s : \n -   1   p e'
```

[163]:
```
for i, (input_idx, target_idx) in enumerate(zip(input_example[:10],␣
 ↪target_example[:10])):
    print('Step {:2d}'.format(i + 1))
```

```
    print('  input: {} ({:s})'.format(input_idx, repr(tokenizer.
 ↪sequences_to_texts([[input_idx.numpy()]])[0])))
    print('  expected output: {} ({:s})'.format(target_idx,
                                                repr(tokenizer.
 ↪sequences_to_texts([[target_idx.numpy()]])[0])))
```

```
Step  1
  input: 31 ('<')
  expected output: 7 ('t')
Step  2
  input: 7 ('t')
  expected output: 32 ('>')
Step  3
  input: 32 ('>')
  expected output: 40 ('R')
Step  4
  input: 40 ('R')
  expected output: 2 ('e')
Step  5
  input: 2 ('e')
  expected output: 11 ('c')
Step  6
  input: 11 ('c')
  expected output: 4 ('i')
Step  7
  input: 4 ('i')
  expected output: 13 ('p')
Step  8
  input: 13 ('p')
  expected output: 2 ('e')
Step  9
  input: 2 ('e')
  expected output: 30 (':')
Step 10
  input: 30 (':')
  expected output: 1 (' ')
```

0.5 SPLIT IN BATCHES

Before feeding this data into the model, shuffle the data and pack it into batches.

[164]:
```
BATCH_SIZE = 32
SHUFFLE_BUFFER_SIZE = 1000

dataset_train = (dataset_target
                 .shuffle(SHUFFLE_BUFFER_SIZE)
                 .batch(BATCH_SIZE, drop_remainder=True)
                 .repeat())
```

```
print(dataset_train)
```

```
<_RepeatDataset element_spec=(TensorSpec(shape=(32, 500), dtype=tf.int32,
name=None), TensorSpec(shape=(32, 500), dtype=tf.int32, name=None))>
```

0.6 BUILD MODEL

@see https://www.tensorflow.org/text/tutorials/text_generation

Using tf.keras.Sequential we define: - tf.keras.layers.Embedding: The input layer. A trainable lookup table, maps each character's numbers to a vector with embedding_dim dimensions; - tf.keras.layers.LSTM - tf.keras.layers.Dense: The output layer, with vocab_size outputs.

The model will take as input an integer matrix of size (batch, input_length). The largest integer (word index) in the input should be no larger than tmp_vocab_size. Now model.output_shape == (batch_dimension, tmp_vocab_size, batch_size)??

[165]:
```
vocab_size = VOCABULARY_SIZE  # length of the vocabulary in chars
embedding_dim = 256
rnn_units = 1024
```

[166]:
```
def build_model_1(vocab_size, embedding_dim, rnn_units, batch_size):
    model = tf.keras.models.Sequential()

    model.add(tf.keras.layers.Embedding(
        input_dim=vocab_size,
        output_dim=embedding_dim,
        batch_input_shape=[batch_size, None]
    ))

    model.add(tf.keras.layers.LSTM(
        units=rnn_units,
        return_sequences=True,
        stateful=True,
        recurrent_initializer=tf.keras.initializers.GlorotNormal()
    ))

    model.add(tf.keras.layers.Dense(vocab_size))

    return model
```

[167]:
```
model_1 = build_model_1(vocab_size, embedding_dim, rnn_units, BATCH_SIZE)
model_1.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (32, None, 256)           22016
```

```
 lstm_1 (LSTM)               (32, None, 1024)          5246976

 dense_1 (Dense)             (32, None, 86)            88150

=================================================================
Total params: 5357142 (20.44 MB)
Trainable params: 5357142 (20.44 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

For each character the model looks up the embedding, runs the LSTM one timestep with the embedding as input, and applies the dense layer to generate logits predicting the log-likelihood of the next character

[168]:
```
# !pip install pydot
# !pip install graphviz

tf.keras.utils.plot_model(
    model_1,
    show_shapes=True,
    show_layer_names=True,
    to_file='model_1.png'
)
```

```
You must install pydot (`pip install pydot`) and install graphviz (see
instructions at https://graphviz.gitlab.io/download/) for plot_model to work.
```

[170]:
```
for input_example_batch, target_example_batch in dataset_train.take(1):
    example_batch_predictions = model_1(input_example_batch)
    print("(batch_size, sequence_length, vocab_size) = ",␣
 ↪example_batch_predictions.shape)
```

```
(batch_size, sequence_length, vocab_size) =  (32, 500, 86)
```

[171]:
```
print('Prediction for the 1st letter of batch 1st sequence:')
print(example_batch_predictions[0, 0])
```

```
Prediction for the 1st letter of the batch 1st sequense:
tf.Tensor(
[-0.00140358  0.00259074  0.00891904 -0.01520341  0.01623035 -0.00243362
 -0.01448181  0.0106629   0.00697443  0.01969678  0.01065725  0.00618893
  0.00176394 -0.00709569 -0.0206212   0.00508396 -0.01972833 -0.02285493
  0.00100469  0.00383437  0.01187887 -0.01494261  0.03559244  0.00801092
 -0.01532706 -0.01000398  0.00880224  0.01399882  0.00863353  0.01410829
  0.00994116  0.00568161 -0.0256719   0.01485433  0.00734585 -0.00839514
  0.001781    0.0012694  -0.00048264  0.00371801 -0.02856719  0.01499597
  0.00194854 -0.00566038  0.00686628  0.01064852 -0.0043367  -0.00539365
 -0.01363399  0.00015457 -0.01453467 -0.00054858 -0.00506167  0.00383148
  0.00516334  0.01610614 -0.02501759 -0.01092247 -0.01415366  0.00451942
```

```
 0.0052682  -0.01194449 -0.01596361 -0.00163683  0.01104035  0.00059822
-0.01685531  0.00906828 -0.01151857 -0.00045008 -0.00466816  0.01540542
 0.02122791 -0.00463165  0.00974108  0.01287864 -0.00759264 -0.00562551
 0.00643799 -0.00075074  0.01258999 -0.00155468 -0.02149971 -0.01060661
 0.00365201 -0.00917164], shape=(86,), dtype=float32)
```

0.7 TRAINING

[175]:
```
# An objective function with the signature scalar_loss = fn(y_true, y_pred).
def loss(labels, logits):
    entropy = tf.keras.losses.sparse_categorical_crossentropy(
        y_true=labels,
        y_pred=logits,
        from_logits=True
    )

    return entropy


example_batch_loss = loss(target_example_batch, example_batch_predictions)

print("Prediction shape: ", example_batch_predictions.shape, " # (batch_size,␣
 ↪sequence_length, vocab_size)")
print("Loss shape:       ", example_batch_loss.shape)
print("scalar_loss:      ", example_batch_loss.numpy().mean())
```

```
Prediction shape:  (32, 500, 86)  # (batch_size, sequence_length, vocab_size)
Loss shape:        (32, 500)
scalar_loss:       4.4540215
```

[177]:
```
# adam_optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)  # v2.11+␣
 ↪optimizer runs slowly on M1/M2 Macs
adam_optimizer = tf.keras.optimizers.legacy.Adam(learning_rate=0.001)

model_1.compile(
    optimizer=adam_optimizer,
    loss=loss
)
```

[180]:
```
early_stopping_callback = tf.keras.callbacks.EarlyStopping(
    patience=5,
    monitor='loss',
    restore_best_weights=True,
    verbose=1
)
```

[ ]:
```
CHECKPOINT_DIR = "./checkpoints"
os.makedirs(CHECKPOINT_DIR, exist_ok=True)
```

[201]:
```
checkpoint_prefix = os.path.join(CHECKPOINT_DIR, 'ckpt_rnn_{epoch}')
checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_prefix,
    save_weights_only=True
)
```

[205]:
```
INITIAL_EPOCH = 0
EPOCHS_DELTA = 10
EPOCHS = INITIAL_EPOCH + EPOCHS_DELTA
STEPS_PER_EPOCH = 200

# print('\n')
# print('INITIAL_EPOCH:    ', INITIAL_EPOCH)
# print('EPOCHS_DELTA:     ', EPOCHS_DELTA)
# print('EPOCHS:           ', EPOCHS)
# print('STEPS_PER_EPOCH:  ', STEPS_PER_EPOCH)
```

[206]:
```
history_1 = {}
```

[207]:
```
history_1[INITIAL_EPOCH] = model_1.fit(
    x=dataset_train,
    epochs=EPOCHS,
    steps_per_epoch=STEPS_PER_EPOCH,
    initial_epoch=INITIAL_EPOCH,
    callbacks=[checkpoint_callback, early_stopping_callback]
)

model_name = 'p2r_rnn_raw_' + str(INITIAL_EPOCH) + '.h5'
model_1.save(model_name, save_format='h5')
```

```
Epoch 1/10
200/200 [==============================] - 686s 3s/step - loss: 0.0140
Epoch 2/10
200/200 [==============================] - 676s 3s/step - loss: 0.0344
Epoch 3/10
200/200 [==============================] - 699s 3s/step - loss: 0.4600
Epoch 4/10
200/200 [==============================] - 692s 3s/step - loss: 0.0233
Epoch 5/10
200/200 [==============================] - 706s 4s/step - loss: 0.0151
Epoch 6/10
200/200 [==============================] - ETA: 0s - loss: 0.1032Restoring model
weights from the end of the best epoch: 1.
200/200 [==============================] - 671s 3s/step - loss: 0.1032
Epoch 6: early stopping

/Applications/anaconda3/envs/p2r/lib/python3.8/site-
packages/keras/src/engine/training.py:3000: UserWarning: You are saving your
```

```
model as an HDF5 file via `model.save()`. This file format is considered legacy.
We recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')`.
  saving_api.save_model(
```

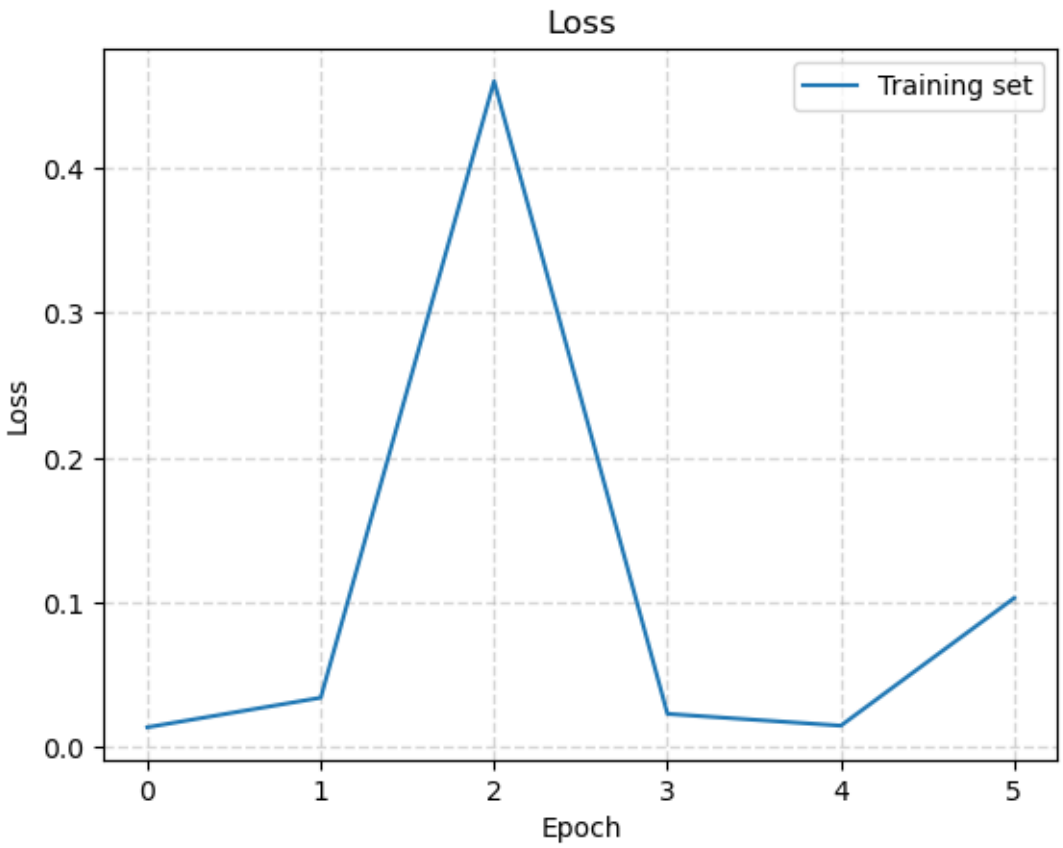
[ ]:
```
# rnn_model_name = f'./models/p2r_lstm_{INITIAL_EPOCH}.h5'
# model_1.save(rnn_model_name, save_format='h5')
```

[ ]:
```

```

0.8 VISUALIZATION

[208]:
```
def render_training_history(training_history):
    if 'history' in training_history:
        loss = training_history.history['loss']
    else:
        loss = []
        for initial_epoch in training_history:
            loss += training_history[initial_epoch].history['loss']

    plt.title('Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.plot(loss, label='Training set')
    plt.legend()
    plt.grid(linestyle='--', linewidth=1, alpha=0.5)
    plt.show()
```

[209]:
```
render_training_history(history_1)
```



0.9 GENERATING A RECIPE

[210]:
```
simplified_batch_size = 1

model_1_simplified = build_model_1(vocab_size, embedding_dim, rnn_units,␣
 ↪simplified_batch_size)
model_1_simplified.load_weights(tf.train.latest_checkpoint(CHECKPOINT_DIR))
model_1_simplified.build(tf.TensorShape([simplified_batch_size, None]))
```

[211]:
```
model_1_simplified.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_2 (Embedding)      (1, None, 256)            22016

 lstm_2 (LSTM)                (1, None, 1024)           5246976

 dense_2 (Dense)              (1, None, 86)             88150
```

```
=================================================================
Total params: 5357142 (20.44 MB)
Trainable params: 5357142 (20.44 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[212]:
```
model_1_simplified.input_shape
```

[212]:
```
(1, None)
```

[213]:
```
model_name = 'p2r_rnn.h5'
model_1_simplified.save(model_name, save_format='h5')
```

```
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet
to be built. `model.compile_metrics` will be empty until you train or evaluate
the model.

/Applications/anaconda3/envs/p2r/lib/python3.8/site-
packages/keras/src/engine/training.py:3000: UserWarning: You are saving your
model as an HDF5 file via `model.save()`. This file format is considered legacy.
We recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')`.
  saving_api.save_model(
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet
to be built. `model.compile_metrics` will be empty until you train or evaluate
the model.
```

1 ———

[214]:
```
def generate_text(model, start_string, num_generate=1000, temperature=1.0):
    """
    Parameters:
        - num_generate: number of characters to generate.
        - temperature:
            - Low temperatures results in more predictable text.
            - Higher temperatures results in more surprising text.
    """
    # Evaluation step
    padded_start_string = MARKER_TITLE + start_string

    # Vectorizing
    input_indices = np.array(tokenizer.
 ↪texts_to_sequences([padded_start_string]))

    # Empty string to store our results.
    text_generated = []
```

```
    # Here batch size == 1.
    model.reset_states()
    for char_index in range(num_generate):
        predictions = model(input_indices)
        predictions = tf.squeeze(predictions, 0)  # remove the batch dimension

        # Using a categorical distribution to predict the character returned by␣
↪the model.
        predictions = predictions / temperature
        predicted_id = tf.random.categorical(
            predictions,
            num_samples=1
        )[-1, 0].numpy()

        # Pass predicted character as the next input to the model
        # along with the previous hidden state.
        input_indices = tf.expand_dims([predicted_id], 0)
        next_character = tokenizer.sequences_to_texts(input_indices.numpy())[0]
        text_generated.append(next_character)

    return (padded_start_string + ''.join(text_generated))
```

[215]:
```
def generate_combinations(model, input_list):
    recipe_length = 1000
    try_temperature = [1.0, 0.8, 0.4, 0.2]

    for entry in input_list:
        for temperature in try_temperature:
            generated_text = generate_text(
                model,
                start_string=entry,
                num_generate=recipe_length,
                temperature=temperature
            )

            print('---------')
            print(f'Attempt: "{entry}" + {temperature}')
            print('---------')
            print(generated_text)
            print('\n\n')
```

[217]:
```
input_ingredients1 = ['avocado', 'rice', 'shrimp', 'tomato']
generate_combinations(model_1_simplified, input_ingredients1)
```

```
---------
Attempt: "avocado" + 1.0
---------
<t>avocado Calaman Bars
```

<i>Ingredients:
- 3 lb. ground chuck
- 1 1/2 lb. lean eggs, floured
- 3 tsp. salt
- 1 2 gre. bell sed oring to pea stress
- 1 panche salsh in sealt, stirling
- 3 eggs
- 1 cup grated cheese

<d>Directions: Preheat oven to 350° der ispredients in smupt butter. Mis whill grush 1/2 Gup wate 2i>I lith and let to flo rmeased. Bak_____
_____
_____
_____
_____
_____
_____
_____
_____

Attempt: "avocado" + 0.8

<t>avocado Sausage Cootoot Cake
- 1 pep anilla to das
- 1 teaspoon dried Osmanthuc salted pot oit
- 1 grees onionsthinchs linges
- 1 stall spread cheese
- 2 lb. cottage cheese
- 1/2 cup hanted baking powder
- 1/4 teaspoon salt

<d>Directions: Preheat the oven to 400 degrees. Cut the potatoes lengthwise into 1/4 inch thick slices, then cut the slices lengthwise into 1/4 inch wide sticks. Place the potatoes in a large bowl, lightly spray with nontsing and alt together. Add to dris mixture. Cook peas with first. Fol_____
_____
_____
_____
_____
_____
_____

Attempt: "avocado" + 0.4

<t>avocado-Hashed and Cool Whip

<i>Ingredients:
- 1 c. sugar
- 1/2 c. white syrup
- 1 can (12 oz.) peanuts
- 1 tsp. baking soda

<d>Directions: Cover the bottom of 9 x 13-inch pan with rice. Sprinkle salt over rice. Cover rice with broccoli. Lay cleaned chicken over top. Mix soup and water. Pour over other layers. Cover with foil. Bake at 350° for 45 minutes. Check to see if meat falls off bones._____

Attempt: "avocado" + 0.2

<t>avocado-Hasher Cake With seeded Pace

<i>Ingredients:
- 1 cup flour
- 1 cup (packed) brown sugar
- 1/2 cup unsweetened cocoa powder
- 1 Tbsp. instant coffee
- 1 1/2 tsp. baking powder
- 1/2 tsp. salt
- 1/2 cup butterscotch chips
- 1/2 cup milk
- 2 Tbsp. vegetable oil
- 1 tsp. vanilla extract
- 1 cup hot water
- whipped cream (optional)

<d>Directions: Preheat oven to 350. Toss garlic with olive oil and place in oven dish, cover and roast for 15 minutes. Let cool and chop. Mix shrimp, goat cheese, egg _____

Attempt: "rice" + 1.0

<t>ricep: Olive And Tomato Care

<i>Ingredients:
- 2 c. flour
- 1 1/2 tsp. baking soda
- 1/2 tsp. cloves
- 1/2 tsp. cinnamon
- 1 c. walnuts
- 1 c. brown sugar
- 1 c. butter
- 1 st. ange zucharo margeh tho oul pie dissolice squars, lemon epan, skineded, cut in 1/2-inch pieces
- 1 (8 ounce) package sugar soak panad

<d>Directions: Cook butter and mix until blends. Spray with Kitchen Bouquett mirrilie press in pinkt in the fridge cofte. varing a deply ground blef ap laghtle soda on to 375 ° minut._____

Attempt: "rice" + 0.8

<t>ricepie: Casirty Vegetha Salad

<i>Ingredients:
- 12 ounces Chilean Sea Bass (preferably 2 six-ounce fillets)
- 1 tablespoon Minced garlic (roughly 3 cloves)
- 1 tablespoon Fresh rosemary, minced
- 1 tablespoon Chile powder
- 1/4 teaspoon salt
- 1/4 teaspoon white pepper
- olive oil flavored cooking spray

<d>Directions: Preheat oven to 325 ° F (165 ° C). Sift together twice: flour,

baking powder, and salt and nutmeg at 375° for 10 to 125 minutes. Serve wather at yourf. Bake 15 minutes of thick bol-. Add cherry put ponks. Cook pastare over 45 minutes. Cool serve and the perning _____

Attempt: "rice" + 0.4

<t>ricepe: South Of The Border Dip

<i>Ingredients:
- 1/2 cups Slivered Almonds
- 1/2 cups Unsweetened Shredded Coconut Plus Extra For Decorating
- 4 ounces, weight Melting Chocolate
- 36 Mini Eggs

<d>Directions: Place almonds in a skillet and toast over medium-high heat for 3-4 minutes. Add shredded coconut and toast with almonds until lightly golden brown and fragrant. Scoop out about 1 tablespoon toasted coconut for decorating. Set aside. Break up a chocolate bar (or use melting _____

Attempt: "rice" + 0.2

<t>rice: Corn Bread

<i>Ingredients:
- 1/2 slices butter
- 1/4 c. sugar
- 1 can walnuts, crasserfled paw (I four over of large)
- 1/4 small reasons, chopped
- 1/4 tsp vanilla in argerdients
- 1 tsp. vanilla

<d>Directions: Grease an 8-inch square dish. Boil carrots in salted water,

uncovered; simmer for 20 minutes or until tender. Drain carrots. Put oleo, eggs, sugar, flour, baking powder and vanilla in blender. Add carrots, a few at a time, and puree. Pour into prepared baking dish. Bake at _____

Attempt: "shrimp" + 1.0

<t>shrimpe: Sour Cream

<i>Ingredients:
- 2 c. vanilla wafer crumbs
- 1 small can frozen or orange juice
- 1 c. lemon juice
- 1/2 gg. water
- 1 c. sugar
- 4 eggs
- 1 c. stropp ngeler, ceeder and chopped (1/2 sp.)
- 1 1/2 c. sugar
- 2 c. sliced romecolie, chopped
- 1 cup zesher frozen in to fligr(sters
- 1/2 lb slaged brease (rbyellors
- 1/4 lb can chopped broccoli
- last and grated
- 2 1/2 c. fell- water to Jall of leam selies)
- 2 tabspesporks cries
- 2astacho pepper

<d>Directions: Mix together brown broccoli, thar usca netr, grated pinta cheese. Brend sits carbery ta s ake nuttereatel trembersillor oil in dheppers froz. Pur in sala onion, stir conastens in sugar until slozes the the small duffin mox for pork, shreld bakes wheet cream cheese and hone. Rello greased. Gan in a serving pett._____

Attempt: "shrimp" + 0.8

<t>shrimpe: Six Cup Fruit Salad

<i>Ingredients:
- 1 cup mandarin orange, drained
- 1 cup pineapple chunk, drained
- 1 cup angel flake coconut
- 1 cup sour cream
- 1 cup chopped nuts
- 1 cup kinlain and milk
- 1/3 to butcep, cornstarch
- 1/2 c. meat for eshiving
- 1/3 cup cider vinegar
- 1/3 cup honey
- 1/3 cup olive oil
- 2 teaspoons salt
- 1/2 teaspoon white pepper

<d>Directions: Roll out pie dough on a lightly floured surface and knead 7 to 8 minutes. Grash 8 or notat. meat. Add chicken on tap 1/2-incho_____

Attempt: "shrimp" + 0.4

<t>shrimpe: Strawberry "Ice Cream"

<i>Ingredients:
- 2 frozen bananas
- 1 c. frozen strawberries
- 1 tsp. vanilla ix cranberry 1/2 c. frozen grape (or 1/2 mead manuan
- 1 car cream of chicken soup
- 1 c. grated Longhorn cheese
- 1/2 stick oleo
- 4 slices toasted bread, crumbled

<d>Directions: Cook chicken and take off of bones. Chop. Cook broccoli, chop up and drain. Butter 9 x 12-inch casserole dish. Put broccoli in dish, cover with the chopped chicken. Mix sour cream, soup and 3/4 cup cheese._____

Attempt: "shrimp" + 0.2

<t>shrimpe: Sour Cream-Lemon Pie

<i>Ingredients:
- 1 1/2 cups all-purpose flour
- 1 pounds boneless, skinless chicken thighs
- 1 1/4 teaspoons salt, divided
- 1/2 teaspoon freshly ground black pepper
- 1 tablespoon olive oil, divided
- 2 tablespoons barbeque seasoning, divided, or to taste
- 2 pounds bulk Italian sausage
- 1 cup finely shredded Cheddar and Monterey Jack cheese blend
- 1/4 cup diced fresh jalapeno pepper (optional)
- 2 green onions, thinly sliced
- 4 cloves garlic, minced
- 1 (12 ounce) bottle barbeque sauce, divided
- wood chips, soaked

<d>Directions: Stack and weave 1 1/2 pounds bacon slices into a 12-inch square lattice. Spow in strips and bowl. Add eggs, for and pumin. Add masharade onion, har in the bread flour and mix with cake pot. Sprinkle chips on top of caramel. Allow ch_____

Attempt: "tomato" + 1.0

<t>tomatoes: Presheror Jello

<i>Ingredients:
- 1 1/2 ounces tenders, diced
- 1/3 tsp. honey
- 1/2 c. colf mushrooms, drained
- 6 or. can tomito (oy or eggs, stirr whepper fresh more wite frize)
- 1 can tonings (lange steace
- 1 1/2 c. sliced fresh carrots
- 1 c. may water
- 3 c. sugar
- 2 Tbsp. sugar
- 1 c. sugar

- 6 lng. jar seasoning inso sceles in stear for hat (Drineap)
- Dres. garonc slarse
- 2 Tbsp. raisina drain
- 1 1/2 tsp. salt
- 1 eggs, well beaten
- 1 can chocolate syrup
- 1 c. flourz, trained
- 8 oz. lestro gill seap, cooked and drained
- 1 (14 ounce) jar of hot pickled banana pepper rings, with juice

<d>Directions: S. Dre sigerry baking sodaas on a medium spa. Cho bowle stir 1 boun. Boand 15 minutes on Hinghed._____
_____
_____
_____

Attempt: "tomato" + 0.8

<t>tomatoes: Broccoli Casserole

<i>Ingredients:
- 1 whole egg, alvoradel
- 1/2 tsp. salt
- 1/8 tsp. pepper
- 1 unbaked (9-inc) thisk) to rsinch then stals
- 6 oz. can plase cheese
- 2 1/2 c. flour
- 3 c. slaw flour
- 1/2 c. whipping cream, whipped
- 1 1/2 tsp. vanilla

<d>Directions: Grease ave in 1/2-inch thick. lasce ross the seved ad with temel-to te meature, ugrain salt over cream cheese of the dripped tomato all sour souce mixture. Cool bot water together torater harrace cheered over top of cream of heathed carrots and eggs, lemon juice, until urely medter. Add mix and cook for 2 minutes. Add beat. Lemon juices. Pour over fruit and mix._____
_____
_____
_____
_____
_____

Attempt: "tomato" + 0.4

```
<t>tomatoesalions and Hot Brain Cake

<i>Ingredients:
- 1/2 c. oleo, melted
- 1 c. powdered sugar
- 1 angel food cake
- 12 oz. carton Cool Whip
- 1 can cherry pie filling

<d>Directions: Break angel food cake into small pieces in square dish with lid.
Add Cool Whip and mix with cake. Mix cream cheese and powdered sugar together
and add to cake mixture. Add cherry pie filling on top. Chill._________________
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-------------------------------------


---------
Attempt: "tomato" + 0.2
---------
<t>tomatoesalions and mold in a diffingg pantage (4 ounchs dier coles, minced
seads cheese

<d>Directions: Cook fiod in a medium-size bowl, mix until bett and add this 1/8
tup oleo, stick salt pork ins a shellet with cream cheese mixture. Place
remainder of pie filling on top._____________________________________________
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-------------------------------------------
```

[216]:
```
input_ingredients2 = ['Mushroom', 'Apple', 'Slow', 'Banana', 'Homemade']
generate_combinations(model_1_simplified, input_ingredients2)
```

```
---------
Attempt: "Mushroom" + 1.0
---------
<t>Mushroomet And spinach Salad
```

<i>Ingredients:
- 1/4 pound chopped flour
- 1 1/2 teaspoons baking powder
- 1/2 teaspoon salt
- darted: sout eggut
- 1 1/2 cups pervet any olake juice (16 minutes)
<t>Directions: Mix all ingredients together in oil in the fir6 laters. Add pasta to diping dish. To nuts your saucepan. Brist with sausage this by brotn side asmext 1 to 10 minutes. Frost bott. Place browning spreat and ole on sterd iet m ix._____

Attempt: "Mushroom" + 0.8

<t>Mushroominallaw Salade For Combies

<i>Ingredients:
- 1 small onion, chopped
- 1 lb. mild sausage
- 1 lb. Velveeta mild Mexican
- 1 lb. Velveeta ice crumble (o live)
- 1 (8 oz.) can musaraded crourd for 3 larges (pour salt) in straines)
- 1 can mannash peer sald drained
- 1 (8 oz.) pkg. cornstarch
- 1/2 c. sugar
- 2 Tbsp. flour
- 1 tsp. baking powder
- 1 tsp. salt
- 2 eggs
- 1 c. sugar (less, if desired
- 3 Tbsp. flour
- 1 tsp. baking powder
- 1 tsp. salt
- 2 eggs
- 1 c. spring water
- 3 c. cook 4 1/2 s. c. frozen carn
- 1 c. milk
- 1 small can Mandarin oranges, drained
- 4 bonins cubars

- 1 stalk ceperry dinger
- 2 c. sugar
- 2 Tbsp. flour
- 1 tsp. salt
- 1 tsp. vanilla
- 1 c. milk (evaporated)

<d>Directions: Cream oleo, shortening and sugar; add salt, veated can and onion. Heat stirs and edjuis. Die for baiting warger to a boil. Sit mild. Brend in together and add to cake mixture. Add cherry pie filling on top. Chill._____

Attempt: "Mushroom" + 0.4

<t>Mushroom Plaute Fruit Spookid Cake

<i>Ingredients:
- 1 c. butter, slightly leame
- 1 c. picken sweess on margarie, creamed
- 1 1/2 cups beef stock
- 1 jar (12 oz) marinated artichoke hearts, drained and halved
- 1/2 cup jarred roasted red bell pepper, drained and thinly sliced
- 1/4 cup preserved lemon peel, thinly sliced
- 1/4 cup fresh flat-leaf parsley, coarsely ch_____

Attempt: "Mushroom" + 0.2

<t>Mushroom Plades And Bored Cake

<i>Ingredients:
- 1 c. butter, softened
- 2 1/2 c. sugar
- 6 eggs
- 3 c. plain flour
- 1/2 c. whipping cream, whipped
- 1 1/2 tsp. vanilla

<d>Directions: Beat butter and sugar until light and fluffy. Add eggs, one at a time, beating 1 minute after each. Whip cream until stiff peaks form. Add flour and cream alternately, just until combined after each. Beat in vanilla. Pour batter in a greased 10-inch tube pan. Bake at 300° for 1 hour and 15 minutes. Cool for 10 m_____
_____
_____
_____
_____
_____
_____

Attempt: "Apple" + 1.0

<t>Appleions And thimberry And Cake Mix 1 box and powder in a sara no blender pap water and lish with tookes, conkes, and flave butter, ats to firting eggry mix mexture on serves 40._____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Attempt: "Apple" + 0.8

<t>Apple: Pound Cake

<i>Ingredients:
- 1 peavs peas of mushroom soup or cream of chicken
- 2 cans water
- uncooked rice
- 2 boxes chopped broccoli
- salt

<d>Directions: Cream together butter, cream cheese and hondey white pasta and out oil ha dis boil. Add cherry put pon soft beats. Add eggs and beat well. Add chop coase flour hanilla chil in bottom of the Chill. Cover with sode side _____
_____

Attempt: "Apple" + 0.4

<t>Appled Poun Ched

<i>Ingredients:
- 1 cup flour
- 1 cup (packed) brown sugar
- 1/2 cup unsweetened cocoa powder
- 1 Tbsp. instant coffee
- 1 1/2 tsp. baking powder
- 1/2 tsp. salt
- 1/2 cup butterscotch chips
- 1/2 cup milk
- 2 Tbsp. vegetable oil
- 1 tsp. vanilla extract
- 1 cup hot water
- whipped cream (optional)

<d>Directions: Preheat oven to 350. Toss garlic with olive oil and place in oven dish, cover and roast for 15 minutes. Let cool and chop. Mix shrimp, goat cheese, egg ---
---

Attempt: "Apple" + 0.2

<t>Apple: Push Cossagute

<i>Ingredients:
- 2 lbs frozen chopped broccoli
- 1 (8 oz. cart) flour
- 1/4 c. white brown sugar

- 1 1/2 cups bacon thop
- 1 sall tomy to fals
- 2 slices thick- bacon asp rass ast acoupen And mararge
- 1/4 cup olive oil
- 1/4 cup chopped green olives
- 1/4 cup Chopped sun dried tomatoes
- 1 tablespoon Capers
- 2 tablespoons Olive oil

<d>Directions: Add all ingredients to a food processor and pulse until you are left with a corse paste. Should be stored in a jar and kept in the fridge. This can be served right away but I recommend waiting a day or too for the flavors to really blend._____
_____
_____
_____
_____
____

Attempt: "Slow" + 1.0

<t>Slowece: Lasimar Sauce(Por And Pie Cass (1/2 Tbsp.)
- 1 1/4 c. water
- 3 eggs

<d>Directions: Mix all ingredients together and chill for an hour. Adjust salt and pepper to taste just before serving._____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Attempt: "Slow" + 0.8

<t>Sloweece: Pressichoon S/4 1/2 cups (10-5 finute) cans layegh and 1/2 cup sugar
- 2 tablespoons oil
- 2 eggs, servis, cornstarch

- 1/4 cup frozen cranberries, carrot-farsley cubes
- 1 teaspoon crushed red pepper flakes (optional)
- 4 tablespoons flour
- 3 1/2 cups chicken broth
- 3/4 cup wat_____

Attempt: "Slow" + 0.4

<t>Slowederins: Presher Topest

<i>Ingredients:
- 1 cup pan water
- 1 1/2 c. sugar
- 1 1/2 tsp. allord sugar
- 1 c. flour
- 1 tsp. baking soda
- 1 tsp. lay (lay feat for 1 car)
- 2 medium onions, chopped
- 1 c. flour
- 3 tsp. baking soda
- 2 tsp. baking powder
- 1 tsp. salt
- 2 eggs
- 1 c. strong black coffee or 2 tsp. instant coffee plus 1 c. boiling water
- 1 c. buttermilk
- 1/2 c. oil
- 1 tsp. vanilla

<d>Directions: Cream margarine and sugar together. Add eggs and beat well. Add chocolate syrup. Gradually add flour and salt and beat. Add vanilla. Bake in a 9 x 13-inch pan that has been greased and floured. Bake for 30 minutes at 350°. Let cool._____

Attempt: "Slow" + 0.2

<t>Slowered And sauce (about 1/2 heas)
- 1 can pineapple juice
- 1 can orange juice
- 1 c. lemon juice
- 1 (1/2 gal.) carton orange sherbet
- 1 qt. ginger ale, chilled

<d>Directions: Dissolve gelatin in boiling water. Add sugar and stir until dissolved. Add cold water. Combine juices and add to Jell-O mixture. Chill. Spoon sherbet into punch and add ginger ale just before serving. Serves 40._____

Attempt: "Banana" + 1.0

<t>Bananay Bulls

<i>Ingredients:
- 1 lb. ground cheese
- 1/4 c. chopped onion
- 1/2 tsp. dried basil
- 1/2 tsp. salt
- 1/8 tsp. corns
- 1/2 c. shret enga
- 2 c. masim water
- 1 c. Bicon tomstish chunks
- 1 teaspoon Whiped soy cheese

<d>Directions: Grill checreat the potatoes. Mincomaw vegetable or 1 cuples, until seals. Add meltained brown pour and sore dering. Stir botsomi laye che che ns._____

Attempt: "Banana" + 0.8

<t>Bananas: Prush Tomato pous on walnuspon bake and stir until dissolved. Add fruit and put in oiled mold._____

Attempt: "Banana" + 0.4

<t>Bananas: Ita Flish Allot pears (fore for 25 minutes until golden. Add onions and salt pork to saucepan. Add potatoes and water to cover. Cook until potatoes are tender. Add evaporated milk. Heat to serv_____

Attempt: "Banana" + 0.2

<t>Bananas: Inta File Foll in a medium bowl at mish)
- 4 sprays canrots
- 1 cup sugar
- 2 large eggs
- 1 cup buttermilk

<d>Directions: Grease a 9-by-13-inch baking pan. Whisk together flour, baking powder, baking soda, salt and 1 tsp. cinnamon in a me_____

Attempt: "Homemade" + 1.0

<t>Homemade BunAd Bread With Corezey

<i>Ingredients:
- 1 pug. (RoT Bowere)

<D>Directions: Combine all firling ingredients, steep for 2-mince serve. Mix 1/2 cup flour and mix until nit pan. Cook and flour and sugar to heat ly ground. But an cook form the darill. Stir in ccrambersely slice. Bake in a 95 x 13-inch baking dish. Set aside 1 zour caco and nuts. Mix well. Rus. Pour into 2 greased poat. Wh_____

Attempt: "Homemade" + 0.8

<t>Homemade: Chocolate Chipse

<i>Ingredients:
- 1 large chopped froun
- 1/2 c. white verneap
- 1/8 c. mind
- 2 Tbsp. cornsalt
- 1/2 tsp. salt

- 1 tsp. vanilla

<d>Directions: Q. Dressing: combine dressing ingredients and whisk until
blended. Serves 6 to 8._____

Attempt: "Homemade" + 0.4

<t>Homemadelanusa Fruit Rometo

<i>Ingredients:
- 1 cup milk and vanilla
- 1/2 cup KRAFT Light Zesty Italian Dressing
- 1 can (12 oz.) or 2 cans (6 oz. each) white tuna in water, drained
- 1 cup KRAFT 2% Milk Shredded Sharp Cheddar Cheese, divided

<d>Directions: oven to 375°F. vegetables in colander in sink. Cook Macaroni as d
ire_____

Attempt: "Homemade" + 0.2

<t>Homemade: Bake Band

<i>Ingredients:
- 1 c. butter, softened
- 2 1/2 c. sugar
- 6 eggs
- 3 c. plain flour

```
- 1/2 c. whipping cream, whipped
- 1 1/2 tsp. vanilla

<d>Directions: Beat butter and sugar until light and fluffy. Add eggs, one at a
time, beating 1 minute after each. Whip cream until stiff peaks form. Add flour
and cream alternately, just until combined after each. Beat in vanilla. Pour
batter in a greased 10-inch tube pan. Bake at 300° for 1 hour and 15 minutes.
Cool for 10 m_____
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------
```

[ ]:
```
input_ingredients1 = ['avocado', 'rice', 'shrimp', 'tomato']
generate_combinations(model_1_simplified, input_ingredients1)
```

1.1 Save

[223]:
```
# !pip install -U notebook-as-pdf
# !conda install -c conda-forge pandoc

nb_name = f"./nlp/P2R_T5.ipynb"  # notebook name
!jupyter-nbconvert --to pdf ./nlp/P2R_T5.ipynb
!jupyter-nbconvert --to html ./nlp/P2R_T5.ipynb
```

```
[NbConvertApp] WARNING | Config option `kernel_spec_manager_class` not
recognized by `NbConvertApp`.
[NbConvertApp] WARNING | pattern './nlp/P2R_T5.ipynb' matched no files
This application is used to convert notebook files (*.ipynb)
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options,
as listed in the "Equivalent to" description-line of the aliases.
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
    Show the application's configuration (human-readable format)
    Equivalent to: [--Application.show_config=True]
```

```
--show-config-json
    Show the application's configuration (json format)
    Equivalent to: [--Application.show_config_json=True]
--generate-config
    generate default config file
    Equivalent to: [--JupyterApp.generate_config=True]
-y
    Answer yes to any questions instead of prompting.
    Equivalent to: [--JupyterApp.answer_yes=True]
--execute
    Execute the notebook prior to export.
    Equivalent to: [--ExecutePreprocessor.enabled=True]
--allow-errors
    Continue notebook execution even if one of the cells throws an error and
include the error message in the cell output (the default behaviour is to abort
conversion). This flag is only relevant if '--execute' was specified, too.
    Equivalent to: [--ExecutePreprocessor.allow_errors=True]
--stdin
    read a single notebook file from stdin. Write the resulting notebook with
default basename 'notebook.*'
    Equivalent to: [--NbConvertApp.from_stdin=True]
--stdout
    Write notebook output to stdout instead of files.
    Equivalent to: [--NbConvertApp.writer_class=StdoutWriter]
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
            relevant when converting to notebook format)
    Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=]
--clear-output
    Clear output of current file and save in place,
            overwriting the existing notebook.
    Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=
--ClearOutputPreprocessor.enabled=True]
--no-prompt
    Exclude input and output prompts from converted document.
    Equivalent to: [--TemplateExporter.exclude_input_prompt=True
--TemplateExporter.exclude_output_prompt=True]
--no-input
    Exclude input cells and output prompts from converted document.
            This mode is ideal for generating code-free reports.
    Equivalent to: [--TemplateExporter.exclude_output_prompt=True
--TemplateExporter.exclude_input=True
--TemplateExporter.exclude_input_prompt=True]
--allow-chromium-download
    Whether to allow downloading chromium if no suitable version is found on the
system.
```

```
    Equivalent to: [--WebPDFExporter.allow_chromium_download=True]
--disable-chromium-sandbox
    Disable chromium security sandbox when converting to PDF..
    Equivalent to: [--WebPDFExporter.disable_sandbox=True]
--show-input
    Shows code input. This flag is only useful for dejavu users.
    Equivalent to: [--TemplateExporter.exclude_input=False]
--embed-images
    Embed the images as base64 dataurls in the output. This flag is only useful
for the HTML/WebPDF/Slides exports.
    Equivalent to: [--HTMLExporter.embed_images=True]
--sanitize-html
    Whether the HTML in Markdown cells and cell outputs should be sanitized..
    Equivalent to: [--HTMLExporter.sanitize_html=True]
--log-level=<Enum>
    Set the log level by value or name.
    Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR',
'CRITICAL']
    Default: 30
    Equivalent to: [--Application.log_level]
--config=<Unicode>
    Full path of a config file.
    Default: ''
    Equivalent to: [--JupyterApp.config_file]
--to=<Unicode>
    The export format to be used, either one of the built-in formats
            ['PDFviaHTML', 'asciidoc', 'custom', 'html', 'latex', 'markdown',
'notebook', 'pdf', 'pdfviahtml', 'python', 'qtpdf', 'qtpng', 'rst', 'script',
'slides', 'webpdf']
            or a dotted object name that represents the import path for an
            ``Exporter`` class
    Default: ''
    Equivalent to: [--NbConvertApp.export_format]
--template=<Unicode>
    Name of the template to use
    Default: ''
    Equivalent to: [--TemplateExporter.template_name]
--template-file=<Unicode>
    Name of the template file to use
    Default: None
    Equivalent to: [--TemplateExporter.template_file]
--theme=<Unicode>
    Template specific theme(e.g. the name of a JupyterLab CSS theme distributed
    as prebuilt extension for the lab template)
    Default: 'light'
    Equivalent to: [--HTMLExporter.theme]
--sanitize_html=<Bool>
    Whether the HTML in Markdown cells and cell outputs should be sanitized.This
```

```
    should be set to True by nbviewer or similar tools.
    Default: False
    Equivalent to: [--HTMLExporter.sanitize_html]
--writer=<DottedObjectName>
    Writer class used to write the
                                        results of the conversion
    Default: 'FilesWriter'
    Equivalent to: [--NbConvertApp.writer_class]
--post=<DottedOrNone>
    PostProcessor class used to write the
                                        results of the conversion
    Default: ''
    Equivalent to: [--NbConvertApp.postprocessor_class]
--output=<Unicode>
    Overwrite base name use for output files.
                Supports pattern replacements '{notebook_name}'.
    Default: '{notebook_name}'
    Equivalent to: [--NbConvertApp.output_base]
--output-dir=<Unicode>
    Directory to write output(s) to. Defaults
                                    to output to the directory of each notebook.
To recover
                                    previous default behaviour (outputting to the
current
                                    working directory) use . as the flag value.
    Default: ''
    Equivalent to: [--FilesWriter.build_directory]
--reveal-prefix=<Unicode>
    The URL prefix for reveal.js (version 3.x).
            This defaults to the reveal CDN, but can be any url pointing to a
copy
            of reveal.js.
            For speaker notes to work, this must be a relative path to a local
            copy of reveal.js: e.g., "reveal.js".
            If a relative path is given, it must be a subdirectory of the
            current directory (from which the server is run).
            See the usage documentation
            (https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-
html-slideshow)
            for more details.
    Default: ''
    Equivalent to: [--SlidesExporter.reveal_url_prefix]
--nbformat=<Enum>
    The nbformat version to write.
            Use this to downgrade notebooks.
    Choices: any of [1, 2, 3, 4]
    Default: 4
    Equivalent to: [--NotebookExporter.nbformat_version]
```

```
Examples
--------

    The simplest way to use nbconvert is

            > jupyter nbconvert mynotebook.ipynb --to html

            Options include ['PDFviaHTML', 'asciidoc', 'custom', 'html',
'latex', 'markdown', 'notebook', 'pdf', 'pdfviahtml', 'python', 'qtpdf',
'qtpng', 'rst', 'script', 'slides', 'webpdf'].

            > jupyter nbconvert --to latex mynotebook.ipynb

            Both HTML and LaTeX support multiple output templates. LaTeX
includes
            'base', 'article' and 'report'.  HTML includes 'basic', 'lab' and
            'classic'. You can specify the flavor of the format used.

            > jupyter nbconvert --to html --template lab mynotebook.ipynb

            You can also pipe the output to stdout, rather than a file

            > jupyter nbconvert mynotebook.ipynb --stdout

            PDF is generated via latex

            > jupyter nbconvert mynotebook.ipynb --to pdf

            You can get (and serve) a Reveal.js-powered slideshow

            > jupyter nbconvert myslides.ipynb --to slides --post serve

            Multiple notebooks can be given at the command line in a couple of
            different ways:

            > jupyter nbconvert notebook*.ipynb
            > jupyter nbconvert notebook1.ipynb notebook2.ipynb

            or you can specify the notebooks list in a config file, containing::

                c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

            > jupyter nbconvert --config mycfg.py

To see all available configurables, use `--help-all`.

[NbConvertApp] WARNING | Config option `kernel_spec_manager_class` not
```

```
recognized by `NbConvertApp`.
[NbConvertApp] WARNING | pattern './nlp/P2R_T5.ipynb' matched no files
This application is used to convert notebook files (*.ipynb)
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options,
as listed in the "Equivalent to" description-line of the aliases.
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
    Show the application's configuration (human-readable format)
    Equivalent to: [--Application.show_config=True]
--show-config-json
    Show the application's configuration (json format)
    Equivalent to: [--Application.show_config_json=True]
--generate-config
    generate default config file
    Equivalent to: [--JupyterApp.generate_config=True]
-y
    Answer yes to any questions instead of prompting.
    Equivalent to: [--JupyterApp.answer_yes=True]
--execute
    Execute the notebook prior to export.
    Equivalent to: [--ExecutePreprocessor.enabled=True]
--allow-errors
    Continue notebook execution even if one of the cells throws an error and
include the error message in the cell output (the default behaviour is to abort
conversion). This flag is only relevant if '--execute' was specified, too.
    Equivalent to: [--ExecutePreprocessor.allow_errors=True]
--stdin
    read a single notebook file from stdin. Write the resulting notebook with
default basename 'notebook.*'
    Equivalent to: [--NbConvertApp.from_stdin=True]
--stdout
    Write notebook output to stdout instead of files.
    Equivalent to: [--NbConvertApp.writer_class=StdoutWriter]
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
            relevant when converting to notebook format)
    Equivalent to: [--NbConvertApp.use_output_suffix=False
```

```
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=]
--clear-output
    Clear output of current file and save in place,
            overwriting the existing notebook.
    Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=
--ClearOutputPreprocessor.enabled=True]
--no-prompt
    Exclude input and output prompts from converted document.
    Equivalent to: [--TemplateExporter.exclude_input_prompt=True
--TemplateExporter.exclude_output_prompt=True]
--no-input
    Exclude input cells and output prompts from converted document.
            This mode is ideal for generating code-free reports.
    Equivalent to: [--TemplateExporter.exclude_output_prompt=True
--TemplateExporter.exclude_input=True
--TemplateExporter.exclude_input_prompt=True]
--allow-chromium-download
    Whether to allow downloading chromium if no suitable version is found on the
system.
    Equivalent to: [--WebPDFExporter.allow_chromium_download=True]
--disable-chromium-sandbox
    Disable chromium security sandbox when converting to PDF..
    Equivalent to: [--WebPDFExporter.disable_sandbox=True]
--show-input
    Shows code input. This flag is only useful for dejavu users.
    Equivalent to: [--TemplateExporter.exclude_input=False]
--embed-images
    Embed the images as base64 dataurls in the output. This flag is only useful
for the HTML/WebPDF/Slides exports.
    Equivalent to: [--HTMLExporter.embed_images=True]
--sanitize-html
    Whether the HTML in Markdown cells and cell outputs should be sanitized..
    Equivalent to: [--HTMLExporter.sanitize_html=True]
--log-level=<Enum>
    Set the log level by value or name.
    Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR',
'CRITICAL']
    Default: 30
    Equivalent to: [--Application.log_level]
--config=<Unicode>
    Full path of a config file.
    Default: ''
    Equivalent to: [--JupyterApp.config_file]
--to=<Unicode>
    The export format to be used, either one of the built-in formats
            ['PDFviaHTML', 'asciidoc', 'custom', 'html', 'latex', 'markdown',
'notebook', 'pdf', 'pdfviahtml', 'python', 'qtpdf', 'qtpng', 'rst', 'script',
```

```
'slides', 'webpdf']
            or a dotted object name that represents the import path for an
            ``Exporter`` class
    Default: ''
    Equivalent to: [--NbConvertApp.export_format]
--template=<Unicode>
    Name of the template to use
    Default: ''
    Equivalent to: [--TemplateExporter.template_name]
--template-file=<Unicode>
    Name of the template file to use
    Default: None
    Equivalent to: [--TemplateExporter.template_file]
--theme=<Unicode>
    Template specific theme(e.g. the name of a JupyterLab CSS theme distributed
    as prebuilt extension for the lab template)
    Default: 'light'
    Equivalent to: [--HTMLExporter.theme]
--sanitize_html=<Bool>
    Whether the HTML in Markdown cells and cell outputs should be sanitized.This
    should be set to True by nbviewer or similar tools.
    Default: False
    Equivalent to: [--HTMLExporter.sanitize_html]
--writer=<DottedObjectName>
    Writer class used to write the
                                        results of the conversion
    Default: 'FilesWriter'
    Equivalent to: [--NbConvertApp.writer_class]
--post=<DottedOrNone>
    PostProcessor class used to write the
                                        results of the conversion
    Default: ''
    Equivalent to: [--NbConvertApp.postprocessor_class]
--output=<Unicode>
    Overwrite base name use for output files.
                Supports pattern replacements '{notebook_name}'.
    Default: '{notebook_name}'
    Equivalent to: [--NbConvertApp.output_base]
--output-dir=<Unicode>
    Directory to write output(s) to. Defaults
                                    to output to the directory of each notebook.
To recover
                                    previous default behaviour (outputting to the
current
                                    working directory) use . as the flag value.
    Default: ''
    Equivalent to: [--FilesWriter.build_directory]
--reveal-prefix=<Unicode>
```

```
    The URL prefix for reveal.js (version 3.x).
            This defaults to the reveal CDN, but can be any url pointing to a
copy
            of reveal.js.
            For speaker notes to work, this must be a relative path to a local
            copy of reveal.js: e.g., "reveal.js".
            If a relative path is given, it must be a subdirectory of the
            current directory (from which the server is run).
            See the usage documentation
            (https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-
html-slideshow)
            for more details.
    Default: ''
    Equivalent to: [--SlidesExporter.reveal_url_prefix]
--nbformat=<Enum>
    The nbformat version to write.
            Use this to downgrade notebooks.
    Choices: any of [1, 2, 3, 4]
    Default: 4
    Equivalent to: [--NotebookExporter.nbformat_version]

Examples
--------

    The simplest way to use nbconvert is

            > jupyter nbconvert mynotebook.ipynb --to html

            Options include ['PDFviaHTML', 'asciidoc', 'custom', 'html',
'latex', 'markdown', 'notebook', 'pdf', 'pdfviahtml', 'python', 'qtpdf',
'qtpng', 'rst', 'script', 'slides', 'webpdf'].

            > jupyter nbconvert --to latex mynotebook.ipynb

            Both HTML and LaTeX support multiple output templates. LaTeX
includes
            'base', 'article' and 'report'.  HTML includes 'basic', 'lab' and
            'classic'. You can specify the flavor of the format used.

            > jupyter nbconvert --to html --template lab mynotebook.ipynb

            You can also pipe the output to stdout, rather than a file

            > jupyter nbconvert mynotebook.ipynb --stdout

            PDF is generated via latex

            > jupyter nbconvert mynotebook.ipynb --to pdf
```

```
            You can get (and serve) a Reveal.js-powered slideshow

            > jupyter nbconvert myslides.ipynb --to slides --post serve

            Multiple notebooks can be given at the command line in a couple of
            different ways:

            > jupyter nbconvert notebook*.ipynb
            > jupyter nbconvert notebook1.ipynb notebook2.ipynb

            or you can specify the notebooks list in a config file, containing::

                c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

            > jupyter nbconvert --config mycfg.py

To see all available configurables, use `--help-all`.
```

[222]:
```
# --TemplateExporter.exclude_input=True ./nlp/P2R_T5.ipynb
```

```
[NbConvertApp] WARNING | Config option `kernel_spec_manager_class` not
recognized by `NbConvertApp`.
[NbConvertApp] WARNING | pattern './nlp/P2R_T5.ipynb' matched no files
This application is used to convert notebook files (*.ipynb)
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options,
as listed in the "Equivalent to" description-line of the aliases.
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
    Show the application's configuration (human-readable format)
    Equivalent to: [--Application.show_config=True]
--show-config-json
    Show the application's configuration (json format)
    Equivalent to: [--Application.show_config_json=True]
--generate-config
    generate default config file
```

```
    Equivalent to: [--JupyterApp.generate_config=True]
-y
    Answer yes to any questions instead of prompting.
    Equivalent to: [--JupyterApp.answer_yes=True]
--execute
    Execute the notebook prior to export.
    Equivalent to: [--ExecutePreprocessor.enabled=True]
--allow-errors
    Continue notebook execution even if one of the cells throws an error and
include the error message in the cell output (the default behaviour is to abort
conversion). This flag is only relevant if '--execute' was specified, too.
    Equivalent to: [--ExecutePreprocessor.allow_errors=True]
--stdin
    read a single notebook file from stdin. Write the resulting notebook with
default basename 'notebook.*'
    Equivalent to: [--NbConvertApp.from_stdin=True]
--stdout
    Write notebook output to stdout instead of files.
    Equivalent to: [--NbConvertApp.writer_class=StdoutWriter]
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
            relevant when converting to notebook format)
    Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=]
--clear-output
    Clear output of current file and save in place,
            overwriting the existing notebook.
    Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=
--ClearOutputPreprocessor.enabled=True]
--no-prompt
    Exclude input and output prompts from converted document.
    Equivalent to: [--TemplateExporter.exclude_input_prompt=True
--TemplateExporter.exclude_output_prompt=True]
--no-input
    Exclude input cells and output prompts from converted document.
            This mode is ideal for generating code-free reports.
    Equivalent to: [--TemplateExporter.exclude_output_prompt=True
--TemplateExporter.exclude_input=True
--TemplateExporter.exclude_input_prompt=True]
--allow-chromium-download
    Whether to allow downloading chromium if no suitable version is found on the
system.
    Equivalent to: [--WebPDFExporter.allow_chromium_download=True]
--disable-chromium-sandbox
    Disable chromium security sandbox when converting to PDF..
    Equivalent to: [--WebPDFExporter.disable_sandbox=True]
--show-input
```

```
    Shows code input. This flag is only useful for dejavu users.
    Equivalent to: [--TemplateExporter.exclude_input=False]
--embed-images
    Embed the images as base64 dataurls in the output. This flag is only useful
for the HTML/WebPDF/Slides exports.
    Equivalent to: [--HTMLExporter.embed_images=True]
--sanitize-html
    Whether the HTML in Markdown cells and cell outputs should be sanitized..
    Equivalent to: [--HTMLExporter.sanitize_html=True]
--log-level=<Enum>
    Set the log level by value or name.
    Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR',
'CRITICAL']
    Default: 30
    Equivalent to: [--Application.log_level]
--config=<Unicode>
    Full path of a config file.
    Default: ''
    Equivalent to: [--JupyterApp.config_file]
--to=<Unicode>
    The export format to be used, either one of the built-in formats
            ['PDFviaHTML', 'asciidoc', 'custom', 'html', 'latex', 'markdown',
'notebook', 'pdf', 'pdfviahtml', 'python', 'qtpdf', 'qtpng', 'rst', 'script',
'slides', 'webpdf']
            or a dotted object name that represents the import path for an
            ``Exporter`` class
    Default: ''
    Equivalent to: [--NbConvertApp.export_format]
--template=<Unicode>
    Name of the template to use
    Default: ''
    Equivalent to: [--TemplateExporter.template_name]
--template-file=<Unicode>
    Name of the template file to use
    Default: None
    Equivalent to: [--TemplateExporter.template_file]
--theme=<Unicode>
    Template specific theme(e.g. the name of a JupyterLab CSS theme distributed
    as prebuilt extension for the lab template)
    Default: 'light'
    Equivalent to: [--HTMLExporter.theme]
--sanitize_html=<Bool>
    Whether the HTML in Markdown cells and cell outputs should be sanitized.This
    should be set to True by nbviewer or similar tools.
    Default: False
    Equivalent to: [--HTMLExporter.sanitize_html]
--writer=<DottedObjectName>
    Writer class used to write the
```

```
                                    results of the conversion
    Default: 'FilesWriter'
    Equivalent to: [--NbConvertApp.writer_class]
--post=<DottedOrNone>
    PostProcessor class used to write the
                                    results of the conversion
    Default: ''
    Equivalent to: [--NbConvertApp.postprocessor_class]
--output=<Unicode>
    Overwrite base name use for output files.
                Supports pattern replacements '{notebook_name}'.
    Default: '{notebook_name}'
    Equivalent to: [--NbConvertApp.output_base]
--output-dir=<Unicode>
    Directory to write output(s) to. Defaults
                                    to output to the directory of each notebook.
To recover
                                    previous default behaviour (outputting to the
current
                                    working directory) use . as the flag value.
    Default: ''
    Equivalent to: [--FilesWriter.build_directory]
--reveal-prefix=<Unicode>
    The URL prefix for reveal.js (version 3.x).
            This defaults to the reveal CDN, but can be any url pointing to a
copy
            of reveal.js.
            For speaker notes to work, this must be a relative path to a local
            copy of reveal.js: e.g., "reveal.js".
            If a relative path is given, it must be a subdirectory of the
            current directory (from which the server is run).
            See the usage documentation
            (https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-
html-slideshow)
            for more details.
    Default: ''
    Equivalent to: [--SlidesExporter.reveal_url_prefix]
--nbformat=<Enum>
    The nbformat version to write.
            Use this to downgrade notebooks.
    Choices: any of [1, 2, 3, 4]
    Default: 4
    Equivalent to: [--NotebookExporter.nbformat_version]

Examples
--------

    The simplest way to use nbconvert is
```

```
            > jupyter nbconvert mynotebook.ipynb --to html

            Options include ['PDFviaHTML', 'asciidoc', 'custom', 'html',
'latex', 'markdown', 'notebook', 'pdf', 'pdfviahtml', 'python', 'qtpdf',
'qtpng', 'rst', 'script', 'slides', 'webpdf'].

            > jupyter nbconvert --to latex mynotebook.ipynb

            Both HTML and LaTeX support multiple output templates. LaTeX
includes
            'base', 'article' and 'report'.  HTML includes 'basic', 'lab' and
            'classic'. You can specify the flavor of the format used.

            > jupyter nbconvert --to html --template lab mynotebook.ipynb

            You can also pipe the output to stdout, rather than a file

            > jupyter nbconvert mynotebook.ipynb --stdout

            PDF is generated via latex

            > jupyter nbconvert mynotebook.ipynb --to pdf

            You can get (and serve) a Reveal.js-powered slideshow

            > jupyter nbconvert myslides.ipynb --to slides --post serve

            Multiple notebooks can be given at the command line in a couple of
            different ways:

            > jupyter nbconvert notebook*.ipynb
            > jupyter nbconvert notebook1.ipynb notebook2.ipynb

            or you can specify the notebooks list in a config file, containing::

                c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

            > jupyter nbconvert --config mycfg.py

To see all available configurables, use `--help-all`.
```

[ ]: