# Recipe Gen

Data Collection & Preparation

1. Choose Datasets & Collect Data:
   * Food.com:
     + Store data as CSV in a structured format for easier processing.
     + <https://www.kaggle.com/datasets/shuyangli94/food-com-recipes-and-user-interactions/data>
     + Load the CSV file into a DataFrame
       - file\_path = r"\c441\RAW\_recipes.csv"
       - recipes\_df = pd.read\_csv(file\_path)
     + The dataset contains 231,637 rows and 12 columns.
2. Standardize Ingredients & Handle Values:
   * Aimed to create a dataset with 100,000 rows only & remove non-relevant columns as well.
   * Filtered with common ingredients:
     + common\_ingredients = ['tomato', 'egg', 'potato', 'onion', 'garlic', 'butter', 'milk', 'cheese', 'flour', 'sugar']
     + filtered\_df = recipes\_df[recipes\_df['ingredients'].str.lower().apply(lambda x: any(ing in x for ing in common\_ingredients))]
     + The filtered dataset contains 212401 rows and 12 columns.
   * Filtered for recipes with 1 <= minutes < 200 and fewer than 15 ingredients
     + filtered\_df = filtered\_df[(filtered\_df['n\_ingredients'] < 15) & (filtered\_df['minutes'] > 1) & (filtered\_df['n\_steps'] > 1) & (filtered\_df['minutes'] < 200)]
     + The filtered dataset contains 178935 rows and 12 columns.
   * Take a random sample of 100,000 recipes from the filtered dataset
     + filtered\_df = filtered\_df.sample(100000, random\_state=1)
     + The final sampled dataset contains 100000 rows and 12 columns.
   * Drop the unnecessary columns from the DataFrame
     + Columns:
     + filtered\_df = filtered\_df.drop(columns=['id', 'contributor\_id', 'submitted', 'tags', 'nutrition'])
   * Generate a list of unique IDs
     + num\_rows = filtered\_df.shape[0]
     + random\_ids = random.sample(range(1, 100001), num\_rows)
     + filtered\_df['id'] = random\_ids
     + cols = ['id'] + [col for col in filtered\_df.columns if col != 'id']
     + filtered\_df = filtered\_df[cols]
   * Save the filtered DataFrame to a CSV file
     + filtered\_df.to\_csv("filtered\_recipes.csv", index=False)

Tokenization and Embedding

1. Tokenize Ingredients:
   * Convert ingredients to list
     + def convert\_to\_list(ingredient\_str):
     + if isinstance(ingredient\_str, str):
     + try:
     + return eval(ingredient\_str)
     + except:
     + return []
     + else:
     + return ingredient\_str
     + filtered\_df['ingredients'] = filtered\_df['ingredients'].apply(convert\_to\_list)
   * Flatten Ingredients and Count
     + flattened\_ingredients = [ingredient.lower().strip() for ingredients in filtered\_df['ingredients'] for ingredient in ingredients]
     + ingredient\_counts = Counter(flattened\_ingredients)
     + unique\_ingredients = list(ingredient\_counts.keys())
     + Total unique ingredients: 11522
   * Define Stopwords:
     + Common words to ignore during tokenization.
     + stopwords = {'fresh', 'chopped', 'sliced', 'diced', 'organic', 'crushed', 'small', 'large', 'and', 'dried', 'free', 'mix', 'low', 'hot', 'of', 'with', 'red', 'white', 'black', 'brown', 'yellow', 'fat', 'light', 'style', 'in', 'lean', 'dark', 'french', 'smoked', 'roast'}
   * Create Token Dictionary and Tokenize Ingredients
     + builds the token dictionary for all unique ingredients.
     + token\_dict = {}
     + token\_id = 1
     + data = []
     + for ingredient in unique\_ingredients:
     + ingredient\_tokens = re.findall(r'\b\w+\b', ingredient.lower())
     + ingredient\_tokens = [token for token in ingredient\_tokens if token not in stopwords]
     + token\_ids = []
     + for token in ingredient\_tokens:
       - if token not in token\_dict:
       - token\_dict[token] = token\_id
       - token\_id += 1
       - token\_ids.append(token\_dict[token])
     + count = ingredient\_counts[ingredient]
     + data.append({'ingredient': ingredient, 'token\_ids': ', '.join(map(str, token\_ids)), 'count': count})
     + tokenized\_ingredient\_df = pd.DataFrame(data)
   * Save Token Dictionary for reference
     + token\_df = pd.DataFrame(list(token\_dict.items()), columns=['token', 'token\_id'])
   * Function to Tokenize Ingredient Lists
     + tokenizes each recipe's ingredient list using the dictionary
     + def tokenize\_ingredient\_list(ingredient\_list):
     + tokenized\_list = []
     + for ingredient in ingredient\_list:
     + ingredient\_tokens = re.findall(r'\b\w+\b', ingredient.lower())
     + ingredient\_tokens = [token for token in ingredient\_tokens if token not in stopwords]
     + token\_ids = [token\_dict[token] for token in ingredient\_tokens if token in token\_dict]
     + tokenized\_list.append(token\_ids)
     + return tokenized\_list
   * Apply Tokenization to Filtered DataFrame
     + filtered\_df\_tokenized = filtered\_df.copy()
     + filtered\_df\_tokenized['tokenized\_ingredients'] = filtered\_df\_tokenized['ingredients'].apply(tokenize\_ingredient\_list)
   * Save the filtered DataFrame to a CSV file
     + filtered\_df\_tokenized.to\_csv("tokenized\_dataset.csv", index=False)