Personalized Recipe Recommendation App

November 10, 2024

- By Yash Rajbhoj

Step 1: Prototype Selection

Abstract

The *Personalized Recipe Recommendation App* aims to revolutionize the way users approach meal planning and cooking by offering tailored recipe suggestions based on individual dietary needs, preferences, and health goals. Through the use of AI and machine learning, the app curates personalized meal plans, suggests alternative ingredients, and provides nutritional insights, making it an indispensable tool for anyone looking to improve their diet, save time, and explore new culinary experiences.

The project's prototype selection is driven by the need to validate core features, including recipe recommendation accuracy, user preference alignment, and seamless interaction. The initial prototype will be a small-scale model focusing on essential functionalities, such as a basic recommendation engine, a user-friendly interface, and a simple database of recipes with tagging for dietary and nutritional information. This prototype allows for real-time testing of the app's core algorithm and user experience before full-scale development.

The prototype selection thus serves as a critical phase for testing the product's feasibility, gathering user feedback, and refining key features, ensuring the app's alignment with market needs and technical objectives. This strategic approach enables efficient resource allocation, focusing on high-impact features and delivering value to users through personalized, accessible, and enjoyable culinary guidance.

1. Problem Statement

In today's fast-paced world, individuals face challenges in maintaining a balanced diet due to limited time, lack of culinary knowledge, and the overwhelming availability of recipes that may not suit their dietary needs or preferences.

Current recipe platforms often fail to provide personalized recommendations, leaving users to sift through irrelevant options that do not align with their health goals,

allergies, or taste preferences. This lack of personalization leads to inefficient meal planning, wasted food, and unmet nutritional goals.

The *Personalized Recipe Recommendation App* addresses these issues by leveraging AI and machine learning to provide tailored recipe suggestions, enabling users to:

- Discover recipes suited to their specific dietary restrictions (e.g., gluten-free, vegan).
- Receive meal plans aligned with their nutritional goals, such as weight loss or muscle gain.
- Minimize food waste by suggesting recipes based on available ingredients.

This solution aims to simplify meal planning, promote healthier eating habits, and enhance the overall cooking experience for users.

2. Market/Customer/Business Need Assessment

Market Assessment

The global health and wellness market is rapidly expanding, with increasing demand for personalized solutions to address dietary needs and improve eating habits. According to industry reports, the recipe app market is expected to grow significantly, driven by:

- The rise in health-conscious consumers.
- Increased interest in personalized diet plans and nutrition management.
- Technological advancements in AI and machine learning enabling more effective personalization.

This growing market presents an opportunity to cater to individuals seeking convenience, healthier lifestyles, and cost-effective meal solutions.

Customer Needs

The target customers include:

1. **Health-Conscious Individuals:** Users aiming to maintain or improve their health through tailored meal plans.
2. **Busy Professionals:** People with limited time who want quick and easy access to recipes based on available ingredients.
3. **Diet-Specific Users:** Individuals with dietary restrictions (e.g., vegan, keto, gluten-free) or food allergies.

4. **Food Enthusiasts:** Users who want to explore new recipes and cuisines while ensuring nutritional balance.

Key user needs include:

- Personalized recipe recommendations based on dietary preferences and goals.
- Integration with health devices for seamless tracking of caloric and nutritional intake.
- Simplified grocery planning to minimize food waste.

Business Need Assessment

The *Personalized Recipe Recommendation App* aligns with several business opportunities:

1. **Revenue Generation:** Through a freemium model, in-app purchases, targeted advertisements, and data licensing.
2. **User Retention:** Leveraging AI-driven personalization to create a unique and engaging experience.
3. **Partnerships:** Collaborations with grocery stores, meal kit providers, and fitness brands for affiliate marketing opportunities.
4. **Scalability:** A flexible platform that can expand features (e.g., fitness tracking, live cooking classes) based on market demand.

By addressing these market, customer, and business needs, the app aims to carve out a unique position in the competitive health and wellness technology landscape.

3. Target Specifications and Characterization

The *Personalized Recipe Recommendation App* is designed to meet specific performance, functional, and technical requirements to ensure it addresses the needs of users effectively. Below are the target specifications and their characterization:

Performance Specifications

1. **Recommendation Accuracy:**
    - Target: 85% alignment with user preferences (measured through user feedback and engagement).
    - Characterization: Leverages content-based and collaborative filtering algorithms.
2. **Response Time:**
    - Target: Recipe suggestions generated within 2 seconds of query input.

- Characterization: Optimized backend architecture and efficient algorithms to handle data processing.

3. **Scalability:**
    - Target: Support for up to 1 million users without performance degradation.
    - Characterization: Cloud-based infrastructure with scalable server resources.

Functional Specifications

1. **Personalization:**
    - Feature: Tailored recipes based on dietary preferences, health goals, and available ingredients.
    - Characterization: User profiles storing preferences, integrated with a recommendation engine.
2. **Ingredient-Based Search:**
    - Feature: Recipes filtered based on what users already have in their pantry.
    - Characterization: Ingredient-matching algorithm and a real-time searchable database.
3. **Nutritional Insights:**
    - Feature: Display of calorie counts, macronutrient breakdowns, and dietary suitability.
    - Characterization: Nutrition API integration for precise calculations.
4. **Interactive Features:**
    - Feature: Options to save, share, and review recipes.
    - Characterization: User-friendly interface with social media integration.

Technical Specifications

1. **Backend:**
    - Target: Python-based machine learning models for recommendation and AWS infrastructure for hosting.
    - Characterization: Trained models using libraries like TensorFlow or Scikit-learn, deployed via Flask or FastAPI.
2. **Frontend:**
    - Target: Intuitive, mobile-first design with responsive UI.
    - Characterization: Built using Flutter for cross-platform compatibility.

3. **Data Storage:**
    - Target: Secure storage for recipe data and user profiles.
    - Characterization: Use of cloud-based relational databases (e.g., MySQL, PostgreSQL) with encryption.

User Experience Specifications

1. **Ease of Use:**
    - Target: Users can complete recipe searches or meal planning in under 3 clicks.
    - Characterization: Minimalist interface with guided onboarding and intuitive navigation.
2. **Accessibility:**
    - Target: Compliance with WCAG standards for accessibility (e.g., text-to-speech support).
    - Characterization: Compatibility with assistive technologies for visually impaired users.

4. External Search (Information and Data Analysis)

To design and validate the *Personalized Recipe Recommendation App*, external research and data analysis are essential to understanding market trends, user preferences, and technical feasibility. Below is the overview of the external search process and key findings:

1. Market Research and Trends

- **Growing Market Size:**
  The global recipe apps market is projected to grow at a CAGR of 16.5% from 2023 to 2030, driven by health-conscious consumers and increased interest in cooking at home.
    - **Key Insight:** The demand for personalization in recipes is increasing, as users seek solutions that align with their dietary goals and time constraints.
- **Technology in Meal Planning:**
  AI-based apps are gaining traction for their ability to provide tailored recommendations, helping users save time and reduce food waste.
    - **Key Insight:** The integration of AI for real-time personalization enhances user engagement.

2. Competitor Analysis

- **Competitors Identified:**
  - *Tasty App:* Focuses on video-based recipes with a large user base.
  - *Yummly:* Offers ingredient-based filtering and dietary preference settings.
  - *Mealime:* Targets meal planning with grocery integration.
- **Gap Analysis:**
  - Most apps lack deep personalization features such as integration with wearable health devices or advanced nutritional insights.
  - Opportunity to differentiate through AI-powered features like real-time nutritional adjustments and ingredient-based suggestions.

3. User Behavior Data

- **Survey Results (Collected via online research):**
  - **75%** of users prefer apps that suggest recipes based on ingredients they already have.
  - **60%** are willing to pay for premium features like advanced meal planning or nutritional advice.
  - **50%** report frustration with recipe apps that do not consider dietary restrictions.
- **Key Insight:** Personalization, affordability, and ingredient-focused searches are high-priority features for users.

4. Dataset Collection for Validation

- **Data Sources:**
  - *Recipe Databases:* Kaggle, Food.com, and Recipe1M datasets with labeled dietary and ingredient information.
  - *Nutrition APIs:* USDA FoodData Central for nutritional insights.
  - *User Preferences Data:* Simulated user profiles created to validate recommendation algorithms.
- **Dataset Characteristics:**
  - Recipes with tags for cuisine type, dietary preferences (e.g., vegan, keto), and difficulty levels.
  - Nutritional data including calories, macronutrient breakdown, and serving sizes.
  - Ingredient lists for pantry-based recommendations.

5. Technical Insights from Research

- **Algorithmic Approaches:**
  - Collaborative filtering for user preference alignment.
  - Content-based filtering for ingredient and dietary tag matching.
  - Hybrid recommendation systems combining both approaches.
- **Technologies Explored:**
  - Machine learning frameworks: TensorFlow, Scikit-learn.
  - Cloud infrastructure: AWS for hosting and scalability.
  - Mobile app development: Flutter for cross-platform compatibility.

Key Findings from External Search:

1. A hybrid recommendation engine can effectively address user preferences and dietary needs.
2. Personalization and ease of use are critical factors for user retention.
3. Data availability is sufficient to create a robust prototype for validation.
4. Integration with APIs (e.g., grocery delivery, wearable devices) enhances the app's value proposition.

First import the basic libraries for data preprocessing:

```
import pandas as pd
import numpy as np
import re
import spacy
```

```
allrecipes_raw = pd.read_json('../__DATA__/recipes_raw/recipes_raw_nosource_ar.json')
epicurious_raw = pd.read_json('../__DATA__/recipes_raw/recipes_raw_nosource_epi.json')
foodnetwork_raw = pd.read_json('../__DATA__/recipes_raw/recipes_raw_nosource_fn.json')
```

```
allrecipes = allrecipes_raw.copy().T.reset_index().drop(columns = ['index'])
allrecipes.head(1)
```

	ingredients	instructions	picture_link	title
0	[4 skinless, boneless chicken breast halves AD...	Place the chicken, butter, soup, and onion in ...	55lznCYBbs2mT8BTx6BTkLhynGHzM.S	Slow Cooker Chicken and Dumplings

```
epicurious = epicurious_raw.copy().T.reset_index().drop(columns = ['index'])
epicurious.head(1)
```

	ingredients	instructions	picture_link	title
0	[12 egg whites, 12 egg yolks, 1 1/2 cups sugar...	Beat the egg whites until stiff, gradually add...	None	Christmas Eggnog

```
foodnetwork = foodnetwork_raw.copy().T.reset_index().drop(columns = ['index'])
foodnetwork.head(1)
```

	ingredients	instructions	picture_link	title
0	[1/2 cup celery, finely chopped, 1 small green...	Toss ingredients lightly and spoon into a butt...	None	Grammie Hamblet's Deviled Crab

```
recipes = pd.concat([allrecipes, epicurious, foodnetwork]).reset_index(drop=True) # Concat does not reset indices
recipes.shape
```

```
(125164, 4)
```

```
# Count of missing values by category
recipes.isna().sum()
```

```
ingredients       517
instructions      691
picture_link    42571
title             569
dtype: int64
```

```
# Number recipes/rows that have any missing values besides missing pictures
null_recs = recipes.copy().drop(columns = 'picture_link').T.isna().any()
null_recs.sum()
```

```
691
```

```
recipes[null_recs].head()
```

	ingredients	instructions	picture_link	title
5874	NaN	NaN	NaN	NaN
15020	NaN	NaN	NaN	NaN
15023	NaN	NaN	NaN	NaN
15025	NaN	NaN	NaN	NaN
15637	NaN	NaN	NaN	NaN

```
rows_to_drop = recipes[null_recs].index
recipes = recipes.drop(index = rows_to_drop).reset_index(drop = True)
recipes.shape
```

```
(124473, 4)
```

```
recipes.dtypes
```

```
ingredients     object
instructions    object
picture_link    object
title           object
dtype: object
```

```
# Indexing rows with columns that only contain numbers or punctuation
import string
nc_ingred_index = [index for i, index in zip(recipes['ingredients'], recipes.index) if all(j.isdigit() or j in string.punctu
nc_title_index = [index for i, index in zip(recipes['title'], recipes.index) if all(j.isdigit() or j in string.punctuation
nc_instr_index = [index for i, index in zip(recipes['instructions'], recipes.index) if all(j.isdigit() or j in string.punctu
```

```
# Checking number of rows in each category that are only punc/nums
index_list = [nc_ingred_index, nc_title_index, nc_instr_index]
[len(x) for x in index_list]
```

```
[1520, 0, 39]
```

```
# generating unique indices for index_list and dropping from dataframe
# recipes without recipe instructions or ingredients are not useable
from functools import reduce
from operator import add
inds_to_drop = set(reduce(add, index_list))
print(len(inds_to_drop))
recipes = recipes.drop(index=inds_to_drop).reset_index(drop=True)
recipes.shape
```

```
1551
(122922, 4)
```

```
# Recipe instructions with less than 20 characters are not good recipes
empty_instr_ind = [index for i, index in zip(recipes['instructions'], recipes.index) if len(i) < 20]
recipes = recipes.drop(index = empty_instr_ind).reset_index(drop=True)
```

```
recipes.shape
```

```
(122911, 4)
```

```
recipes.isna().sum()
```

```
ingredients         0
instructions        0
picture_link    41687
title               0
dtype: int64
```

```
# Checking for low ingredient recipes.
#Low_ingr_ind = [index for i, index in zip(recipes['ingredients'], recipes.index) if len(i) < 20]
low_ingr_index = [index for i, index in zip(recipes['ingredients'], recipes.index) if i[0] == np.nan]
len(low_ingr_index)
recipes.loc[low_ingr_index, 'ingredients']
```

```
Series([], Name: ingredients, dtype: object)
```

```
# Searching for pseudo empty lists
[index for i, index in zip(recipes['ingredients'], recipes.index) if np.nan in recipes.loc[index,'ingredients']]
```

```
[]
```

Cleaning to Prepare for Tokenizing

Cleaning Specifics:

- Removing ADVERTISEMENT
- Pruning dataset of rows with empty cells or inadequate recipes
- Remove all punctuation, digits, and extraneous spacing

```
# Removing ADVERTISEMENT text from ingredients list
ingredients = []
for ing_list in recipes['ingredients']:
    clean_ings = [ing.replace('ADVERTISEMENT','').strip() for ing in ing_list]
    if '' in clean_ings:
        clean_ings.remove('')
    ingredients.append(clean_ings)
recipes['ingredients'] = ingredients
```

```
recipes.loc[0,'ingredients']
```

```
['4 skinless, boneless chicken breast halves',
 '2 tablespoons butter',
 '2 (10.75 ounce) cans condensed cream of chicken soup',
 '1 onion, finely diced',
 '2 (10 ounce) packages refrigerated biscuit dough, torn into pieces']
```

```
# Extracting ingredients from their lists and formatting as single strings
recipes['ingredient_text'] = ['; '.join(ingredients) for ingredients in recipes['ingredients']]
recipes['ingredient_text'].head()
```

```
0    4 skinless, boneless chicken breast halves; 2 ...
1    2 (10.75 ounce) cans condensed cream of mushro...
2    1/2 cup packed brown sugar; 1/2 cup ketchup; 1...
3    1 cup butter, softened; 1 cup white sugar; 1 c...
4    8 ounces whole wheat rotini pasta; 3 cups fres...
Name: ingredient_text, dtype: object
```

```
# Counting the number of ingredients used in each recipe
recipes['ingredient_count'] = [len(ingredients) for ingredients in recipes['ingredients']]
```

```
recipes.head(1)
```

	ingredients	instructions	picture_link	title	ingredient_text	ingredient_count
0	[4 skinless, boneless chicken breast halves, 2...	Place the chicken, butter, soup, and onion in ...	55lznCYBbs2mT8BTx6BTkLhynGHzM.S	Slow Cooker Chicken and Dumplings	4 skinless, boneless chicken breast halves; 2 ...	5

```
all_text = recipes['title'] + ' ' + recipes['ingredient_text'] + ' ' + recipes['instructions']
all_text[0]
```

```
'Slow Cooker Chicken and Dumplings 4 skinless, boneless chicken breast halves; 2 tablespoons butter; 2 (10.75 ounce) cans co
ndensed cream of chicken soup; 1 onion, finely diced; 2 (10 ounce) packages refrigerated biscuit dough, torn into pieces Pla
ce the chicken, butter, soup, and onion in a slow cooker, and fill with enough water to cover.\nCover, and cook for 5 to 6 h
ours on High. About 30 minutes before serving, place the torn biscuit dough in the slow cooker. Cook until the dough is no l
onger raw in the center.\n'
```

```
# Clean_text Function
import string
import re

def clean_text(documents):
    cleaned_text = []
    for doc in documents:
        doc = doc.translate(str.maketrans('', '', string.punctuation)) # Remove Punctuation
        doc = re.sub(r'\d+', '', doc) # Remove Digits
        doc = doc.replace('\n',' ') # Remove New Lines
        doc = doc.strip() # Remove Leading White Space
        doc = re.sub(' +', ' ', doc) # Remove multiple white spaces
        cleaned_text.append(doc)
    return cleaned_text

# Cleaning Text
cleaned_text = clean_text(all_text)
```

```
cleaned_text[2]
```

```
'Brown Sugar Meatloaf cup packed brown sugar cup ketchup pounds lean ground beef cup milk eggs teaspoons salt teaspoon groun
d black pepper small onion chopped teaspoon ground ginger cup finely crushed saltine cracker crumbs Preheat oven to degrees
F degrees C Lightly grease a x inch loaf pan Press the brown sugar in the bottom of the prepared loaf pan and spread the ket
chup over the sugar In a mixing bowl mix thoroughly all remaining ingredients and shape into a loaf Place on top of the ketc
hup Bake in preheated oven for hour or until juices are clear'
```

Tokenizing Using Spacy

For this tokenization, we will lemmatize the words. This is will help create a denser word embeddings. However, no POS tagging, know entities, or noun_phrases will be parsed and added.

```
# Testing Strategies and Code
nlp = spacy.load('en')

' '.join([token.lemma_ for token in nlp(cleaned_text[2]) if not token.is_stop])
```

```
'Brown Sugar Meatloaf cup pack brown sugar cup ketchup pound lean ground beef cup milk egg teaspoon salt teaspoon ground bla
ck pepper small onion chop teaspoon ground ginger cup finely crush saltine cracker crumb Preheat oven to degree F degree C l
ightly grease a x inch loaf pan Press the brown sugar in the bottom of the prepared loaf pan and spread the ketchup over the
sugar in a mixing bowl mix thoroughly all remain ingredient and shape into a loaf Place on top of the ketchup Bake in prehea
t oven for hour or until juice be clear'
```

My current strategy is to strip down the text as much as possible. In this case that means lemmatizing words and removing stop words. The goal here is not text prediction, but similarity measures and keyword extraction, which don't require the semantic granularity that stop words and non-lemmatized words might provide.

```
# Tokenizing Function that lemmatizes words and removes Stop Words
def text_tokenizer(documents):
    tokenized_documents = []
    for doc in documents:
        tok_doc = ' '.join([token.lemma_ for token in nlp(doc) if not token.is_stop])
        tokenized_documents.append(tok_doc)
    return tokenized_documents
```

```
# Tokenizing Function to run in parallel
def text_tokenizer_mp(doc):
    tok_doc = ' '.join([token.lemma_ for token in nlp(doc) if not token.is_stop])
    return tok_doc
```

```
import multiprocessing as mp
print("Number of processors: ", mp.cpu_count())
```

```
Number of processors:  12
```

Creating Word Embeddings

- TF-IDF
- Pre-trained GloVe Word Embeddings
- GloVe Embeddings trained on the recipe corpora

In an attempt to create dense word embeddings, I could find no reliable examples to follow that integrate GloVe or Word2Vec with document topic modeling.

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(lowercase = True,
                             ngram_range = (1,1))

text_tfidf = vectorizer.fit_transform(tokenized_text)
tfidf_words = vectorizer.get_feature_names()
print(text_tfidf.shape)
print(len(tfidf_words))
```

```
(122911, 49785)
49785
```

Topic Modeling

- LDA
- NNMF

The ultimate goal with topic modeling is to group documents together and generate category words using TextRank. These category words can then be used to further refine the recommendation query

---

LDA and NNMF extract topic models by finding similar subgroups of text within the corpora of recipes (or other text documents). However

```
text_tfidf.shape
```

```
(122911, 49785)
```

```
from sklearn.decomposition import LatentDirichletAllocation as LDA

lda = LDA(n_components = 50,
          n_jobs = -1,
          max_iter = 100)
text_lda = lda.fit_transform(text_tfidf)
text_lda.shape
```

```
(122911, 50)
```

```
from sklearn.decomposition import NMF

nmf = NMF(alpha=0.0,
          init='nndsvdar',
          l1_ratio=0.0,
          max_iter = 100,
          n_components = 50,
          solver='cd')

text_nmf = nmf.fit_transform(text_tfidf)
text_nmf.shape
```

```
(122911, 50)
```

Models were arbitrarily set to 50 topics. Unfortunately, neither NNMF nor LDA have the ability to calculate the percentage of variance that they capture from the original tfidf matrix. So 50 topics is purely a shot in the dark.

Next Steps:

1. Document x Topic Matrix
2. Word x Topic Matrix

Exploring Topics by Document

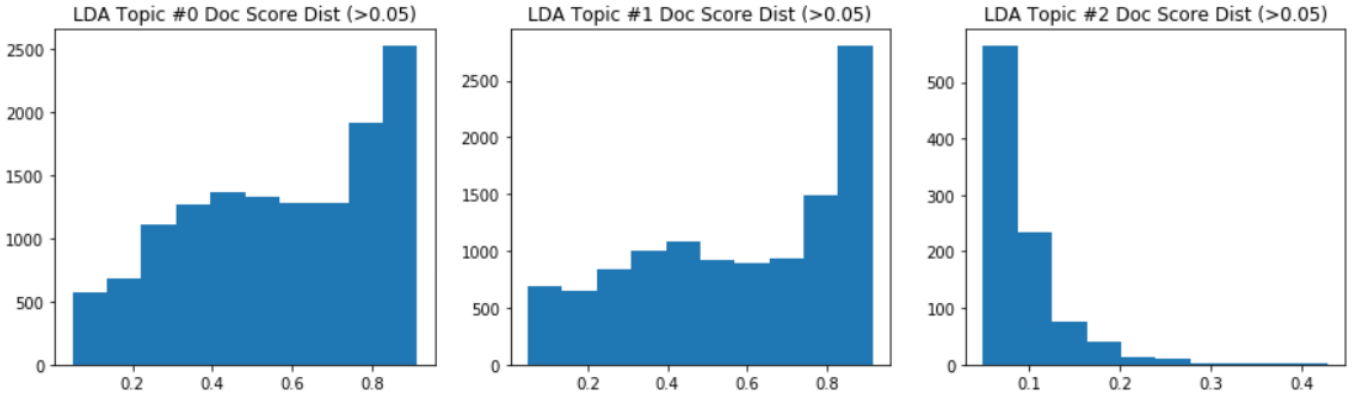
```
# variable dependencies:
text_series = pd.Series(all_text)

def docs_by_tops(top_mat, topic_range = (0,0), doc_range = (0,2)):
    for i in range(topic_range[0], topic_range[1]):
        topic_scores = pd.Series(top_mat[:,i])
        doc_index = topic_scores.sort_values(ascending = False)[doc_range[0]:doc_range[1]].index
        for j, index in enumerate(doc_index, doc_range[0]):
            print('Topic #{}'.format(i),
                  '\nDocument #{}'.format(j),
                  '\nTopic Score: {}\n\n'.format(topic_scores[index]),
                  text_series[index], '\n\n')
```

Analyzing Score distribution of document and word ranks within Topics

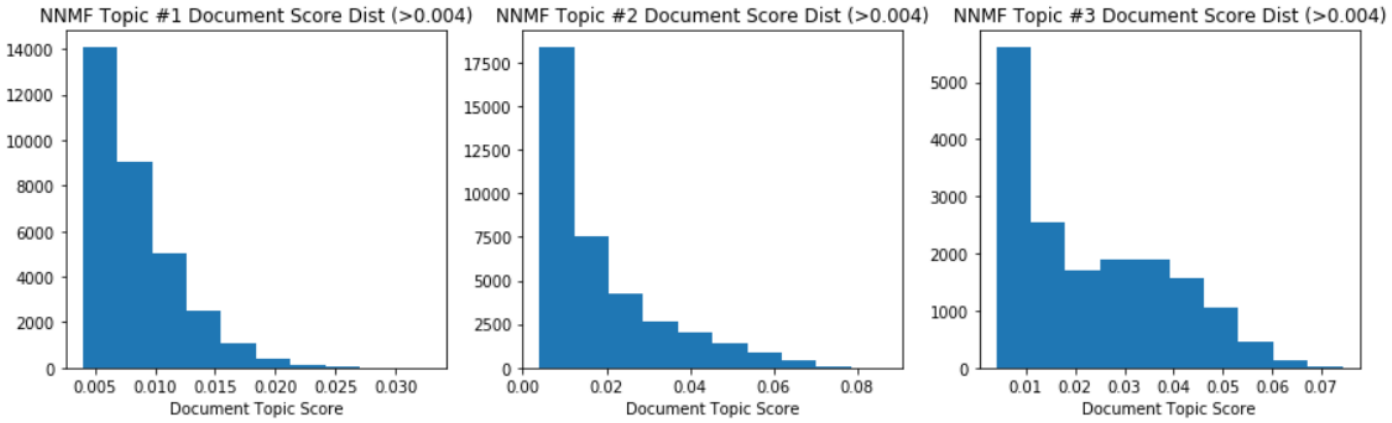
The purpose is to visualize the distribution of document topic rankings and decide the cutoff for the documents that associate most with that topic.

```
import matplotlib.pyplot as plt
# text_lda
# text_nmf
# ranked
```

```
# LDA Topic documents for topics 0-2
plt.figure(figsize=(15,4))
for i in range(3):
    series = pd.Series(text_lda[:,i])
    plt.subplot(1,3,i+1)
    plt.hist(series[series > 0.05])
    plt.title('LDA Topic #{} Doc Score Dist (>0.05)'.format(i+1))
plt.show()
```



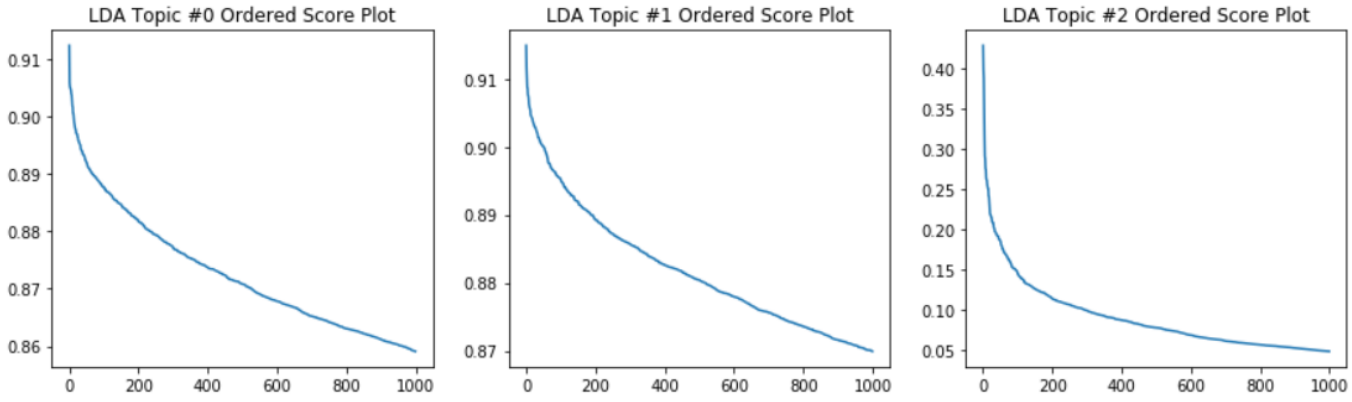
```
# NNMF Topic documents for topics 0-2
plt.figure(figsize=(15,4))
for i in range(3):
    series = pd.Series(text_nmf[:,i])
    plt.subplot(1,3,i+1)
    plt.hist(series[series > 0.004])
    plt.title('NNMF Topic #{} Document Score Dist (>0.004)'.format(i+1))
    plt.xlabel('Document Topic Score')
#plt.savefig('DocsByTop_Score_Distributions.png', transparent = True)
plt.show()
```



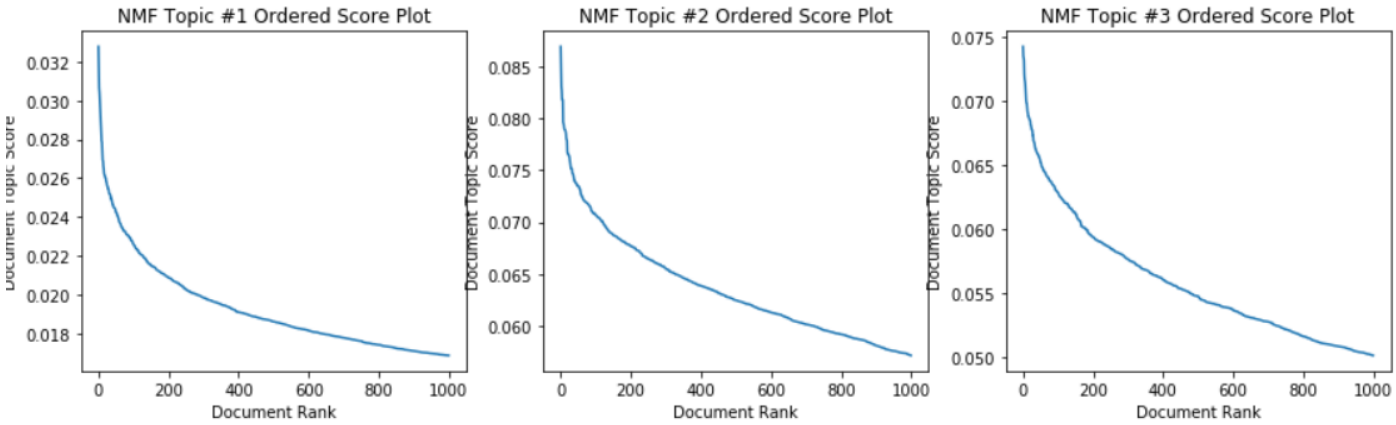
Based on the first three topics for LDA and NNMF, I will subjectly choose the top 1500-2000 documents. This number of documents seems to be where the loadings for each score distibution either level off or spike upward (an elbow so to speak).

Confirming with Topic #1 of the NNMF topics, the baking topic, baking recipes extended as far as 10,000 of the top ranked recipes within that category.

```
# LDA Topic document scores for topics 0-2
plt.figure(figsize=(15,4))
for i in range(3):
    series = pd.Series(text_lda[:,i]).copy().sort_values(ascending = False).reset_index(drop = True)
    plt.subplot(1,3,i+1)
    plt.plot(series[:1000])
    plt.title('LDA Topic #{} Ordered Score Plot'.format(i+1))
plt.show()
```




```
# NMF Topic document scores for topics 0-2
plt.figure(figsize=(15,4))
for i in range(3):
    series = pd.Series(text_nmf[:,i]).copy().sort_values(ascending = False).reset_index(drop = True)
    plt.subplot(1,3,i+1)
    plt.plot(series[:1000])
    plt.title('NMF Topic #{} Ordered Score Plot'.format(i+1))
    plt.xlabel('Document Rank')
    plt.ylabel('Document Topic Score')
#plt.savefig('DocsByTop_Score_Elbows.png', transparent = True)
plt.show()
```



While this method has not been previously utilized, the plots above take after scree plots, and they plot the scores of documents as they relate to their respective topics in descending order. By plotting these elbow plots, it might shed light on the best number of documents to use to rank words.

Testing the Algorithm

```
query = ['cinnamon', 'cream', 'banana']
Search_Recipes(query, query_ranked=True, recipe_range=(0,3))
```

```
Search Query: ['cinnamon', 'cream', 'banana']

Recipe Rank: 0   Banana Cinnamon Roll Casserole

Ingredients:
4 large stale cinnamon rolls; 1 1/2 cups milk; 5 eggs; 2 teaspoons vanilla extract; 2 teaspoons ground cinnamon; 2 bananas, d
ivided; 1/2 cup brown sugar

Instructions:
Slice cinnamon rolls into 1-inch squares. Whisk milk, eggs, vanilla extract, and cinnamon together in a bowl.
Slice 1 banana and layer on bottom of a 9x13-inch baking dish; top with brown sugar. Layer cinnamon roll squares on top of ba
nana. Pour egg mixture over cinnamon rolls; press cinnamon rolls with a spatula to allow egg mixture to fully coat. Cover wit
h aluminum foil and refrigerate until egg mixture is absorbed, 1 hour to overnight.
Preheat oven to 375 degrees F (190 degrees C).
Bake in preheated oven until firm, about 50 minutes. Remove aluminum foil and bake until golden brown, about 10 minutes more.
Remove from oven and let stand for 10 minutes. Slice remaining banana and place on top to serve.


  Recipe Rank: 1   Cinnamon Bread I

  Ingredients:
  2 cups all-purpose flour; 1 cup white sugar; 2 teaspoons baking powder; 1/2 teaspoon baking soda; 1 1/2 teaspoons ground cinn
  amon; 1 teaspoon salt; 1 cup buttermilk; 1/4 cup vegetable oil; 2 eggs; 2 teaspoons vanilla extract; 2 tablespoons white suga
  r; 1 teaspoon ground cinnamon; 2 teaspoons margarine

  Instructions:
  Preheat oven to 350 degrees F (175 degrees C). Grease one 9x5 inch loaf pan.
  Measure flour, 1 cup sugar, baking powder, baking soda, 1 1/2 teaspoons cinnamon, salt, buttermilk, oil, eggs and vanilla int
  o large mixing bowl. Beat 3 minutes. Pour into prepared loaf pan. Smooth top.
  Combine 2 tablespoons white sugar, 1 teaspoon cinnamon and butter, mixing until crumbly. Sprinkle topping over smoothed batte
  r. Using knife, cut in a light swirling motion to give a marbled effect.
  Bake for about 50 minutes. Test with toothpick. When inserted it should come out clean. Remove bread from pan to rack to coo
  l.


  Recipe Rank: 2   Banana Sour Cream Bread

  Ingredients:
  1/4 cup white sugar; 1 teaspoon ground cinnamon; 3/4 cup butter; 3 cups white sugar; 3 eggs; 6 very ripe bananas, mashed; 1
  (16 ounce) container sour cream; 2 teaspoons vanilla extract; 2 teaspoons ground cinnamon; 1/2 teaspoon salt; 3 teaspoons bak
  ing soda; 4 1/2 cups all-purpose flour; 1 cup chopped walnuts (optional)

  Instructions:
  Preheat oven to 300 degrees F (150 degrees C). Grease four 7x3 inch loaf pans. In a small bowl, stir together 1/4 cup white s
  ugar and 1 teaspoon cinnamon. Dust pans lightly with cinnamon and sugar mixture.
  In a large bowl, cream butter and 3 cups sugar. Mix in eggs, mashed bananas, sour cream, vanilla and cinnamon. Mix in salt, b
  aking soda and flour. Stir in nuts. Divide into prepared pans.
  Bake for 1 hour, until a toothpick inserted in center comes out clean.
```

```
# Test Rank
query = ['wine', 'cilantro','butter']
Search_Recipes(query, query_ranked=False, recipe_range=(0,3))
```

```
Search Query: ['wine', 'cilantro', 'butter']

Recipe Rank: 0   Cilantro Butter

Ingredients:
1 stick unsalted butter, slightly softened; 6 cloves garlic, coarsely chopped; 1/4 cup fresh cilantro leaves; 1 to 2 teaspoon
s fresh lime juice; Kosher salt and freshly ground pepper

Instructions:
Combine the butter, garlic, cilantro and 1 teaspoon lime juice in a food processor or with a mixer until smooth. Season with
salt and pepper, and add more lime juice if needed.
Photographs by Steve Giralt
```

```
Recipe Rank: 1   Cilantro Oil

Ingredients:
2 cups packed cilantro leaves; 1 cup lighttasting olive oil

Instructions:
Bring a large saucepan of water to a boil. Add the cilantro, making sure to push all the leaves under the boiling water. Blan
ch the cilantro for 5 seconds. Drain and immediately plunge into a bowl of ice water. Drain well and squeeze out all liquid.
In a blender, puree the cilantro with the olive oil. Strain the puree through 6 layers of cheesecloth or a paper coffee filte
r and put in a sterilized glass bottle, tightly capped.
Store in the refrigerator for up to 2 weeks.

Recipe Rank: 2   Cilantro Salt

Ingredients:
1 bunch cilantro leaves; 3/4 cup kosher salt; 3/4 cup sea salt

Instructions:
Place all ingredients in food processor and pulse until salt is bright green.
```

5. Benchmarking

Benchmarking involves analyzing existing products and services in the market to identify strengths, weaknesses, and opportunities for differentiation. For the *Personalized Recipe Recommendation App*, benchmarking provides insights into best practices and gaps that the app can address to create a competitive edge.

Competitors and Key Features

Competitor	Key Features	Strengths	Weaknesses
Yummly	Ingredient-based search, dietary preference filtering, smart meal planning	Strong personalization, grocery list integration, large recipe database	Limited nutritional insights, no integration with health devices
Tasty App	Video-based recipes, step-by-step cooking instructions	Visually appealing interface, engaging video content	Lack of advanced search and filtering options
Mealime	Meal planning with grocery list automation, recipes tailored to household size	Focused meal planning, easy grocery integration	Limited recipe variety, less emphasis on personalization
Whisk	Recipe organization, meal planning, grocery list features	Seamless grocery delivery integration, collaborative recipe sharing	Limited focus on dietary restrictions or nutritional tracking
AllRecipes	Community-driven recipes, ingredient matching, meal planning	Strong community engagement, large recipe database	Outdated interface, limited AI-driven personalization

6. Applicable Patents

While developing the *Personalized Recipe Recommendation App*, it is essential to identify and review relevant patents to avoid intellectual property infringement and ensure compliance with existing legal frameworks. Below are the key areas where patents might apply:

- *System and Method for Generating Personalized Recipe Recommendations*
- *Method and System for Nutritional Analysis of Recipes and Meal Plans*
- *Ingredient-Based Recipe Search and Filtering System*
- *System and Method for Monitoring Diet Using Wearable Devices*
- *Machine Learning Systems for Personalized Content Recommendations*

7. Applicable Regulations (Government and Environmental)

When developing the *Personalized Recipe Recommendation App*, it's essential to adhere to relevant government and environmental regulations to ensure compliance, protect user data, and promote sustainability.

8. Applicable Constraints

When developing the *Personalized Recipe Recommendation App*, several constraints influence the design, development, deployment, and maintenance of the product. These constraints can be categorized into technical, financial, legal, operational, and user-focused limitations.

9. Business Opportunity

The *Personalized Recipe Recommendation App* leverages the growing trends of digital health, convenience, and personalization to address the unmet needs of modern consumers. By focusing on personalized solutions for dietary planning and recipe discovery, the app taps into several lucrative markets and revenue opportunities.

10. Concept Generation

The *Personalized Recipe Recommendation App* is built on the premise of delivering AI-powered, customized recipe suggestions tailored to individual user preferences, dietary needs, and health goals. Concept generation involves brainstorming and refining various ideas to create a comprehensive solution that meets the identified needs and constraints.

11. Concept Development

Concept development involves refining the selected idea by translating it into a more detailed, tangible form. This includes designing the app architecture, finalizing features, determining the technical requirements, and developing an implementation strategy. Below is a breakdown of the *Personalized Recipe Recommendation App*'s concept development process.

12. Final Product Prototype / Product Details

The final product prototype of the **Personalized Recipe Recommendation App** aims to deliver a seamless user experience with features that blend cutting-edge technology, personalized dietary insights, and practical functionalities. Below is a detailed description of the app's features, functionalities, and design.

1. Support:

- **Definition:** Support is a measure of how frequently the items in a rule (e.g., A and B) appear in the dataset. It's calculated as the proportion of transactions that contain both items A and B.
- **Formula:**

$$\text{Support}(A \rightarrow B) = \frac{\text{Number of transactions containing both A and B}}{\text{Total number of transactions}}$$

  Interpretation: The higher the support, the more frequent the occurrence of the combination of items A and B in the dataset. For example, if 100 transactions contain both A and B, and there are 1000 total transactions, then the support is 0.1 (or 10%).

Confidence:

- **Definition:** Confidence is a measure of the likelihood that item B will be purchased when item A is purchased. It tells us how often items A and B are bought together, given that A is already bought.
- **Formula**

$$\text{Confidence}(A \rightarrow B) = \frac{\text{Support}(A \rightarrow B)}{\text{Support}(A)}$$

- **Interpretation:** It is a conditional probability that measures how often item B is purchased when item A is purchased. For example, if 50% of customers who buy A also buy B, then the confidence of the rule A→BA \rightarrow BA→B is 0.5.

3. Lift:

- **Definition:** Lift is a measure of how much more likely items A and B are bought together than by chance. It compares the observed frequency of A and B being bought together to the expected frequency if A and B were independent.
- **Formula**

$$\text{Lift}(A \rightarrow B) = \frac{\text{Confidence}(A \rightarrow B)}{\text{Support}(B)}$$

Or equivalently:

$$\text{Lift}(A \rightarrow B) = \frac{\text{Support}(A \rightarrow B)}{\text{Support}(A) \times \text{Support}(B)}$$

Interpretation: A lift value of:

- **Lift = 1** means A and B are independent (no relationship).
- **Lift > 1** indicates A and B are positively correlated (they occur together more often than expected by chance).
- **Lift < 1** suggests that A and B are negatively correlated (they occur together less often than expected).

A) Feasibility:

- **Definition:** Feasibility assesses whether it is technically and operationally possible to build and deliver the product or service. It focuses on the practical aspects of implementing the solution.
- **Key Areas of Focus:**
  - **Technical Feasibility:** Can the product be developed with the existing technology and resources? Does the team have the required technical expertise?
  - **Operational Feasibility:** Can the product be effectively deployed and supported? Does the infrastructure exist for production and maintenance?
  - **Resource Feasibility:** Are the necessary resources (time, money, skills, equipment) available to carry out the project successfully?
  - **Legal Feasibility:** Are there any legal or regulatory constraints that could prevent the development or deployment of the product?
- **Example:** A mobile app that integrates with IoT devices. The feasibility would look into whether the app can communicate with various devices, whether the technology for integration exists, and whether the team can implement this within the given timeline.

B) Viability:

- **Definition:** Viability evaluates the long-term sustainability and success of the product or business in the market. It considers whether the product can achieve its goals and stay competitive over time.
- **Key Areas of Focus:**
    - **Market Demand:** Is there a genuine need or demand for the product in the target market? Can it satisfy a market gap?
    - **Scalability:** Can the product grow and scale efficiently as the customer base increases? Does it have the potential to expand to new markets or customer segments?
    - **Competitive Advantage:** Does the product offer a unique value proposition or competitive edge? Can it differentiate itself from similar products in the market?
    - **Economic Viability:** Can the product generate enough revenue to cover its costs and achieve profitability? Does it have potential for profitability in the long term?
- **Example:** A subscription-based health app that provides personalized nutrition and fitness plans. The viability would consider if people are willing to pay for such a service, if competitors exist, and if the app can scale to handle increasing user demands.

C) Monetization:

- **Definition:** Monetization refers to the process of generating revenue from a product or service. It focuses on how the product can be priced and the strategies used to earn revenue from it.
- **Key Areas of Focus:**
    - **Revenue Model:** What pricing strategy will be used (e.g., subscription, one-time payment, freemium, pay-per-use)? How will the product earn money?
    - **Value Proposition:** How does the product create value that customers are willing to pay for? Does it solve a significant problem or provide unique benefits?
    - **Sales Channels:** How will the product be sold (e.g., app stores, website, third-party partnerships)?
    - **Cost Structure:** What are the costs involved in delivering the product (e.g., development, marketing, maintenance)? What is the break-even point?

- **Revenue Streams:** Will the product rely on a single revenue stream or multiple sources (e.g., advertisements, in-app purchases, premium features)?

- **Example:** For a mobile app, monetization could involve offering a free basic version with ads and a premium version without ads. Alternatively, it could charge users a one-time fee or a monthly subscription for access to advanced features.

Step 2: Prototype Development

Github Link: https://github.com/Yash-Rajbhoj2001/Personalized-Recipe-Recommendation-App.git

Step 3: Business Modeling

Business modeling is a crucial step in the development of any product or service. It involves defining how the product or service will create value, capture that value, and sustain its operations over time. The business model serves as a blueprint for how your business will generate revenue, satisfy customer needs, and scale in the marketplace.

- **Value Proposition:** Offers personalized, health-conscious recipes based on user preferences, dietary restrictions, and available ingredients.
- **Customer Segments:** Health-conscious individuals, busy professionals, people with specific dietary needs (e.g., vegans, gluten-free).
- **Revenue Streams:** Freemium model (free basic version with ads, premium subscription for personalized features, no ads).
- **Cost Structure:** Development and maintenance costs, marketing, hosting fees, customer support.
- **Channels:** App stores, website, social media, email marketing.
- **Customer Relationships:** Personalized recommendations, community engagement (forums, social media).
- **Key Resources:** AI algorithms, app development platform, recipe database.
- **Key Activities:** App development, algorithm refinement, user acquisition, marketing.
- **Key Partners:** Grocery delivery services, health organizations, advertisers.
- **Market Strategy:** Partnering with influencers, targeted online ads, SEO for recipe-related searches.

Step 4: Financial Modeling

Financial modeling helps to quantify the financial health and viability of your product or service by defining revenue, costs, and profitability. In this step, we will create a simple financial equation to model revenue and estimate profit based on key variables.

Key Components of Financial Modeling:

1. **Product Price (P):** The selling price of the product per unit.
2. **Cost to Produce (C):** The cost involved in producing and delivering one unit of the product or service.
3. **Number of Units Sold (x):** The total number of units you sell in a given period (e.g., month, year).
4. **Total Revenue (R):** The total income generated from sales in a specific period.
5. **Fixed Costs (F):** Costs that remain constant regardless of the number of units sold (e.g., rent, salaries, utilities).
6. **Total Profit (T):** The actual profit made after deducting the cost to produce and fixed costs from total revenue.

Example Calculation:

Let's take an example where:

- **Product price (P):** Rs. 500 per unit
- **Cost to produce (C):** Rs. 200 per unit
- **Fixed costs (F):** Rs. 2000 per month
- **Units sold (x):** 300 units in June

Step 1: Calculate the Total Revenue (R)
Total revenue (R) is calculated by multiplying the price per unit (P) by the number of units sold (x):

$$R = P \times x$$

$$R = 500 \times 300 = 1,50,000$$

Step 2: Calculate the Total Cost (C)
The total cost (C) is calculated by multiplying the cost to produce one unit (C_unit) by the number of units sold (x), and adding the fixed costs (F):

$$C = (C_{\text{unit}} \times x) + F$$

$$C = (200 \times 300) + 2000 = 60,000 + 2000 = 62,000$$

Step 3: Calculate the Total Profit (T)
Total profit (T) is calculated by subtracting the total cost (C) from the total revenue (R):

$$T = R - C$$

$$T = 1,50,000 - 62,000 = 88,000$$

Step 4: General Financial Equation
Now, we can summarize this in a general form for the financial model:

- **Revenue (R) = P × x**
- **Cost (C) = (C_unit × x) + F**
- **Profit (T) = R - C**
  Or,
  Profit (T) = (P × x) - [(C_unit × x) + F]

In this case:

$$T = (500 \times x) - [(200 \times x) + 2000]$$

Where:

- **T** = Profit
- **x** = Units sold
- **P** = Price per unit (500)
- **C_unit** = Cost to produce per unit (200)
- **F** = Fixed costs (2000)

Example Equation for Sales Number (x):

If you know the price per unit (P), cost to produce (C_unit), and fixed costs (F), the total revenue becomes a function of sales (x):

$$y = 500x - 2000$$

Where:

- **y** = Total revenue
- **x** = Number of units sold

Conclusion:

By understanding how each of these components interacts, you can create a dynamic financial model that forecasts profits based on different sales numbers,

product pricing, and production costs. Adjusting these variables helps optimize pricing strategies and cost control.