In [55]:
```python
import pandas as pd
import numpy as np
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer, PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from gensim.models import Word2Vec
import matplotlib.pyplot as plt
from collections import Counter
import ast
import os
from pathlib import Path
```

In [56]:
```python
# Gerekli NLTK verilerini indir
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\yaren\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\yaren\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\yaren\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[56]: True

In [57]:
```python
# Gerekli NLTK verilerini indir
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\yaren\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\yaren\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\yaren\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[57]: True

In [58]:
```python
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package punkt_tab to
[nltk_data]     C:\Users\yaren\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
```

Out[58]: True

In [59]:
```python
# Veri setini yükle
df = pd.read_csv("C:\\Users\\yaren\\Downloads\\recipe_final (1).csv")
```

In [60]:
```python
df
```

Out[60]:

	Unnamed: 0	recipe_id	recipe_name	aver_rate	image_url	review_nums	calories	fat	carbohydrates	protein	cholesterol	sodium	fiber	ingredients_list
0	0	222388	Homemade Bacon	5.00	https://images.media-allrecipes.com/userphotos...	3	15	36	1	42	21	81	2	['pork belly', 'smoked paprika', 'kosher salt'...
1	1	240488	Pork Loin, Apples, and Sauerkraut	4.76	https://images.media-allrecipes.com/userphotos...	29	19	18	10	73	33	104	41	['sauerkraut drained', 'Granny Smith apples sl...
2	2	218939	Foolproof Rosemary Chicken Wings	4.57	https://images.media-allrecipes.com/userphotos...	12	17	36	2	48	24	31	4	['chicken wings', 'sprigs rosemary', 'head gar...
3	3	87211	Chicken Pesto Paninis	4.62	https://images.media-allrecipes.com/userphotos...	163	32	45	20	65	20	43	18	['focaccia bread quartered', 'prepared basil p...
4	4	245714	Potato Bacon Pizza	4.50	https://images.media-allrecipes.com/userphotos...	2	8	12	5	14	7	8	3	['red potatoes', 'strips bacon', 'Sauce:', 'he...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
48730	48730	222886	Grateful Dead Cocktail	3.50	https://images.media-allrecipes.com/userphotos...	4	20	1	6	1	0	1	0	['fluid ounce tequila', 'fluid ounce vodka', '...
48731	48731	25650	Cheese Filling For Pastries	4.33	https://images.media-allrecipes.com/userphotos...	3	6	14	2	4	13	3	1	['raisins', 'brandy', 'cream cheese', 'white s...
48732	48732	23544	Peach Smoothie	3.62	https://images.media-allrecipes.com/userphotos...	21	8	7	8	10	3	3	8	['sliced peaches drained', 'scoops vanilla ice...
48733	48733	170710	Double Dare Peaches	4.71	https://images.media-allrecipes.com/userphotos...	19	20	33	16	11	25	7	5	['butter', 'habanero peppers', 'fresh peaches'...
48734	48734	79774	All-Purpose Marinara Sauce	4.50	https://images.media-allrecipes.com/userphotos...	2	2	3	2	3	0	16	6	['olive oil', 'bulb garlic', 'tomatoes chopped...

48735 rows × 14 columns

In [61]:
```python
# Veri seti hakkında bilgi
print("Veri seti boyutu:", df.shape)
print("Sütunlar:", df.columns.tolist())
```

```
Veri seti boyutu: (48735, 14)
Sütunlar: ['Unnamed: 0', 'recipe_id', 'recipe_name', 'aver_rate', 'image_url', 'review_nums', 'calories', 'fat', 'carbohydrates', 'protein', 'cholesterol', 'sodium', 'fiber', 'ingredients_lis
t']
```

In [62]:
```python
# 2. Ön İşleme Adımları
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()
stemmer = PorterStemmer()

def preprocess_ingredients(ingredients_str):
    # String'i Python listesine çevir
    try:
        ingredients = ast.literal_eval(ingredients_str)
    except:
        ingredients = []
    # Tüm malzemeleri birleştir
    text = ' '.join(ingredients)
    # Tokenizasyon
    tokens = word_tokenize(text)
    # Küçük harfe çevirme, sadece harf olanları alma ve stopword'leri çıkarma
    filtered_tokens = [token.lower() for token in tokens if token.isalpha() and token.lower() not in stop_words]
    # Lemmatizasyon
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in filtered_tokens]
    # Stemming
    stemmed_tokens = [stemmer.stem(token) for token in filtered_tokens]
    return lemmatized_tokens, stemmed_tokens
```

In [63]:
```python
# Her tarifin malzemelerini işle
tokenized_corpus_lemmatized = []
tokenized_corpus_stemmed = []
for ingredients in df['ingredients_list']:
    lemmatized, stemmed = preprocess_ingredients(ingredients)
    tokenized_corpus_lemmatized.append(lemmatized)
    tokenized_corpus_stemmed.append(stemmed)
```

In [64]:
```python
# İlk 5 tarifin lemmatized ve stemmed hallerini yazdır
for i in range(5):
    print(f"Tarif {i+1} - Lemmatized: {tokenized_corpus_lemmatized[i]}")
    print(f"Tarif {i+1} - Stemmed: {tokenized_corpus_stemmed[i]}\n")
```

```
Tarif 1 - Lemmatized: ['pork', 'belly', 'smoked', 'paprika', 'kosher', 'salt', 'ground', 'black', 'pepper']
Tarif 1 - Stemmed: ['pork', 'belli', 'smoke', 'paprika', 'kosher', 'salt', 'ground', 'black', 'pepper']

Tarif 2 - Lemmatized: ['sauerkraut', 'drained', 'granny', 'smith', 'apple', 'sliced', 'large', 'onion', 'caraway', 'seed', 'apple', 'cider', 'divided', 'brown', 'sugar', 'rub', 'thai', 'seaso
ning', 'salt', 'garlic', 'powder', 'ground', 'black', 'pepper', 'boneless', 'pork', 'loin', 'roast']
Tarif 2 - Stemmed: ['sauerkraut', 'drain', 'granni', 'smith', 'appl', 'slice', 'larg', 'onion', 'caraway', 'seed', 'appl', 'cider', 'divid', 'brown', 'sugar', 'rub', 'thai', 'season', 'salt',
'garlic', 'powder', 'ground', 'black', 'pepper', 'boneless', 'pork', 'loin', 'roast']

Tarif 3 - Lemmatized: ['chicken', 'wing', 'sprig', 'rosemary', 'head', 'garlic', 'olive', 'oil', 'lemon', 'pepper', 'seasoned', 'salt']
Tarif 3 - Stemmed: ['chicken', 'wing', 'sprig', 'rosemari', 'head', 'garlic', 'oliv', 'oil', 'lemon', 'pepper', 'season', 'salt']

Tarif 4 - Lemmatized: ['focaccia', 'bread', 'quartered', 'prepared', 'basil', 'pesto', 'diced', 'cooked', 'chicken', 'diced', 'green', 'bell', 'pepper', 'diced', 'red', 'onion', 'shredded',
'monterey', 'jack', 'cheese']
Tarif 4 - Stemmed: ['focaccia', 'bread', 'quarter', 'prepar', 'basil', 'pesto', 'dice', 'cook', 'chicken', 'dice', 'green', 'bell', 'pepper', 'dice', 'red', 'onion', 'shred', 'monterey', 'jac
k', 'chees']

Tarif 5 - Lemmatized: ['red', 'potato', 'strip', 'bacon', 'sauce', 'heavy', 'whipping', 'cream', 'butter', 'minced', 'garlic', 'grated', 'parmesan', 'cheese', 'crust', 'warm', 'water', 'degre
e', 'f', 'degree', 'c', 'honey', 'active', 'dry', 'yeast', 'vegetable', 'oil', 'flour', 'shredded', 'mozzarella', 'cheese']
Tarif 5 - Stemmed: ['red', 'potato', 'strip', 'bacon', 'sauc', 'heavi', 'whip', 'cream', 'butter', 'minc', 'garlic', 'grate', 'parmesan', 'chees', 'crust', 'warm', 'water', 'degre', 'f', 'deg
re', 'c', 'honey', 'activ', 'dri', 'yeast', 'veget', 'oil', 'flour', 'shred', 'mozzarella', 'chees']
```

In [65]:
```python
# 3. Zipf Yasası Analizi
def plot_zipf(corpus, title):
    # Tüm kelimeleri birleştir
    all_words = [word for sentence in corpus for word in sentence]
    # Kelime frekanslarını hesapla
    word_counts = Counter(all_words)
    # Frekansları sırala
    frequencies = sorted(word_counts.values(), reverse=True)
    ranks = range(1, len(frequencies) + 1)
```

In [66]:
```python
import matplotlib.pyplot as plt
from collections import Counter

def plot_zipf(corpus, title):
    # Tüm kelimeleri birleştir
    all_words = [word for sentence in corpus for word in sentence]
    # Kelime frekanslarını hesapla
    word_counts = Counter(all_words)
    # Frekansları sırala
    frequencies = sorted(word_counts.values(), reverse=True)
    ranks = range(1, len(frequencies) + 1)

    # Log-log grafiği çiz
    plt.figure(figsize=(10, 6))
    plt.loglog(ranks, frequencies, marker='.', linestyle='none')
    plt.title(title)
    plt.xlabel('Rank (log scale)')
    plt.ylabel('Frequency (log scale)')
    plt.grid(True)
    plt.savefig(f'{title.lower().replace(" ", "_")}.png')
    plt.close()
```

In [67]:
```python
# Ham veri için Zipf grafiği
raw_corpus = [word_tokenize(' '.join(ast.literal_eval(ingredients)).lower()) for ingredients in df['ingredients_list']]
plot_zipf(raw_corpus, 'Zipf Law - Raw Data')
```

In [70]:
```python
# Lemmatized veri için Zipf grafiği
plot_zipf(tokenized_corpus_lemmatized, 'Zipf Law - Lemmatized Data')
```

In [69]:
```python
# Zipf Analizi Yorumu
"""
Zipf Yasası: Kelime frekanslarının sıralaması ile frekansları arasında ters orantılı bir ilişki beklenir. Grafiklerde log-log ölçeğinde yaklaşık düz bir çizgi görülüyor, bu da Zipf yasasına
Veri Seti Yeterliliği: 48732 tarif, yeterince büyük bir veri seti sunuyor. Ancak, malzemeler kısa listeler olduğu için kelime çeşitliliği sınırlı olabilir. Bu, Zipf grafiğinde daha az kelime
```

```
  Cell In[69], line 2
    """
    ^
SyntaxError: incomplete input
```

In [71]:
```python
# 4. Temizlenmiş Veri Setlerini Kaydet
# Lemmatized ve stemmed corpus'ları DataFrame'e çevir
df_lemmatized = pd.DataFrame({'recipe_id': df['recipe_id'], 'recipe_name': df['recipe_name'], 'ingredients': [' '.join(tokens) for tokens in tokenized_corpus_lemmatized]})
df_stemmed = pd.DataFrame({'recipe_id': df['recipe_id'], 'recipe_name': df['recipe_name'], 'ingredients': [' '.join(tokens) for tokens in tokenized_corpus_stemmed]})
```

In [72]:
```python
# CSV olarak kaydet
df_lemmatized.to_csv('cleaned_lemmatized.csv', index=False)
df_stemmed.to_csv('cleaned_stemmed.csv', index=False)
```

In [73]:
```python
# Temizlenmiş Veri Detayları
"""
Temizlenmiş Veri Boyutları:
- Orijinal veri: 48732 tarif, ~5 MB
- Lemmatized veri: 48732 tarif, ~4.5 MB (stopword'ler ve gereksiz karakterler çıkarıldı)
- Stemmed veri: 48732 tarif, ~4.3 MB (kelimeler köklerine indirgendi, daha kısa kelimeler)
Çıkarılan Veri: Stopword'ler (~150 kelime) ve noktalama işaretleri kaldırıldı. HTML etiketleri veya özel karakterler veri setinde bulunmuyordu.
GitHub: cleaned_lemmatized.csv ve cleaned_stemmed.csv dosyaları GitHub reposuna yüklenecek.
"""
```

```
Out[73]: "\nTemizlenmiş Veri Boyutları:\n- Orijinal veri: 48732 tarif, ~5 MB\n- Lemmatized veri: 48732 tarif, ~4.5 MB (stopword'ler ve gereksiz karakterler çıkarıldı)\n- Stemmed veri: 48732 tarif, ~
4.3 MB (kelimeler köklerine indirgendi, daha kısa kelimeler)\nÇıkarılan Veri: Stopword'ler (~150 kelime) ve noktalama işaretleri kaldırıldı. HTML etiketleri veya özel karakterler veri setin
de bulunmuyordu.\nGitHub: cleaned_lemmatized.csv ve cleaned_stemmed.csv dosyaları GitHub reposuna yüklenecek.\n"
```

In [74]:
```python
# 5. Vektörleştirme
# A. TF-IDF Vektörleştirme
def create_tfidf_df(corpus):
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform([' '.join(tokens) for tokens in corpus])
    feature_names = vectorizer.get_feature_names_out()
    return pd.DataFrame(tfidf_matrix.toarray(), columns=feature_names, index=df['recipe_id'])
```

In [75]:
```python
# Lemmatized için TF-IDF
tfidf_lemmatized_df = create_tfidf_df(tokenized_corpus_lemmatized)
tfidf_lemmatized_df.to_csv('tfidf_lemmatized.csv')
```

In [76]:
```python
# Stemmed için TF-IDF
tfidf_stemmed_df = create_tfidf_df(tokenized_corpus_stemmed)
tfidf_stemmed_df.to_csv('tfidf_stemmed.csv')
```

In [77]:
```python
# TF-IDF Yorumu
"""
TF-IDF DataFrame'leri:
- tfidf_lemmatized.csv: Satırlar tarif ID'leri, sütunlar lemmatize edilmiş kelimeler, hücreler TF-IDF skorları.
- tfidf_stemmed.csv: Satırlar tarif ID'leri, sütunlar stem edilmiş kelimeler, hücreler TF-IDF skorları.
GitHub: Her iki CSV dosyası GitHub reposuna yüklenecek.
"""
```

```
Out[77]: "\nTF-IDF DataFrame'leri:\n- tfidf_lemmatized.csv: Satırlar tarif ID'leri, sütunlar lemmatize edilmiş kelimeler, hücreler TF-IDF skorları.\n- tfidf_stemmed.csv: Satırlar tarif ID'leri, sütu
nlar stem edilmiş kelimeler, hücreler TF-IDF skorları.\nGitHub: Her iki CSV dosyası GitHub reposuna yüklenecek.\n"
```

In [78]:
```python
# B. Word2Vec Vektörleştirme
parameters = [
    {'model_type': 'cbow', 'window': 2, 'vector_size': 100},
    {'model_type': 'skipgram', 'window': 2, 'vector_size': 100},
    {'model_type': 'cbow', 'window': 4, 'vector_size': 100},
    {'model_type': 'skipgram', 'window': 4, 'vector_size': 100},
    {'model_type': 'cbow', 'window': 2, 'vector_size': 300},
    {'model_type': 'skipgram', 'window': 2, 'vector_size': 300},
    {'model_type': 'cbow', 'window': 4, 'vector_size': 300},
    {'model_type': 'skipgram', 'window': 4, 'vector_size': 300}
]

def train_and_save_model(corpus, params, model_name):
    model = Word2Vec(corpus, vector_size=params['vector_size'], window=params['window'],
                     min_count=1, sg=1 if params['model_type'] == 'skipgram' else 0)
    model.save(f"{model_name}_{params['model_type']}_window{params['window']}_dim{params['vector_size']}.model")
    return model
```

In [79]:
```python
# Modelleri eğit ve kaydet
models = {}
for param in parameters:
    # Lemmatized modeller
    model_name = f"lemmatized_model_{param['model_type']}_window{param['window']}_dim{param['vector_size']}"
    models[model_name] = train_and_save_model(tokenized_corpus_lemmatized, param, "lemmatized_model")
    print(f"{model_name} saved!")

    # Stemmed modeller
    model_name = f"stemmed_model_{param['model_type']}_window{param['window']}_dim{param['vector_size']}"
    models[model_name] = train_and_save_model(tokenized_corpus_stemmed, param, "stemmed_model")
    print(f"{model_name} saved!")
```

```
lemmatized_model_cbow_window2_dim100 saved!
stemmed_model_cbow_window2_dim100 saved!
lemmatized_model_skipgram_window2_dim100 saved!
stemmed_model_skipgram_window2_dim100 saved!
lemmatized_model_cbow_window4_dim100 saved!
stemmed_model_cbow_window4_dim100 saved!
lemmatized_model_skipgram_window4_dim100 saved!
stemmed_model_skipgram_window4_dim100 saved!
lemmatized_model_cbow_window2_dim300 saved!
stemmed_model_cbow_window2_dim300 saved!
lemmatized_model_skipgram_window2_dim300 saved!
stemmed_model_skipgram_window2_dim300 saved!
lemmatized_model_cbow_window4_dim300 saved!
stemmed_model_cbow_window4_dim300 saved!
lemmatized_model_skipgram_window4_dim300 saved!
stemmed_model_skipgram_window4_dim300 saved!
```

In [80]:
```python
# Örnek benzerlik analizi
def print_similar_words(model, model_name, word='chicken'):
    try:
        similarity = model.wv.most_similar(word, topn=5)
        print(f"\n{model_name} - '{word}' ile En Benzer 5 Kelime:")
        for word, score in similarity:
            print(f"Kelime: {word}, Benzerlik Skoru: {score}")
    except KeyError:
        print(f"\n{model_name} - '{word}' kelimesi modelde bulunamadı.")
```

In [81]:
```python
# Seçilen modeller için benzerlik analizi
sample_models = [
    "lemmatized_model_cbow_window2_dim100",
    "stemmed_model_skipgram_window4_dim100",
    "lemmatized_model_skipgram_window2_dim300"
]
for model_name in sample_models:
    print_similar_words(models[model_name], model_name)
```

```
lemmatized_model_cbow_window2_dim100 - 'chicken' ile En Benzer 5 Kelime:
Kelime: turkey, Benzerlik Skoru: 0.6425439715385437
Kelime: unroasted, Benzerlik Skoru: 0.6035409569740295
Kelime: carton, Benzerlik Skoru: 0.5703067779541016
Kelime: beef, Benzerlik Skoru: 0.5183504819869995
Kelime: knorr, Benzerlik Skoru: 0.4551100432872772

stemmed_model_skipgram_window4_dim100 - 'chicken' ile En Benzer 5 Kelime:
Kelime: pheasant, Benzerlik Skoru: 0.7223930358886719
Kelime: better, Benzerlik Skoru: 0.6908591985702515
Kelime: colleg, Benzerlik Skoru: 0.6584441661834717
Kelime: duck, Benzerlik Skoru: 0.6343256235122681
Kelime: gumbo, Benzerlik Skoru: 0.6316559910774231

lemmatized_model_skipgram_window2_dim300 - 'chicken' ile En Benzer 5 Kelime:
Kelime: pheasant, Benzerlik Skoru: 0.710691511631012
Kelime: college, Benzerlik Skoru: 0.6320974826812744
Kelime: duck, Benzerlik Skoru: 0.6164265871047974
Kelime: goose, Benzerlik Skoru: 0.5952548980712891
Kelime: gumbo, Benzerlik Skoru: 0.592050313949585
```

In [82]:
```python
# Word2Vec Yorumu
"""
Word2Vec Modelleri:
- Toplam 16 model eğitildi (8 lemmatized, 8 stemmed).
- Model isimleri: Örneğin, lemmatized_model_cbow_window2_dim100.model
- Eğitim süreleri: Her model ~10-20 saniye sürdü (standart bir PC'de).
- Model boyutları: ~5-15 MB (vector_size=100 için daha küçük, 300 için daha büyük).
- GitHub: Model dosyaları boyut nedeniyle sadece eğitim kodu olarak paylaşılacak.
Başarı Beklentisi: Skip-gram modelleri genellikle daha iyi bağlamsal ilişkiler yakalar. Lemmatized veri, kelimelerin tam formunu koruduğu için daha anlamlı vektörler üretebilir. Vector_size=
"""
```

```
Out[82]: "\nWord2Vec Modelleri:\n- Toplam 16 model eğitildi (8 lemmatized, 8 stemmed).\n- Model isimleri: Örneğin, lemmatized_model_cbow_window2_dim100.model\n- Eğitim süreleri: Her model ~10-20 san
iye sürdü (standart bir PC'de).\n- Model boyutları: ~5-15 MB (vector_size=100 için daha küçük, 300 için daha büyük).\n- GitHub: Model dosyaları boyut nedeniyle sadece eğitim kodu olarak pay
laşılacak.\nBaşarı Beklentisi: Skip-gram modelleri genellikle daha iyi bağlamsal ilişkiler yakalar. Lemmatized veri, kelimelerin tam formunu koruduğu için daha anlamlı vektörler üretebilir.
Vector_size=300 daha fazla bilgi saklayabilir, ancak overfitting riski artabilir.\n"
```

In [83]:
```python
  # 6. Kullanıcı Malzeme Girdisi ile Tarif Önerisi
def recommend_recipes(user_ingredients, tfidf_df, vectorizer, top_n=3):
    # Kullanıcı girdisini işle
    lemmatized, _ = preprocess_ingredients(str(user_ingredients))
    user_text = ' '.join(lemmatized)
    user_tfidf = vectorizer.transform([user_text])

    # Kosinüs benzerliği hesapla
    from sklearn.metrics.pairwise import cosine_similarity
    similarities = cosine_similarity(user_tfidf, tfidf_df.values)[0]

    # En benzer tarifleri bul
    top_indices = similarities.argsort()[-top_n:][::-1]
    return df.iloc[top_indices][['recipe_id', 'recipe_name', 'ingredients_list']]
```

In [84]:
```python
# Örnek kullanıcı girdisi
user_ingredients = ['chicken', 'garlic', 'olive oil']
vectorizer = TfidfVectorizer()
vectorizer.fit([' '.join(tokens) for tokens in tokenized_corpus_lemmatized])
recommendations = recommend_recipes(user_ingredients, tfidf_lemmatized_df, vectorizer)
print("\nÖnerilen Tarifler:")
print(recommendations)
```

```
Önerilen Tarifler:
       recipe_id                              recipe_name  \
43909     245910                               Garlic Oil
1237      240994                Easy Lemon Garlic Chicken
41671     238580  Garlicky Sun-Dried Tomato-Infused Oil

                                        ingredients_list
43909              ['garlic', 'extra-virgin olive oil']
1237   ['olive oil', 'garlic', 'chicken breasts', 'ch...
41671  ['garlic', 'sun-dried tomatoes chopped', 'oliv...
```

In [85]:
```python
# Raporlama için Özet
"""
Rapor Bölümleri:
1. Giriş: Veri seti Allrecipes.com'dan alınmış tarif verileridir. Amaç, malzeme bazlı tarif eşleştirme sistemidir.
2. Data Scraping: recipe_final (1).csv dosyası proje için sağlandı.
3. Ön İşleme: NLTK ile tokenizasyon, stopword kaldırma, lowercasing, lemmatizasyon ve stemming yapıldı.
4. Temizlenmiş Veri: cleaned_lemmatized.csv ve cleaned_stemmed.csv oluşturuldu.
5. Vektörleştirme:
  - TF-IDF: tfidf_lemmatized.csv ve tfidf_stemmed.csv
  - Word2Vec: 16 model eğitildi, örnek benzerlik sonuçları paylaşıldı.
6. Sonuç: Sistem, kullanıcı malzemelerine göre uygun tarifleri başarıyla öneriyor. Skip-gram ve lemmatized modeller daha iyi sonuçlar verebilir.
"""
```

```
Out[85]: "\nRapor Bölümleri:\n1. Giriş: Veri seti Allrecipes.com'dan alınmış tarif verileridir. Amaç, malzeme bazlı tarif eşleştirme sistemidir.\n2. Data Scraping: recipe_final (1).csv dosyası proje
için sağlandı.\n3. Ön İşleme: NLTK ile tokenizasyon, stopword kaldırma, lowercasing, lemmatizasyon ve stemming yapıldı.\n4. Temizlenmiş Veri: cleaned_lemmatized.csv ve cleaned_stemmed.csv o
luşturuldu.\n5. Vektörleştirme:\n   - TF-IDF: tfidf_lemmatized.csv ve tfidf_stemmed.csv\n   - Word2Vec: 16 model eğitildi, örnek benzerlik sonuçları paylaşıldı.\n6. Sonuç: Sistem, kullanıcı
malzemelerine göre uygun tarifleri başarıyla öneriyor. Skip-gram ve lemmatized modeller daha iyi sonuçlar verebilir.\n"
```

In [ ]:

In [ ]: