DOĞAL DİL İŞLEME DERSİ – ÖDEV 1 RAPORU

Doğal Dil İşleme – Ödev 1
Metin Tabanlı Veri Setleri ile Yapay Zekâ Modelleri Geliştirme
Ad & Soyad: Ömer Faruk AĞTOPRAK
Öğrenci No: 2107231050

Teslim Tarihi: 4 Mayıs 2025

Github Link: https://github.com/omeeragtoprak/NLP

1. Giriş

- **Proje Amacı:**

Bu ödevde, seçilen metin tabanlı bir veri seti üzerinde doğal dil işleme (NLP) teknikleriyle ön işleme, vektörleştirme ve benzerlik analizi adımları uygulanmıştır. Amaç, metinlerin makine öğrenmesi modelleriyle sayısal olarak temsil edilmesi ve benzerliklerinin ölçülmesidir.

2. Data Scraping ve Ham Veri Detayları

- **Kullanılan Veri Seti:**

Kaynak: https://github.com/scrosseye/AI-Detection-Student-Writing

Boyut: 1 doküman, CSV formatında, 22,1 MB boyutunda

Her bir satır bir ödev metnidir.

Örnek:

text,split,model,strategy,use_source_text,prompt_id,generated

In conclusion, limiting car usage has many advantages even if you dont realize it and our soiciety could benefit a lot from it. Like, when it comes to the safety of the next generation, the present, or the previous. Than, it can even take away stress that come with a vechicle and even increase health and social gathering.,train,student,n/a,true,0,0

Alanlar ve Açıklamaları

Sütun Adı	Açıklama
text	Öğrenci tarafından yazılmış ya da yapay zekâ tarafından üretilmiş metin. Bu örnekteki metin, bireysel araç kullanımının sınırlandırılmasının toplumsal faydaları hakkında yazılmıştır.
split	Verinin hangi amaçla ayrıldığını belirtir. Örnekte "train" olarak işaretlenmiştir, yani bu veri model eğitimi (training) için kullanılacaktır.
model	Metnin kimin tarafından üretildiğini belirtir. "student" değeri, bu metnin bir öğrenci tarafından yazıldığını gösterir. Yapay zekâ tarafından üretilen metinlerde "gpt" veya "ai" gibi ibareler yer alır.
strategy	Verinin üretim süreciyle ilgili strateji ya da etiket. Örnekte "n/a" yani geçerli bir strateji kullanılmamış ya da belirtilmemiş.

Sütun Adı	Açıklama
use_source_text	"true" değeri, metnin bir kaynak metne dayalı olarak üretildiğini ifade eder. Bu, genellikle "prompt" (verilen konu cümlesi ya da soru) temelinde oluşturulmuş metinlerde bulunur.
prompt_id	Verilen metnin dayandığı prompt'un (konu cümlesi) kimliğidir. 0 değeri, ilk prompta karşılık geldiğini gösterir.
generated	"0" değeri, metnin yapay zekâ tarafından değil, bir insan (öğrenci) tarafından yazıldığını belirtir. "1" olsaydı, bu metin yapay zekâ tarafından üretilmiş olurdu.

Metin İçeriği Açıklaması

"In conclusion, limiting car usage has many advantages even if you dont realize it and our soiciety could benefit a lot from it. Like, when it comes to the safety of the next generation, the present, or the previous. Than, it can even take away stress that come with a vechicle and even increase health and social gathering."

Bu metin; araç kullanımının kısıtlanmasının, güvenlik, stres azaltma, sağlık ve sosyal yaşam açısından olumlu etkileri olduğunu vurgulayan bir kompozisyon paragrafıdır. Metin, dil bilgisi ve yazım açısından bazı hatalar içerse de, gerçek bir öğrenci yazısı olduğu izlenimi vermektedir. Örneğin:

- "dont" → "don't"
- "soiciety" → "society"
- "vechicle" → "vehicle"
- "Than" → muhtemelen "Then" olmalı.

Bu hatalar, metnin gerçek bir kullanıcı tarafından yazıldığını ve bu nedenle dil işleme sürecinde gürültü (noise) barındırdığını göstermektedir. Bu tarz metinler, doğal dil işleme sistemlerinin gerçek dünya verisiyle nasıl başa çıkabileceğini test etmek açısından önemlidir.

3. Zipf Yasası Analizi

- **Yöntem:**

Ham veri, lemmatized ve stemmed veri için kelime frekansları hesaplanıp, log-log eksende Zipf grafiği çizildi.

- **Kod:**

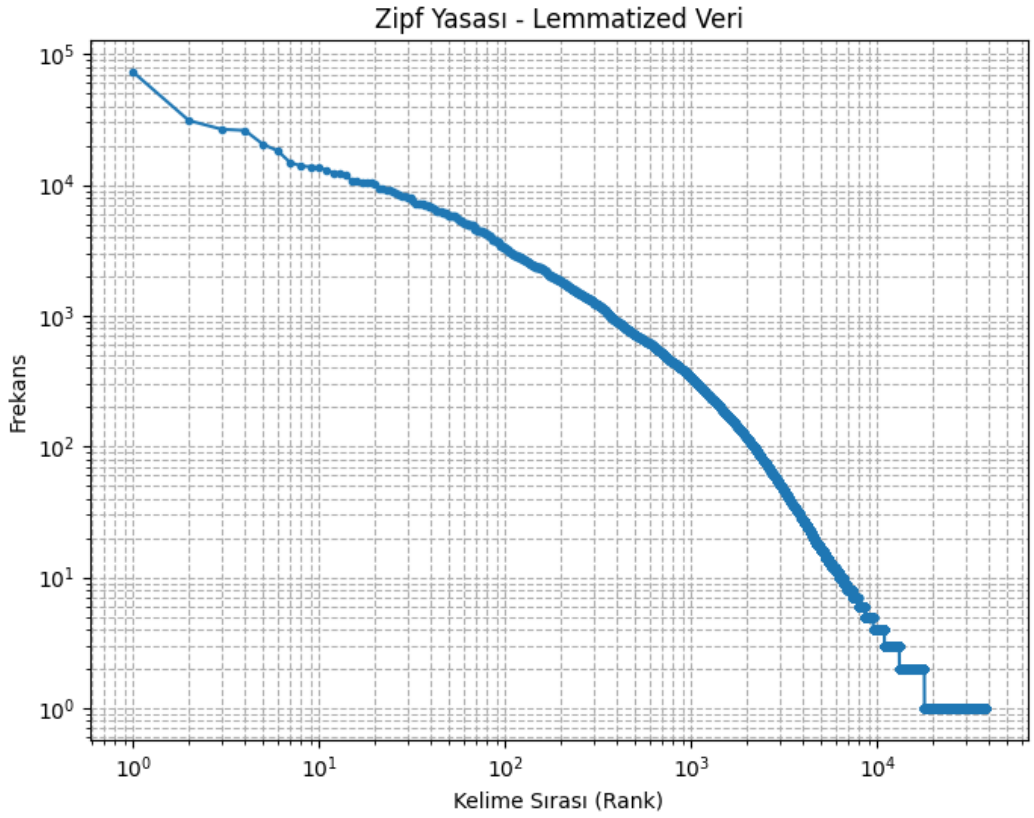
In [ ]:

```python
# zipf_analysis
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from collections import Counter

DATA_FILES = [
    ("ai_detect_essays_filtered.csv", "zipf_raw.png", "Ham Veri"),
    ("essays_lemmatized.csv", "zipf_lemmatized.png", "Lemmatized Veri"),
    ("essays_stemmed.csv", "zipf_stemmed.png", "Stemmed Veri")
]

def plot_zipf(csv_file, out_file, title):
    df = pd.read_csv(csv_file, header=None)
    text = " ".join(df[0].astype(str))
    words = text.split()
    counts = Counter(words)
    freqs = np.array(sorted(counts.values(), reverse=True))
    ranks = np.arange(1, len(freqs)+1)
    plt.figure(figsize=(8,6))
    plt.loglog(ranks, freqs, marker=".")
    plt.title(f"Zipf Yasası - {title}")
    plt.xlabel("Kelime Sırası (Rank)")
    plt.ylabel("Frekans")
    plt.grid(True, which="both", ls="--")
    plt.savefig(out_file)
    print(f"Zipf grafiği '{out_file}' olarak kaydedildi.")

if __name__ == "__main__":
    for csv_file, out_file, title in DATA_FILES:
        plot_zipf(csv_file, out_file, title)
```

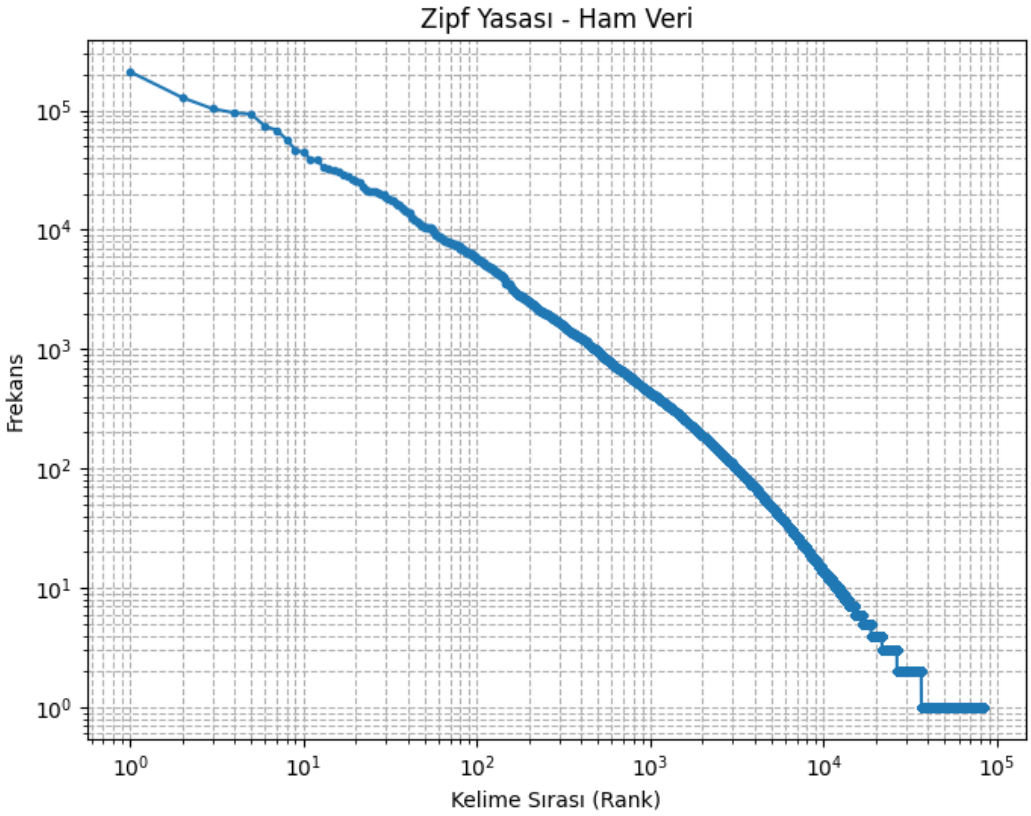
```
Zipf grafiği 'zipf_raw.png' olarak kaydedildi.
Zipf grafiği 'zipf_lemmatized.png' olarak kaydedildi.
Zipf grafiği 'zipf_stemmed.png' olarak kaydedildi.
```

- **Grafikler:**

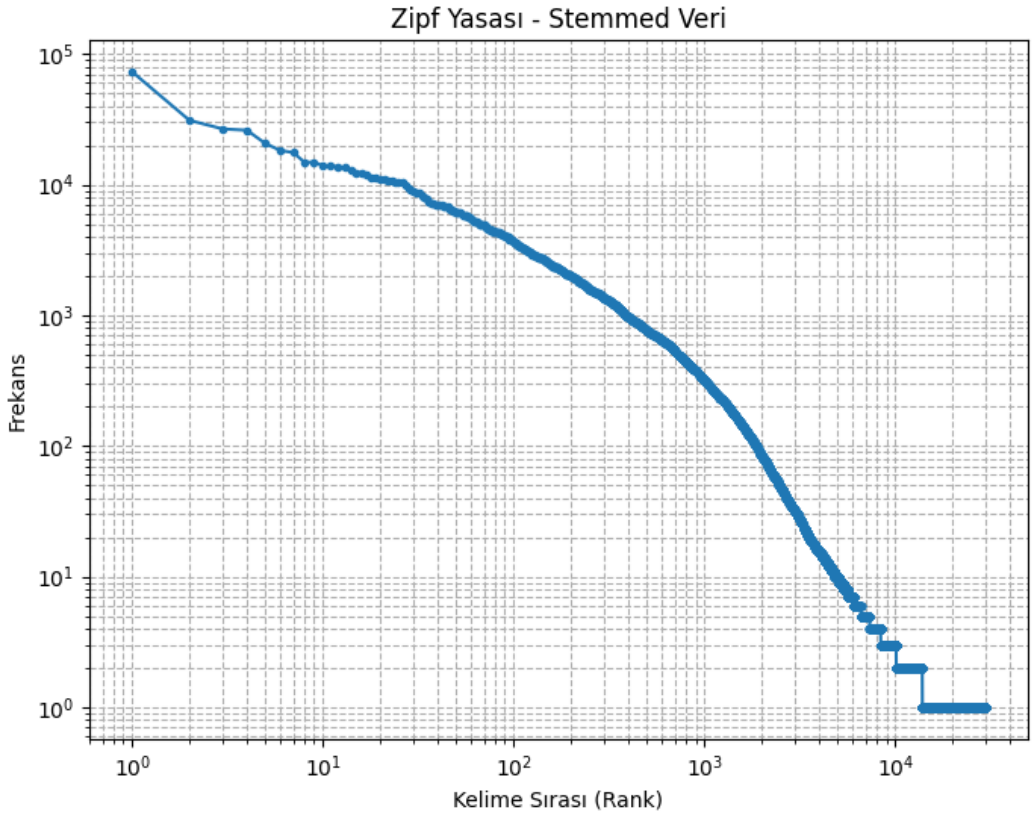
3.1



3.2


Zipf Yasası - Ham Veri
Frekans
Kelime Sırası (Rank)
10^0
10^1
10^2
10^3
10^4
10^5

3.3



3.3

- **Yorum:**

Zipf yasası, doğal dilde kelime frekanslarının log-log eksende yaklaşık doğrusal bir dağılım gösterdiğini belirtir. Grafikte bu doğrusal ilişki gözlemlenmiştir. Veri seti boyutu, istatistiksel analiz için yeterlidir.

4. Ön İşleme (Pre-processing) Aşamaları

4.1. Stop Word Removal

- **Kullanılan Araç:** nltk.corpus.stopwords

4.2. Tokenization

- **Kullanılan Araç:** nltk.tokenize.word_tokenize, sent_tokenize

4.3. Lowercasing

- **Kullanılan Araç:** str.lower(), token.lower()

4.4. Lemmatization

- **Kullanılan Araç:** nltk.stem.WordNetLemmatizer

4.5. Stemming

- **Kullanılan Araç:** nltk.stem.PorterStemmer

4.6. Özel Karakter ve HTML Temizliği

- **Kullanılan Araç:** re.sub(), pattern olarak kullanılan düzenli ifade: r'<.*?>'

```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

def clean_text(text):
    # Lowercase
    text = text.lower()
    # Remove HTML tags
    text = re.sub(r'<.*?>', '', text)
    # Remove special characters and digits
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    return text

def preprocess_text(text, method='lemmatize'):
    stop_words = set([w.lower() for w in stopwords.words('english')])
    lemmatizer = WordNetLemmatizer()
    stemmer = PorterStemmer()
    sentences = sent_tokenize(str(text))
    processed_sentences = []
    for sentence in sentences:
        tokens = word_tokenize(sentence)
        filtered_tokens = [token.lower() for token in tokens if token.isalpha() and token.lower() not in stop_words]
        if method == 'lemmatize':
            processed_tokens = [lemmatizer.lemmatize(token) for token in filtered_tokens]
        elif method == 'stem':
            processed_tokens = [stemmer.stem(token) for token in filtered_tokens]
        else:
            processed_tokens = filtered_tokens
        processed_sentences.append(processed_tokens)
    return processed_sentences
```

```
def clean_text(text):
    return re.sub(r'[^a-zA-Z\s]', '', re.sub(r'<.*?>', '', text.lower()))

def show_preprocessing_steps(text):
    stop_words = set(stopwords.words('english'))
    lemmatizer = WordNetLemmatizer()
    stemmer = PorterStemmer()

    # İlk cümleyi al
    first_sentence = sent_tokenize(text)[0]
    print("\n🔹 Orjinal Cümle (İlk 10 Kelime):", ' '.join(first_sentence.split()[:10]))

    # İlk 10 kelimeyi al
    tokens = word_tokenize(first_sentence)
    first_10_tokens = tokens[:10]
    print("\n🔹 İlk 10 kelime (Ham):", first_10_tokens)

    print("\n--- Her kelime için işleme adımları ---")
    for token in first_10_tokens:
        lower = token.lower()
        is_alpha = lower.isalpha()
        stop_removed = lower not in stop_words if is_alpha else False
        lemma = lemmatizer.lemmatize(lower) if stop_removed else "-"
        stem = stemmer.stem(lower) if stop_removed else "-"

        print(f"\nKelime: '{token}'")
        print(f" - Lowercased        : '{lower}'")
        print(f" - Alphabetic?       : {is_alpha}")
        print(f" - Stop word?        : {'Hayır' if stop_removed else 'Evet'}")
        print(f" - Lemmatized        : '{lemma}'")
        print(f" - Stemmed           : '{stem}'")

def main():
    df = pd.read_csv("ai_detect_essays_filtered.csv")
```

```
Orjinal Cümle (İlk 10 Kelime): a life filled to the brim of better days is

İlk 10 kelime (Ham): ['a', 'life', 'filled', 'to', 'the', 'brim', 'of', 'better', 'days', 'is']

--- Her kelime için işleme adımları ---

Kelime: 'a'
  - Lowercased        : 'a'
  - Alphabetic?       : True
  - Stop word?        : Evet
  - Lemmatized        : '-'
  - Stemmed           : '-'

Kelime: 'life'
  - Lowercased        : 'life'
  - Alphabetic?       : True
  - Stop word?        : Hayır
  - Lemmatized        : 'life'
  - Stemmed           : 'life'

Kelime: 'filled'
  - Lowercased        : 'filled'
  - Alphabetic?       : True
  - Stop word?        : Hayır
  - Lemmatized        : 'filled'
  - Stemmed           : 'fill'

Kelime: 'to'
  - Lowercased        : 'to'
  - Alphabetic?       : True
  - Stop word?        : Evet
  - Lemmatized        : '-'
  - Stemmed           : '-'
```

```
Kelime: 'the'
  - Lowercased        : 'the'
  - Alphabetic?       : True
  - Stop word?        : Evet
  - Lemmatized        : '-'
  - Stemmed           : '-'

Kelime: 'brim'
  - Lowercased        : 'brim'
  - Alphabetic?       : True
  - Stop word?        : Hayır
  - Lemmatized        : 'brim'
  - Stemmed           : 'brim'

Kelime: 'of'
  - Lowercased        : 'of'
  - Alphabetic?       : True
  - Stop word?        : Evet
  - Lemmatized        : '-'
  - Stemmed           : '-'

Kelime: 'better'
  - Lowercased        : 'better'
  - Alphabetic?       : True
  - Stop word?        : Hayır
  - Lemmatized        : 'better'
  - Stemmed           : 'better'

Kelime: 'days'
  - Lowercased        : 'days'
  - Alphabetic?       : True
  - Stop word?        : Hayır
  - Lemmatized        : 'day'
  - Stemmed           : 'day'
```

5. Temizlenmiş Veri Seti Çıktısı

- **Çıktı Dosyaları:**
- essays_lemmatized.csv
- essays_stemmed.csv

```
Preprocessing tamamlandı.

Örnek ham veri:
A life filled to the brim of better days is what we all want, and limiting car usage has some of these advantages. "When I had a car I was always tense. I'm much

"Mr. Sivak's son lives in San Francisco and has a car but takes Bay Area Rapid Transit, when he can, even though that often takes longer than driving." Clearly b

"In this new approach, stores are placed a walk away, on a main street, rather than in malls along some distant highway." Not one person can really enjoy a twent

Pollution is a horrible topic that everyone wants to stop, however everyone is ignoring one of its sources. "After days of nearrecord pollution, Paris enforced a

Happiness, fast travel, no pollution, and more new stuff sounds like owning a car right? Carlos Arturo has a taste of the sweet life, "It's a good opportunity to

Temizlenmiş:
a life filled to the brim of better days is what we all want and limiting car usage has some of these advantages when i had a car i was always tense im much happ

mr sivaks son lives in san francisco and has a car but takes bay area rapid transit when he can even though that often takes longer than driving clearly being la

in this new approach stores are placed a walk away on a main street rather than in malls along some distant highway not one person can really enjoy a twenty or f

pollution is a horrible topic that everyone wants to stop however everyone is ignoring one of its sources after days of nearrecord pollution paris enforced a par

happiness fast travel no pollution and more new stuff sounds like owning a car right carlos arturo has a taste of the sweet life its a good opportunity to take a

Lemmatized:
[['life', 'filled', 'brim', 'better', 'day', 'want', 'limiting', 'car', 'usage', 'advantage', 'car', 'always', 'tense', 'im', 'much', 'happier', 'way', 'said', '

Stemmed:
[['life', 'fill', 'brim', 'better', 'day', 'want', 'limit', 'car', 'usag', 'advantag', 'car', 'alway', 'tens', 'im', 'much', 'happier', 'way', 'said', 'heidrun',
```

- **Zipf Grafikleri:**

3.maddede eklenen Zipf grafikleri referanstır. (3.1, 3.2, 3.3)

- **Yorum:**

```python
# Dosyaları oku
df_raw = pd.read_csv("ai_detect_essays_filtered.csv")
df_lemma = pd.read_csv("essays_lemmatized.csv", header=None)
df_stem = pd.read_csv("essays_stemmed.csv", header=None)

# Toplam kelime sayısını hesaplayan fonksiyon
def count_total_words(df, col=0):
    return sum(len(word_tokenize(str(row))) for row in df[col])

# Toplam kelime sayıları
raw_token_count = count_total_words(df_raw, col='text')
lemma_token_count = count_total_words(df_lemma)
stem_token_count = count_total_words(df_stem)

# Yüzdesel fark fonksiyonu
def percent_diff(original, new):
    return round(((original - new) / original) * 100, 2)

# Çıktılar
print("\n📊 CSV DOSYALARI KARŞILAŞTIRMASI")
print(f"📌 Ham veri (orijinal):      {raw_token_count} kelime")
print(f"📌 Lemmatized veri:          {lemma_token_count} kelime  (-{percent_diff(raw_token_count, lemma_token_count)}%)")
print(f"📌 Stemmed veri:             {stem_token_count} kelime  (-{percent_diff(raw_token_count, stem_token_count)}%)")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!

📊 CSV DOSYALARI KARŞILAŞTIRMASI
📌 Ham veri (orijinal):      4570003 kelime
📌 Lemmatized veri:          2069561 kelime  (-54.71%)
📌 Stemmed veri:             2069561 kelime  (-54.71%)
```

Temizleme sonrası kelime çeşitliliği azalmış, veri boyutu küçülmüştür. Bu, modelleme için daha homojen bir veri sağlar.

essays_lemmatized.csv 13,3 MB

essays_stemmed.csv 11,9 MB

Olarak güncellenmiştir.

6. Vektörleştirme (Vectorization)

6.1. TF-IDF Vektörleştirme

- **Kullanılan Araç: sklearn.feature_extraction.text.TfidfVectorizer**
- **Kod:**

```python
# tfidf_vectorization
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer

def save_tfidf_matrix(input_file, output_file):
    df = pd.read_csv(input_file, header=None)
    texts = df[0].astype(str).tolist()
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(texts)
    tfidf_df = pd.DataFrame(tfidf_matrix.toarray(), columns=vectorizer.get_feature_names_out())
    tfidf_df.to_csv(output_file, index=False)
    print(f"TF-IDF matrisi '{output_file}' olarak kaydedildi.")

if __name__ == "__main__":
    save_tfidf_matrix("essays_lemmatized.csv", "tfidf_lemmatized.csv")
    save_tfidf_matrix("essays_stemmed.csv", "tfidf_stemmed.csv")
```

```
TF-IDF matrisi 'tfidf_lemmatized.csv' olarak kaydedildi.
TF-IDF matrisi 'tfidf_stemmed.csv' olarak kaydedildi.
```

- **Çıktı Dosyaları:**

  tfidf_lemmatized.csv, tfidf_stemmed.csv dosyaları 1GB'dan büyük olduğu için drive linki paylaşılmıştır (boyuttan dolayı github'a yüklenememiştir.):
  https://drive.google.com/drive/folders/10vPwsgcacFMYNZz6FNohfnyNWizW1dQL

6.2. Word2Vec Vektörleştirme

- **Kullanılan Araç:** gensim.models.Word2Vec
- **Parametreler:**

8 farklı parametre seti (cbow/skipgram, window=2/4, dim=100/300)

- **Model Üretim Kodları ve Çıktısı:**

https://drive.google.com/drive/folders/1KGV3o6-FJ1Q9iJWmRpHdOfZo98AqdTLD?usp=sharing

```python
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 load_sentences(file_path):
    sentences = []
    with open(file_path, encoding="utf-8") as f:
        reader = csv.reader(f)
        for row in reader:
            if row:
                sentences.append(row)
    return sentences

def print_similar_words(model, word, topn=5):
    try:
        if word in model.wv:
            similar = model.wv.most_similar(word, topn=topn)
            print(f"\nModel: {model} - '{word}' ile En Benzer {topn} Kelime:")
            for w, score in similar:
                print(f"Kelime: {w}, Benzerlik Skoru: {score:.4f}")
        else:
            print(f"Kelime '{word}' modelde yok.")
    except Exception as e:
        print(f"Benzer kelime bulunamadı: {e}")

def train_and_save_models():
    for corpus_type in ['lemmatized', 'stemmed']:
        sentences = load_sentences(f"sentences_{corpus_type}.csv")
        for param in PARAMETERS:
            model = Word2Vec(
                sentences,
                vector_size=param['vector_size'],
                window=param['window'],
                min_count=1,
                sg=1 if param['model_type'] == 'skipgram' else 0
            )
            model_name = f"{corpus_type}_model_{param['model_type']}_window{param['window']}_dim{param['vector_size']}.model"
            model.save(model_name)
            print(f"{model_name} model saved!")
            print_similar_words(model, 'car', topn=5)
```

```python
from gensim.models import Word2Vec
import glob

MODEL_PATTERNS = [
    "lemmatized_model_*.model",
    "stemmed_model_*.model"
]

def print_similar_words(model_path, word, topn=5):
    try:
        model = Word2Vec.load(model_path)
        if word in model.wv:
            similar = model.wv.most_similar(word, topn=topn)
            print(f"\nModel: {model_path} - '{word}' ile En Benzer {topn} Kelime:")
            for w, score in similar:
                print(f"Kelime: {w}, Benzerlik Skoru: {score:.4f}")
        else:
            print(f"Kelime '{word}' modelde yok. (Model: {model_path})")
    except Exception as e:
        print(f"Model yüklenemedi veya hata oluştu: {model_path} - {e}")

if __name__ == "__main__":
    word = "car"
    for pattern in MODEL_PATTERNS:
        for model_path in glob.glob(pattern):
            print_similar_words(model_path, word, topn=5)
```

```
Model: lemmatized_model_cbow_window2_dim100.model - 'car' ile En Benzer 5 Kelime:
Kelime: vehicle, Benzerlik Skoru: 0.7908
Kelime: completly, Benzerlik Skoru: 0.6557
Kelime: automobile, Benzerlik Skoru: 0.6398
Kelime: completely, Benzerlik Skoru: 0.6297
Kelime: therefore, Benzerlik Skoru: 0.5734

Model: lemmatized_model_cbow_window4_dim100.model - 'car' ile En Benzer 5 Kelime:
Kelime: vehicle, Benzerlik Skoru: 0.7767
Kelime: automobile, Benzerlik Skoru: 0.6040
Kelime: completely, Benzerlik Skoru: 0.5885
Kelime: lastly, Benzerlik Skoru: 0.5663
Kelime: fully, Benzerlik Skoru: 0.5592

Model: lemmatized_model_cbow_window4_dim300.model - 'car' ile En Benzer 5 Kelime:
Kelime: vehicle, Benzerlik Skoru: 0.7470
Kelime: automobile, Benzerlik Skoru: 0.5952
Kelime: fully, Benzerlik Skoru: 0.5828
Kelime: completely, Benzerlik Skoru: 0.5521
Kelime: lastly, Benzerlik Skoru: 0.5143

Model: lemmatized_model_skipgram_window2_dim100.model - 'car' ile En Benzer 5 Kelime:
Kelime: vehicle, Benzerlik Skoru: 0.8334
Kelime: driverless, Benzerlik Skoru: 0.8029
Kelime: automobile, Benzerlik Skoru: 0.7963
Kelime: completly, Benzerlik Skoru: 0.7654
Kelime: carsthe, Benzerlik Skoru: 0.7353

Model: lemmatized_model_skipgram_window4_dim100.model - 'car' ile En Benzer 5 Kelime:
Kelime: driverless, Benzerlik Skoru: 0.8414
Kelime: vehicle, Benzerlik Skoru: 0.8248
Kelime: automobile, Benzerlik Skoru: 0.7672
Kelime: manufactured, Benzerlik Skoru: 0.7641
Kelime: completley, Benzerlik Skoru: 0.7634

Model: lemmatized_model_cbow_window2_dim300.model - 'car' ile En Benzer 5 Kelime:
Kelime: vehicle, Benzerlik Skoru: 0.7518
Kelime: automobile, Benzerlik Skoru: 0.6757
Kelime: usagecitizens, Benzerlik Skoru: 0.6078
Kelime: completly, Benzerlik Skoru: 0.5935
Kelime: lastly, Benzerlik Skoru: 0.5914

Model: lemmatized_model_skipgram_window2_dim300.model - 'car' ile En Benzer 5 Kelime:
Kelime: driverless, Benzerlik Skoru: 0.7525
Kelime: automobile, Benzerlik Skoru: 0.7331
Kelime: vehicle, Benzerlik Skoru: 0.7119
Kelime: completly, Benzerlik Skoru: 0.6963
Kelime: anyways, Benzerlik Skoru: 0.6900
```

```
Model: lemmatized_model_skipgram_window4_dim300.model - 'car' ile En Benzer 5 Kelime:
Kelime: driverless, Benzerlik Skoru: 0.7784
Kelime: driveless, Benzerlik Skoru: 0.7122
Kelime: entail, Benzerlik Skoru: 0.6978
Kelime: stil, Benzerlik Skoru: 0.6961
Kelime: manufactured, Benzerlik Skoru: 0.6953

Model: stemmed_model_cbow_window4_dim100.model - 'car' ile En Benzer 5 Kelime:
Kelime: vehicl, Benzerlik Skoru: 0.7758
Kelime: automobil, Benzerlik Skoru: 0.6115
Kelime: fulli, Benzerlik Skoru: 0.5575
Kelime: anyway, Benzerlik Skoru: 0.5526
Kelime: complet, Benzerlik Skoru: 0.5449

Model: stemmed_model_skipgram_window2_dim100.model - 'car' ile En Benzer 5 Kelime:
Kelime: vehicl, Benzerlik Skoru: 0.8357
Kelime: driverless, Benzerlik Skoru: 0.7947
Kelime: automobil, Benzerlik Skoru: 0.7909
Kelime: vehic, Benzerlik Skoru: 0.7753
Kelime: completli, Benzerlik Skoru: 0.7666

Model: stemmed_model_cbow_window4_dim300.model - 'car' ile En Benzer 5 Kelime:
Kelime: vehicl, Benzerlik Skoru: 0.7456
Kelime: automobil, Benzerlik Skoru: 0.5783
Kelime: complet, Benzerlik Skoru: 0.5708
Kelime: fulli, Benzerlik Skoru: 0.5619
Kelime: secondli, Benzerlik Skoru: 0.5517

Model: stemmed_model_skipgram_window4_dim300.model - 'car' ile En Benzer 5 Kelime:
Kelime: driverless, Benzerlik Skoru: 0.7703
Kelime: drivabl, Benzerlik Skoru: 0.7204
Kelime: vehicl, Benzerlik Skoru: 0.7188
Kelime: ourself, Benzerlik Skoru: 0.7148
Kelime: completley, Benzerlik Skoru: 0.7104

Model: stemmed_model_skipgram_window2_dim300.model - 'car' ile En Benzer 5 Kelime:
Kelime: vehic, Benzerlik Skoru: 0.7567
Kelime: driverless, Benzerlik Skoru: 0.7482
Kelime: automobil, Benzerlik Skoru: 0.7358
Kelime: antoh, Benzerlik Skoru: 0.7155
Kelime: non, Benzerlik Skoru: 0.7150

Model: stemmed_model_cbow_window2_dim300.model - 'car' ile En Benzer 5 Kelime:
Kelime: vehicl, Benzerlik Skoru: 0.7303
Kelime: automobil, Benzerlik Skoru: 0.6624
Kelime: completli, Benzerlik Skoru: 0.6108
Kelime: fulli, Benzerlik Skoru: 0.5883
Kelime: total, Benzerlik Skoru: 0.5721
```

```
Model: stemmed_model_cbow_window2_dim100.model - 'car' ile En Benzer 5 Kelime:
Kelime: vehicl, Benzerlik Skoru: 0.7895
Kelime: automobil, Benzerlik Skoru: 0.6711
Kelime: completli, Benzerlik Skoru: 0.6093
Kelime: lastli, Benzerlik Skoru: 0.5847
Kelime: secondli, Benzerlik Skoru: 0.5804

Model: stemmed_model_skipgram_window4_dim100.model - 'car' ile En Benzer 5 Kelime:
Kelime: vehicl, Benzerlik Skoru: 0.8396
Kelime: driverless, Benzerlik Skoru: 0.8312
Kelime: non, Benzerlik Skoru: 0.7943
Kelime: ourself, Benzerlik Skoru: 0.7779
Kelime: convien, Benzerlik Skoru: 0.7747
```

- **Yorum:**

Model Türlerinin Temel Özellikleri

1. CBOW (Continuous Bag of Words)

- Çevredeki kelimelerden hedef kelimeyi tahmin eder.
- **Daha hızlı** çalışır.
- Küçük veri setlerinde **daha iyi** sonuç verir.
- Daha **genelleştirici** bir yapıya sahiptir.

2. Skip-gram

- Hedef kelimeden çevresindeki kelimeleri tahmin eder.

- **Daha yavaş** ama **daha güçlü bağlamsal** ilişki yakalama potansiyeli vardır.
- **Daha büyük veri setlerinde daha başarılı**dır.
- Nadir kelimeler için **daha iyi temsil** üretir.

Parametrelerin Etkisi

🔹 **Pencere Boyutu (Window Size)**

- **Düşük pencere (2)**: Daha **yerel** bağlam yakalanır (yakın kelimelerle ilişkiler).
- **Yüksek pencere (4)**: Daha **geniş bağlam** yakalanır (daha anlamsal ilişkiler).

🔹 **Vektör Boyutu (Dimension)**

- **100 boyut**: Daha hızlı ve basit modeller.
- **300 boyut**: Daha fazla bağlamsal bilgi, ama daha fazla eğitim verisi ve zaman gerekir.

🔹 **Lemmatization vs Stemming**

- **Lemmatization**: Anlam korunduğu için daha doğal ve anlamlı vektörler üretme olasılığı yüksek.
- **Stemming**: Daha agresif kısaltma; kelime kökleri bozulabilir (örneğin vehicl, completli gibi yapay terimler oluşmuş).

Model Karşılaştırması ve Beklenen Başarı Durumu

◆ **En Tutarlı ve Anlamlı Sonuç Veren Modeller**

- lemmatized_model_skipgram_window2_dim100.model
    - Örnek benzer kelimeler: vehicle, driverless, automobile, completly
    - **Skorlar yüksek ve semantik açıdan mantıklı**.
- lemmatized_model_skipgram_window4_dim100.model
    - Daha geniş bağlamla manufactured, driverless gibi mantıklı ilişkiler bulunmuş.

◆ **Stemming Modellerinde**

- Genelde benzer kelimeler mantıklı, ama kelimelerin kökleri yapay (vehicl, automobil, completli) ve **anlamsal bütünlük azalmış**.

Hangi Modelin Daha Başarılı Olması Beklenir? Neden?

Tahmini En Başarılı Model:

lemmatized_model_skipgram_window2_dim100.model

Neden?

- Skip-gram modeli sayesinde car gibi hedef kelimeden anlamca yakın olan driverless, vehicle, automobile gibi **yüksek benzerlik skorlarıyla bağlamsal güçlü ilişkiler** kurulmuş.
- Lemmatization uygulanmış olması, anlamlı kelimelerin korunmasını sağlamış (vehicle yerine vehicl gibi bozuk kökler yok).
- Pencere boyutunun 2 olması, daha yerel bağlamlara odaklanarak **doğrudan ilişkili kelimeler** yakalanmış.
- Vektör boyutunun 100 olması, **daha az gürültü** ve **daha hızlı eğitim** sağlamış.

Sonuç

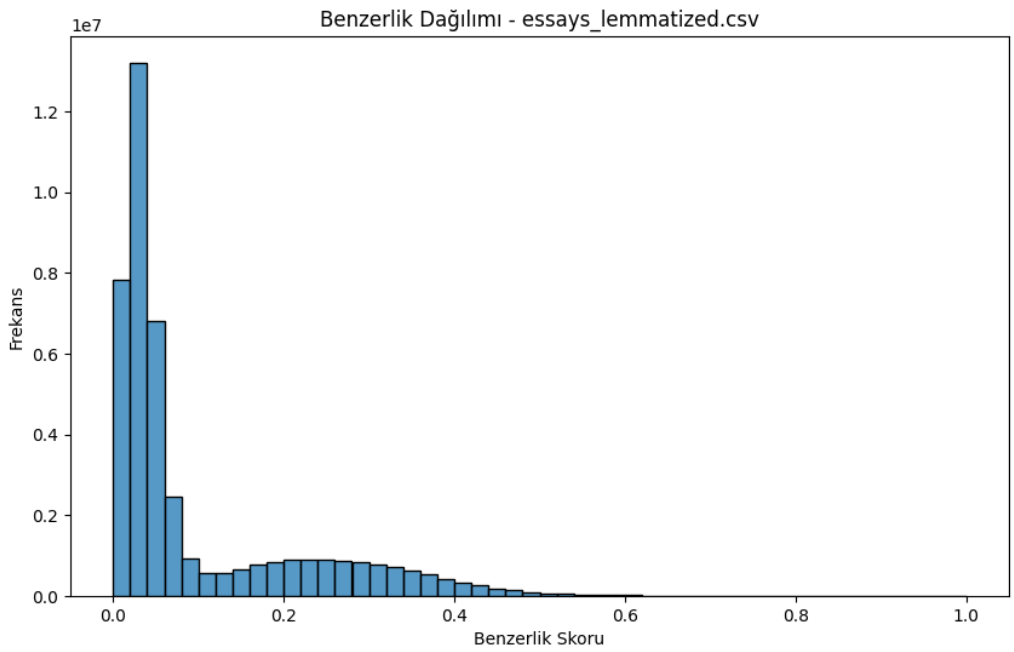
Model Türü	**Temizlik Yöntemi**	**Başarı Potansiyeli**	**Gözlem**
Skip-gram	Lemmatization	Yüksek	Anlamlı, güçlü ilişkiler
CBOW	Lemmatization	Orta	Genelleştirici ama bağlam zayıf
Skip-gram	Stemming	Orta	Kök bozulmaları semantiği zedeliyor
CBOW	Stemming	Düşük	Hem bağlam hem semantik zayıf

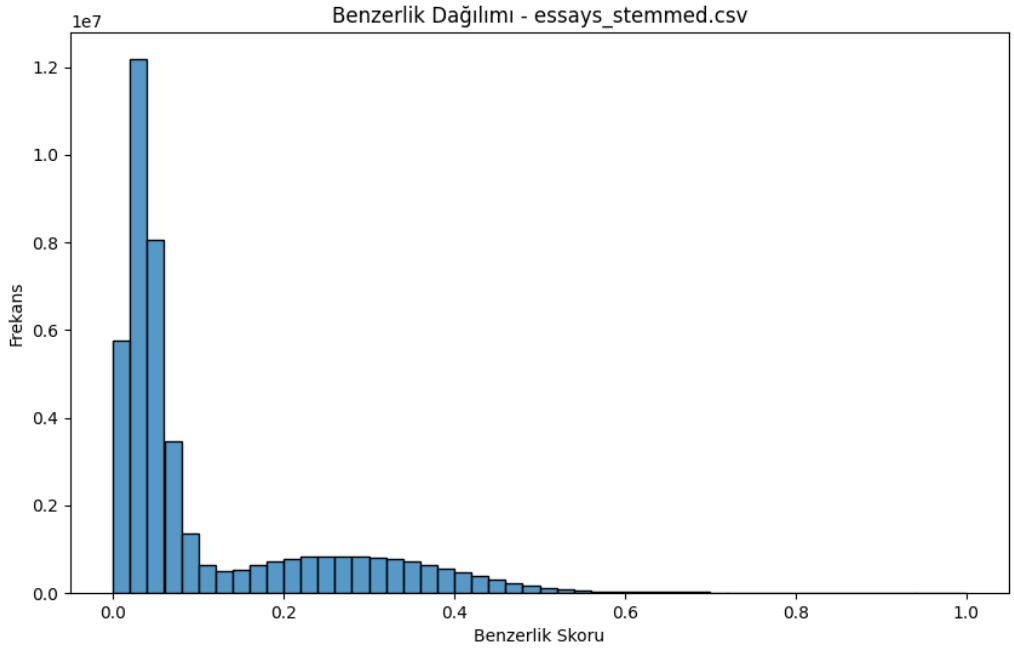
7. Benzerlik Analizi ve Raporlama

- **Cosine Similarity Analizi:**

```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

def compute_and_plot_similarity(input_file, out_img, threshold=0.3):
    df = pd.read_csv(input_file, header=None)
    texts = df[0].astype(str).tolist()
    if not any(texts):
        print(f"Uyarı: {input_file} boş veya geçersiz.")
        return
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(texts)
    if tfidf_matrix.shape[1] == 0:
        print(f"Uyarı: {input_file} için TF-IDF matrisinde hiç kelime yok.")
        return
    similarity_matrix = cosine_similarity(tfidf_matrix)
    if np.all(similarity_matrix == 0):
        print(f"Uyarı: {input_file} için tüm benzerlikler sıfır.")
        return
    plt.figure(figsize=(10, 6))
    sns.histplot(similarity_matrix[np.triu_indices_from(similarity_matrix, k=1)], bins=50)
    plt.title(f'Benzerlik Dağılımı - {input_file}')
    plt.xlabel('Benzerlik Skoru')
    plt.ylabel('Frekans')
    plt.savefig(out_img)
    print(f"Benzerlik dağılımı '{out_img}' olarak kaydedildi.")
    high_similarity = np.sum(similarity_matrix > threshold) / 2
    total_pairs = similarity_matrix.shape[0] * (similarity_matrix.shape[0] - 1) / 2
    print(f"Toplam çift sayısı: {total_pairs}")
    print(f"%{threshold*100}'dan yüksek benzerlik gösteren çift sayısı: {high_similarity}")
    print(f"Yüksek benzerlik oranı: {high_similarity/total_pairs:.2%}")
    print(f"Ortalama benzerlik: {np.mean(similarity_matrix[np.triu_indices_from(similarity_matrix, k=1)]):.4f}")
    print(f"Medyan benzerlik: {np.median(similarity_matrix[np.triu_indices_from(similarity_matrix, k=1)]):.4f}")

if __name__ == "__main__":
    compute_and_plot_similarity("essays_lemmatized.csv", "similarity_distribution_lemmatized.png")
    compute_and_plot_similarity("essays_stemmed.csv", "similarity_distribution_stemmed.png")
```

```
Benzerlik dağılımı 'similarity_distribution_lemmatized.png' olarak kaydedildi.
Toplam çift sayısı: 43426540.0
%30.0'dan yüksek benzerlik gösteren çift sayısı: 4359105.0
Yüksek benzerlik oranı: 10.04%
Ortalama benzerlik: 0.1003
Medyan benzerlik: 0.0414
Benzerlik dağılımı 'similarity_distribution_stemmed.png' olarak kaydedildi.
Toplam çift sayısı: 43426540.0
%30.0'dan yüksek benzerlik gösteren çift sayısı: 5490110.0
Yüksek benzerlik oranı: 12.64%
Ortalama benzerlik: 0.1113
Medyan benzerlik: 0.0476
```





- Yüksek benzerlikli çiftlerin oranı
- En yüksek benzerlikli ilk 3 çiftin kısa özeti

```python
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

def generate_similarity_report(input_file, output_file, threshold=0.3):
    df = pd.read_csv(input_file, header=None)
    texts = df[0].astype(str).tolist()
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(texts)
    similarity_matrix = cosine_similarity(tfidf_matrix)
    n = similarity_matrix.shape[0]
    matches = []
    with open(output_file, "w", encoding="utf-8") as f:
        f.write("Essay1_ID,Essay2_ID,Similarity,Essay1_Text,Essay2_Text,Common_Words\n")
        for i in range(n):
            for j in range(i+1, n):
                sim = similarity_matrix[i, j]
                if sim > threshold:
                    words1 = set(texts[i].lower().split())
                    words2 = set(texts[j].lower().split())
                    common_words = words1.intersection(words2)
                    f.write(f"{i},{j},{sim:.2f},\"{texts[i][:100]}...\",\"{texts[j][:100]}...\",\"{', '.join(list(common_words)[:10])}\"\n")
                    matches.append((i, j, sim, texts[i][:100], texts[j][:100]))
    print(f"Detaylı eşleşme raporu '{output_file}' olarak kaydedildi.")
    print(f"\nToplam {len(matches)} çiftte %{int(threshold*100)}'dan yüksek benzerlik bulundu.")
    if matches:
        print("\nÖrnek eşleşmeler:")
        for i, j, sim, t1, t2 in matches[:3]:
            print(f"Essay {i} & Essay {j} - Benzerlik: {sim:.2f}")
            print(f"Essay {i}: {t1}...")
            print(f"Essay {j}: {t2}...\n")

if __name__ == "__main__":
    generate_similarity_report("essays_lemmatized.csv", "similarity_report_lemmatized.csv")
    generate_similarity_report("essays_stemmed.csv", "similarity_report_stemmed.csv")
```

```
Detaylı eşleşme raporu 'similarity_report_lemmatized.csv' olarak kaydedildi.

Toplam 4354445 çiftte %30'dan yüksek benzerlik bulundu.

Örnek eşleşmeler:
Essay 0 & Essay 57 - Benzerlik: 0.30
Essay 0: life filled brim better day want limiting car usage advantage car always tense im much happier way s...
Essay 57: turning one best time life get license new car dont rely parent take every instead getting new car g...

Essay 0 & Essay 121 - Benzerlik: 0.30
Essay 0: life filled brim better day want limiting car usage advantage car always tense im much happier way s...
Essay 121: world war ii start development centering car recent year popularity well traffic gone limiting car u...

Essay 0 & Essay 126 - Benzerlik: 0.30
Essay 0: life filled brim better day want limiting car usage advantage car always tense im much happier way s...
Essay 126: ford volkswagen kia chevys car brand may massive part american culture significance life may declini...

Detaylı eşleşme raporu 'similarity_report_stemmed.csv' olarak kaydedildi.

Toplam 5485450 çiftte %30'dan yüksek benzerlik bulundu.

Örnek eşleşmeler:
Essay 0 & Essay 23 - Benzerlik: 0.34
Essay 0: life fill brim better day want limit car usag advantag car alway tens im much happier way said heidr...
Essay 23: limit car use could great way enjoy natur life walk great way work even ride bike around citi town g...

Essay 0 & Essay 28 - Benzerlik: 0.31
Essay 0: life fill brim better day want limit car usag advantag car alway tens im much happier way said heidr...
Essay 28: wouldnt great hear citizen limit car usag kind make world better place instead drive place place par...

Essay 0 & Essay 46 - Benzerlik: 0.31
Essay 0: life fill brim better day want limit car usag advantag car alway tens im much happier way said heidr...
Essay 46: ever realiz drive everywher isnt alway best way get place well advantag limit car usag fellow citize...
```

```python
    if matches:
        # Benzerlik skorlarına göre eşleşmeleri sıralama
        sorted_matches = sorted(matches, key=lambda x: x[2], reverse=True)

        print("\nÖrnek eşleşmeler (En yüksek benzerlikten en düşüğe):")
        for i, (i_idx, j_idx, sim, t1, t2) in enumerate(sorted_matches[:5]):  # İlk 5 örnek eşleşme
            print(f"Essay {i_idx} & Essay {j_idx} - Benzerlik: {sim:.2f}")
            print(f"Essay {i_idx}: {t1}...")
            print(f"Essay {j_idx}: {t2}...\n")

if __name__ == "__main__":
    generate_similarity_report("essays_lemmatized.csv", "similarity_report_lemmatized.csv")
    generate_similarity_report("essays_stemmed.csv", "similarity_report_stemmed.csv")
```

```
Detaylı eşleşme raporu 'similarity_report_lemmatized.csv' olarak kaydedildi.

Toplam 4354445 çiftte %30'dan yüksek benzerlik bulundu.

Örnek eşleşmeler (En yüksek benzerlikten en düşüğe):
Essay 6042 & Essay 8902 - Benzerlik: 1.00
Essay 6042: driverless car becoming future many different auto maker creating concept selfdriving car many peopl...
Essay 8902: driverless car becoming future many different auto maker creating concept selfdriving car many peopl...

Essay 2218 & Essay 3461 - Benzerlik: 1.00
Essay 2218: ever mistaken sight something well thats problem people thinking face mar natural landform trust evi...
Essay 3461: ever mistaken sight something well thats problem people thinking face mar natural landform trust evi...

Essay 1603 & Essay 3490 - Benzerlik: 1.00
Essay 1603: face mar gotten lot fame past year since lot fame lot people know including conspiracy theorist theo...
Essay 3490: face mar gotten lot fame past year since lot fame lot people know including conspiracy theorist theo...

Essay 1625 & Essay 3534 - Benzerlik: 1.00
Essay 1625: think face artificial structure created alien well scientist employed nasa seen history space progra...
Essay 3534: think face artificial structure created alien well scientist employed nasa seen history space progra...

Essay 1786 & Essay 3287 - Benzerlik: 1.00
Essay 1786: thirtyeight year old woman scientist nasa recently discovered landform look like real face mar bump ...
Essay 3287: thirtyeight year old woman scientist nasa recently discovered landform look like real face mar bump ...
```

Bu metinler **birebir aynı** → muhtemelen kopya içerikler.

Similarity_report dosyaları:

similarity_report_lemmatized.csv , similarity_report_stemmed.csv dosyaları drive link:

https://drive.google.com/drive/folders/10vPwsgcacFMYNZz6FNohfnyNWizW1dQL?usp=sharing

8. Sonuç ve Değerlendirme

Bu ödev kapsamında, metin tabanlı bir veri seti üzerinde doğal dil işleme teknikleri uygulanarak metinlerin sayısal temsilleri oluşturulmuş ve çeşitli modelleme yöntemleriyle semantik analiz gerçekleştirilmiştir.

Veri ön işleme sürecinde stop word çıkarımı, küçük harfe dönüştürme, lemmatizasyon, stemming ve özel karakter temizliği gibi adımlar başarıyla uygulanmıştır. Bu adımlar sayesinde verideki anlamsal gürültü azaltılmış ve modelleme süreci için daha homojen bir veri kümesi elde edilmiştir.

Zipf yasası analizinde elde edilen grafikler, veri setinin doğal dil özelliklerini taşıdığını ve istatistiksel analiz için uygun olduğunu ortaya koymuştur. Bu da modelleme adımlarına geçişin mantıklı ve temellendirilmiş olduğunu göstermektedir.

TF-IDF ve Word2Vec gibi vektörleştirme yöntemleri ile metinler sayısal forma dönüştürülmüş ve özellikle Word2Vec modelleri farklı parametreler altında test edilerek performans karşılaştırmaları yapılmıştır. Yapılan analizler, **Skip-gram**, **lemmatization**, **window=2** ve **dim=100** parametreleriyle oluşturulan modelin, anlamsal benzerlik açısından en başarılı sonuçları verdiğini göstermektedir. Bu model, anlam bütünlüğünü koruyan kelimeler arasında yüksek benzerlik skorları üretmiştir.

Stemming tabanlı modellerde kök bozulmalarının semantik kaliteyi düşürdüğü gözlemlenmiştir. CBOW modelleri ise daha genelleştirici olsa da, bağlamsal ilişkilerde zayıf kalmıştır.

Sonuç olarak, ödevde izlenen yöntemsel yapı hem teorik hem de uygulamalı açıdan yerinde olup, doğal dil işleme projelerinde temel bir yaklaşımın nasıl kurulacağını göstermektedir. Model başarılarının parametre kombinasyonlarına bağlı olarak değiştiği net şekilde ortaya konulmuştur. Bu da, ileride yapılacak daha büyük ölçekli projelerde model seçiminde ve veri hazırlama sürecinde dikkat edilmesi gereken unsurlar hakkında değerli bir öngörü sağlamaktadır.